SMath Studio mit Maxima

Handbuch zur FHB-Distribution

Prof. Dr.-Ing. Martin Kraska
kraska@fh-brandenburg.de

3. August 2014

Fachhochschule
Brandenburg
University of
Applied Sciences

Inhaltsverzeichnis

1. Einleitung

Dieses Handbuch soll sowohl die Einarbeitung in als auch die Anwendung von SMath unterstützen. Es befindet sich noch im Aufbau, daher sind einige Abschnitte nicht so ausführlich, wie sie sein sollten und einige Themen fehlen auch komplett.

Das Handbuch bezieht sich auf die portable Distribution „SMath mit Maxima", wie sie vom Autor zusammengestellt wurde (Download) Die an der Fachhochschule Brandenburg entwickelte Maxima-Schnittstelle erweitert SMath Studio um leistungsfähige Computer-Algebra-Funktionen. So ermöglicht sie z.B. die symbolische Integration und Grenzwertberechnung. Zudem erschließt sie die Grafikmöglichkeiten von Gnuplot für den SMath-Anwender.

Kapitel 1 ordnet SMath in die Welt der Mathematikprogramme ein und gibt Hinweise zum Leistungsumfang.

Kapitel 2 führt in einer Schritt-für-Schritt-Anleitung durch die Installation und die Bedienung des Programms. Man kann sie in 1-2 Stunden durcharbeiten.

Kapitel 3 beschreibt den Umgang mit der Benutzeroberfläche und dokumentiert die Menüstruktur sowie die Bereichstypen, also Text, Formeln, Diagramme in SMath-Dokumenten (Rechenblättern). Zudem werden die Import- und Export-Möglichkeiten dokumentiert.

Kapitel 4 beschreibt die mathematischen Fähigkeiten von SMath. Hier sind auch die auf Maxima basierenden Funktionen beschrieben.

Kapitel 5 dokumentiert die Diagrammfunktionen, die vom Maxima-Plugin bereitgestellt werden sowie den eingebauten 2D-Diagrammbereich mit seinen Animationsfunktionen.

Kapitel 6 dokumentiert den Umgang mit Maßeinheiten. Eine Auflistung aller in SMath vordefinierter Maßeinheiten und Konstanten findet sich in Anhang C.

Kapitel 7 behandelt die Möglichkeiten zur Definition eigener Funktionen und Algorithmen in SMath-Arbeitsblättern. Abschnitt 7.5 zeigt, wie SMath-Programme als Bausteine in anderen Arbeitsblättern verwendet werden können. Der Umgang mit Zeichenketten wird in Abschnitt 7.4 erläutert.

Seit der Version 0.96 (Mai 2013) kann SMath Studio selbstständige Anwendungen exportieren, die unter dem Namen *SMath Viewer* laufen. Kapitel 8 erläutert dies näher.

Kapitel 9 behandelt den Erweiterungsmanager und externe Plugins, mit denen der Funktionsumfang von SMath gegenüber der Standardinstallation erweitert werden kann.

Das letzte Kapitel enthält Anwendungsbeispiele und Hinweise zur Verwendung des Programms bei der Dokumentation technischer Berechnungen, darunter auch Mechanik-Aufgaben.

Die Anhänge enthalten Referenzmaterial wie z.B.

- Liste der Operatoren und deren Tastatureingabe
- Tastenbelegung
- Vordefinierte Maßeinheiten und Konstanten

- Beschreibung zu fast allen Funktionen
- Details zur FHB-Distribution von SMath Studio mit Maxima
- „Dachboden" mit historischen Abschnitten zu Funktionen, die keine wesentliche Rolle mehr in der Anwendung von SMath Studio spielen.

Abgeschlossen wird das Handbuch durch ein Literatur- und ein Stichwortverzeichnis.

Mit Kritik und Anregungen können Sie sich gern per email an den Autor wenden: kraska@fh-brandenburg.de

1.1. Was ist SMath Studio?

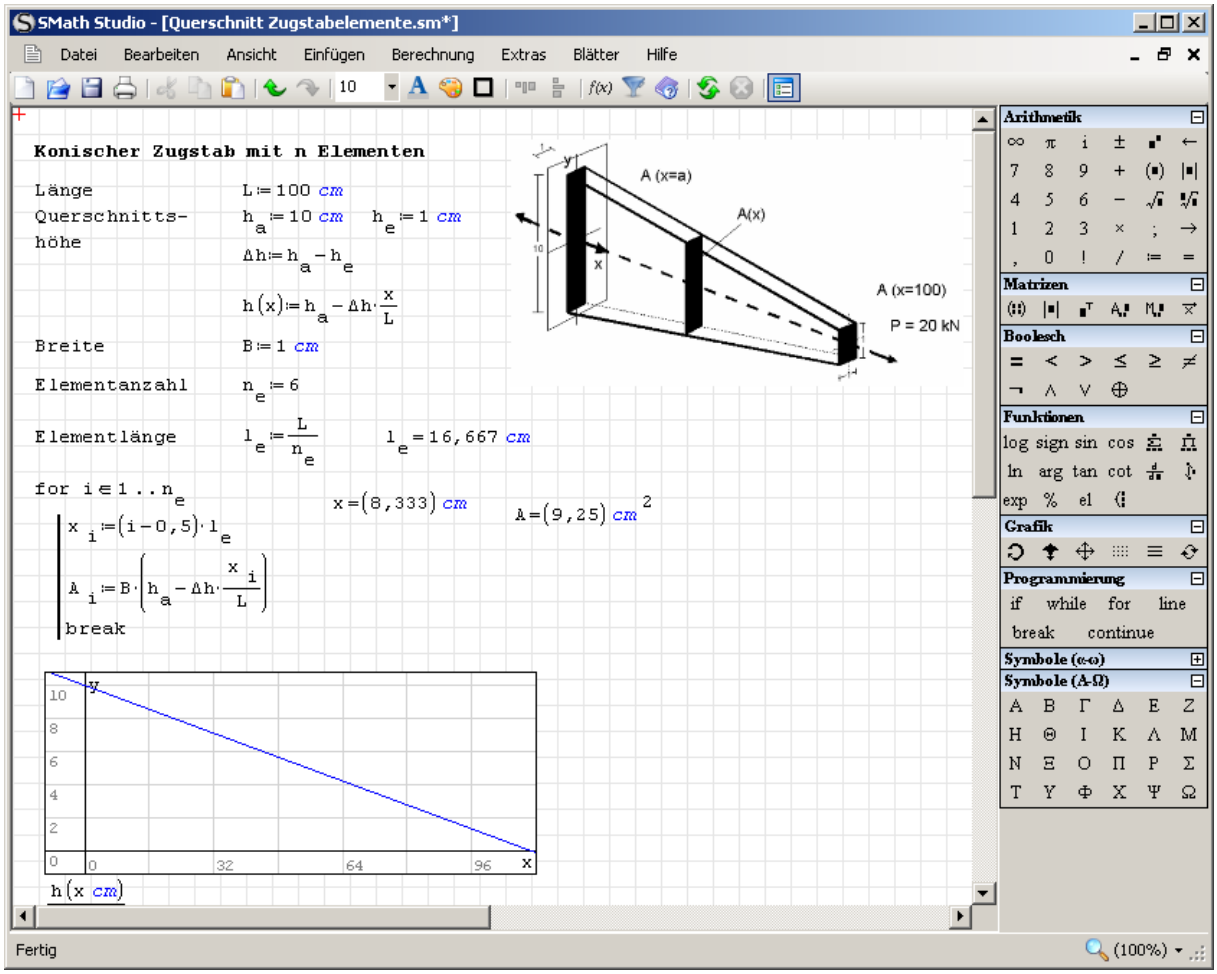


Abbildung 1.1.: Benutzeroberfläche von SMath

SMath Studio ist freies, allerdings nicht quelloffenes Programm zur Durchführung und Dokumentation technischer Berechnungen. Es stellt ein elektronisches Rechenpapier bereit, auf welchem man Formeln, Kommentare und Diagrammbereiche frei plazieren kann. Das Bedienkonzept orientiert sich an Mathcad, dem Klassiker auf diesem Gebiet. Mit diesem teilt SMath Studio die folgenden Vorteile:

- Der Formelsatz entspricht der gewohnten mathematischen Notation. Damit kann auch jemand, der das Programm nicht kennt, die damit erstellten Rechenblätter verstehen und nachvollziehen.

- Ausgezeichnete Bedienbarkeit mit der Tastatur (schnelles Editieren)
- Leistungsfähiger Umgang mit Maßeinheiten

Einschränkungen gegenüber Mathcad gibt es natürlich:

- Weniger Formatierungsmöglichkeiten (Zahlendarstellung, Textbereiche)
- Rudimentäre Dokumentation (das vorliegende Handbuch versucht hier, etwas zu helfen)
- Keine impliziten Schleifen mit Bereichsvariablen, Schleifen müssen immer explizit geschrieben werden.
- Schwache Editierfunktionen für Matrizen,
- Keine Lösungsblöcke (aber leistungsfähige numerische und symbolische Löser),
- Geringerer mathematischer Funktionsumfang

Es ist nicht quelloffen, aber kostenlos. Entwickler ist der Sankt Petersburger Andrey Ivashov. Das Programm hat verschiedene Erweiterungsschnittstellen, über die neue Funktionen bereitgestellt werden können. Diese werden in der Regel im SMath-Forum vorgestellt und diskutiert.

Die vorliegende Einführung ist noch im Aufbaustadium. Eine sehr ausführliche, allerdings englischsprachige Anleitung findet man unter Interaktives Rechenblatt zur Einführung in SMath.

1.2. Was ist Maxima?

Maxima ist ein quelloffenes und freies Computer-Algebra-Programm. Es hat eine Kommandozeilen-Schnittstelle, die zugehörigen Dokumente ähneln Programmquelltexten und sind in der Regel für Außenstehende nicht verständlich. Das Programm basiert auf dem am MIT entwickelten Macsyma und wird bis heute aktiv weiterentwickelt.

In der FHB-Distribution von SMath Studio ist ein komplettes Maxima enthalten und wird über eine lokale IP-Verbindung (Socket) in SMath Studio eingebunden.

1.3. Funktionsüberblick

Merkmale der Basisversion:

- Numerische und symbolische Berechnungen mit reellen und komplexen Zahlen
- Symbolische Differentiation (Ableitungsbildung)
- Numerische Integration (bestimmte Integrale berechnen)
- Viele Berechnungen können mit Maßeinheiten durchgeführt werden.
- Matrizenrechnung (Addition, Multiplikation, Inversion, Determinanten), allerdings keine Eigenwerte und Eigenvektoren).
- Lösen von linearen und nichtlinearen Gleichungen und Gleichungssystemen (Nullstellensuche)
- Einfache 2D-Diagramme mit Animationsfunktion

- Programmierung mit benutzerdefinierten Funktionen, Schleifen und Anweisungsblöcken
- dynamische Auswahl für das Einfügen von Einheiten, Variablen- und Funktionsnamen sowie Code-Bausteinen
- Eingebautes mathematisches Nachschlagewerk
- Import von ASCII-Daten

Über Plugins bereitgestellt werden (Auswahl):

- Computer-Algebra-Funktionen und Diagramme mit Gnuplot (*Maxima*). Im Grunde ist der gesamte Funktionsumfang von Maxima verfügbar.
- Komfortable numerische Lösungsverfahren (*Nonlinear Solvers*)
- Interaktive Bedienelemente wie Auswahllisten, Schieberegler, Schaltflächen
- Dynamische Anzeige von Bilddateien und Matrizen (*ImageRegion*)

Prüfen Sie stets kritisch Ihre Arbeitsergebnisse. Wie jede Software enthält SMath Studio auch Fehler. Sie benutzen das Programm und die FHB-Distribution auf eigene Gefahr.

1.4. SMath im Internet

Wenn Sie nach dem Programm im Internet suchen, verwenden Sie am besten den Suchbegriff „SMath Studio", sonst kommen zuviele irrelevante Ergebnisse. Die offizielle Seite ist smath.info, die vom Programm-Entwickler und -Besitzer Andrey Ivashov auf einem eigenen Server gehostet wird.

1.4.1. SMath Forum

Es gibt zwei Foren für die Benutzergemeinde, ein englisches und ein russisches. Im englischen Forum können Sie an den Entwickler spenden (*donate*).

Wenn man selbst Fragen stellen oder anderen helfen will, dann muss man sich registrieren (Angabe der email reicht).

Hier findet man die aktuellen Programmversionen zum Download.

Das Wiki ist eine Sammlung von Anleitungen zu einzelnen Themen, die zum Teil über den Inhalt dieser Anleitung hinausgehen. Allerdings ist vieles nicht auf dem neuesten Stand.

Hier geht es zum SVN-Server, auf dem Quelltext und zum Teil auch kompilierte Versionen externer Plugins liegen. Dieser Bereich ist für Nutzer interessant, die an der Entwicklung mitarbeiten wollen. An den eigentlichen Programmcode kommt man aber nicht heran, denn wie bereits erwähnt, ist SMath kein quelloffenes Programm.

Die Foren bieten eine Live-Funktion, in der Sie das Programm ohne lokale Installation ausprobieren können. Es gibt allerdings Einschränkungen:

- Das Dezimaltrennzeichen ist der Punkt und das Argumenttrennzeichen in Funktionen ist das Komma. In der vorliegenden Anleitung werden stets das Komma als Dezimaltrenner und das Semikolon als Argumenttrenner verwendet.
- Für die Definition eigener Funktionen, Anweisungsblöcke und Schleifen muss man die internen Namen der entsprechenden Befehle kennen.
- Plotfunktionen stehen nicht zur Verfügung.
- Maßeinheiten funktionieren nicht.
- Textbereiche funktionieren nicht.

Eine verbesserte Live-Version von SMath Studio ist Basis der *Knovel Interactive Equations* (siehe Abschnitt 1.6).

Hier kommt man in den Bereich der Sprachanpassung. Nutzer können für Texte in der Benutzeroberfläche Vorschläge machen. Dieser Mechanismus macht es möglich, dass SMath in sehr vielen Sprachversionen verfügbar ist. Ein Hilfe- und Dokumentationssystem ist darin aber nicht inbegriffen.

Seit März 2014 gibt es ein System zur Verwaltung von Fehlermeldungen und Anwenderwünschen (*bug tracking system*). Um dort Inhalte einzustellen, benötigt man ein separates Login. Dort wird eine Liste von Fehlermeldungen und Anwenderwünschen gepflegt. Sie ist in die Projekte

- *SMath Studio* (das Programm zur Erstellung von Rechenblättern)
- *SMath Viewer* (Ausführung compilierter Rechenblätter als eigenständige Anwendung, Details dazu im Kapitel 8)

unterteilt.

1.4.2. Seiten von SMath Anwendern

Hier findet man eine Liste von SMath-Anleitungen in Englisch und Spanisch.

Gilberto Urroz, Utah State University

Inoffizielles russisches Handbuch

1.5. SMath unter Linux

Die FHB-Distribution ist grundsätzlich auch unter Linux lauffähig, allerdings ohne die Maxima-Funktionen und ohne dass dies getestet wurde. Unter Linux ist Mono, eine plattformübergreifende, quelloffene .NET-Umgebung, erforderlich. Auf den meisten Linux-Distributionen

muss die aktuellste Version von Mono nachinstalliert werden. Dann kann SMath mit folgender Kommandozeile gestartet werden:

```
mono SMathStudio_Desktop.exe
```

Einige Anwender verpacken dies noch in ein Startskript, welches eventuelle Parameter (zu öffnende Dateien) durchreicht.

1.6. Knovel Interactive Equations

Knovel ist ein e-Book-Anbieter für den technischen Bereich mit interaktiven Zusatzfunktionen wie Volltextsuche oder Diagramm-Digitalisierung, Seit März 2014 gibt es ein weiteres Werkzeug, die sogenannten Knovel Interactive Equations, welches im Wesentlichen eine verbesserte Version von SMath Life ist. Der Zugriff auf diese Version ist kostenlos, erfordert aber eine vorherige Registrierung.

Das Produkt ist noch etwas hakelig, so dass es im Vergleich zu einer lokalen SMath-Studio-Installation nur wenig Spaß macht (Klammer auf, Hochkomma - wichtig für Maßeinheiten - funktionieren nicht, beim Schnellschreiben werden Zeichen verschluckt).

Allerdings finden sich in der ohne Registrierung zugänglichen Hilfe unter HELP gut gemachte Videos und ein Einsteigerhandbuch, die auch gut für SMath Studio selbst geeignet sind.

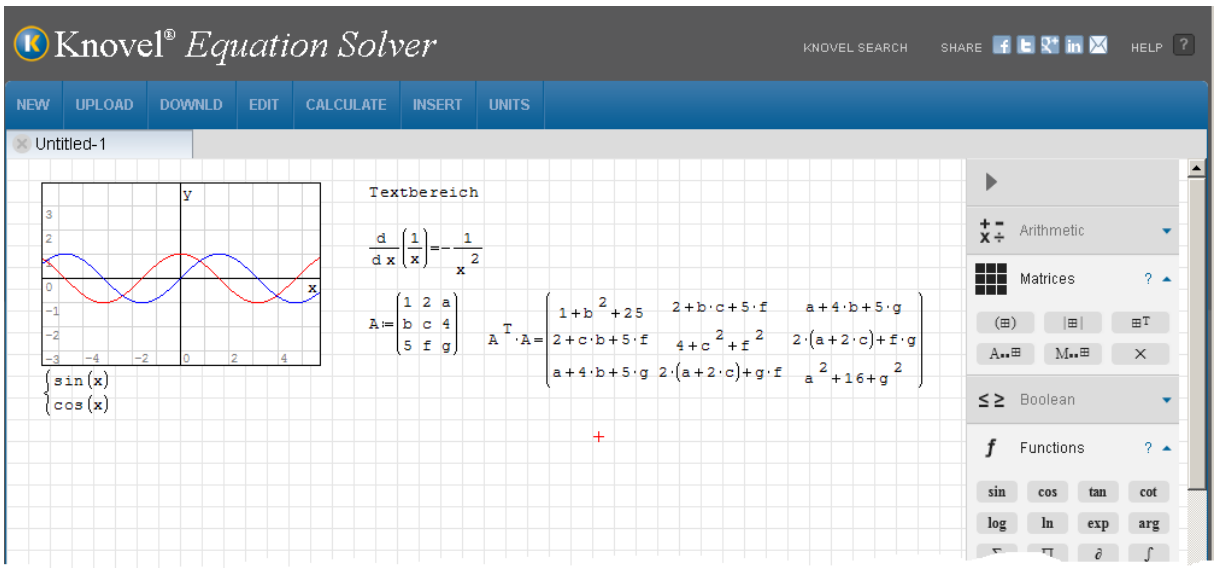


Abbildung 1.2.: SMath Studio, eingebettet als Webanwendung bei Knovel.

1.7. Alternativen

Hier werden einige Mathematikprogramme aufgeführt, deren Funktionsumfang sich mit dem von SMath Studio überschneidet.

Mathcad ist das Original, von dem die Entwicklung von SMath Studio inspiriert ist. Die Vollversion ist für 120€ als Studentenversion (unbegrenzte Dauer) und für kommerzielle Anwendung für ca. 1700€ verfügbar (Stand März 2013). Es gibt auch die kostenlose Version Mathcad Express, die auf die Basisfunktionen beschränkt ist. Gegenüber der Vollversion fehlen beispielsweise:

- Programmierfunktionen
- 3D-Diagramme
- Lösungsblöcke
- Symbolische Mathematik
- Viele Funktionsbibliotheken
- Viele numerische Verfahren

Seit der Verfügbarkeit der Maxima-Schnittstelle kann SMath Studio gut mit der Vollversion mithalten. Es fehlen hauptsächlich die Lösungsblöcke und Bereichsvariable.

MathStudio verfügbar für iOS und OS-X ($20) und Android ($15). Dies ist eine kommandozeilenorientierte Anwendung mit großem Funktionsumfang und guten Grafikfunktionen.

Vorteil:

- Version für iOS und Android verfügbar
- Umfangreiche Grafik- und Rechenfunktionen

Nachteile:

- kein 2D Dokumentformat
- keine Einheitenverwaltung

TI Nspire CAS Taschenrechner mit Computeralgebrasystem, basierend auf dem nicht mehr erhältlichen Derive. Das System gibt es auch als Software für PCs und als iOS-App.

Vorteile:

- Besserer Funktionsumfang und Reifegrad
- Interaktive Grafik (Geometrie und Diagramme)

Nachteile:

- kein 2D Dokumentformat.

Maxima ist das beste quelloffene Computeralgebrasystem derzeit.

Vorteile:

- Quelloffen, aktive Benutzer- und Entwicklergemeinschaft
- Besserer Funktionsumfang und Reifegrad

Nachteile: kein 2D Dokumentformat.

Durch die Integration mit SMath Studio werden die Vorteile genutzt und der Nachteil ausgeglichen.

Giac/Xcas: Giac ist eine quelloffene Bibliothek für CAS-Funktionen, Xcas ist eine Benutzeroberfläche dafür. Die Bibliothek ist Grundlage der CAS-Funktionen des Taschenrechners HP50.

Vorteile (nur im Vergleich zu SMath Studio ohne Maxima):

- Quelloffen, aktive Benutzer- und Entwicklergemeinschaft
- Besserer Funktionsumfang und Reifegrad

Nachteile: kein 2D Dokumentformat.

Miramath Ein Nachbau von MathCAD, basierend auf numerischen und symbolischen Python-Bibliotheken. Für den Download muss man sich registrieren. Das Programm wurde vom Autor dieser Anleitung nicht getestet. Nur unter Linux verfügbar.

Vorteile:

- 2D Dokumentformat
- Hochwertiger Formelsatz

Nachteile:

- Nicht quelloffen
- Keine Maßeinheitenbehandlung
- Entwicklungsintensität unklar

CompPad ist eine LibreOffice-Erweiterung, die Formeln aktiv auswertet.

Vorteile:

- Direkte Integration in LibreOffice
- Maßeinheitenbehandlung (nicht ganz so leistungsfähig wie in SMath, insbesondere sind Einheitensymbole nicht von Variablen getrennt/unterscheidbar)

Nachteile:

- Plotfunktion funktioniert nicht richtig (Stand Januar 2014)
- Geringer Funktionsumfang
- Minderwertiger Formelsatz

Microsoft Mathematics 4.0 gibt es als eigenständige Anwendung und als Add-In für den Formeleditor in Microsoft Word.

Vorteile:

- Nahtlose Integration in Microsoft Word (Add-in)

Nachteile:

- keine formelübergreifende Definition von Funktionen und Variablen (Add-In)
- Keine Maßeinheiten in Formeln

RedCrab ist ein Mathematikprogramm, welches ähnlich wie Mathcad und SMath ein 2D Dokumentenformat bietet. Es ist Freeware, jedoch nicht quelloffen und es gibt eine etwas erweiterte Shareware-Version.

Vorteile:

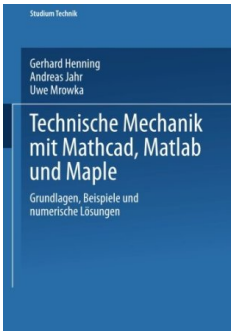
- 2D-Dokumentformat
- Handbuch, Einführungsvideos

Nachteile:

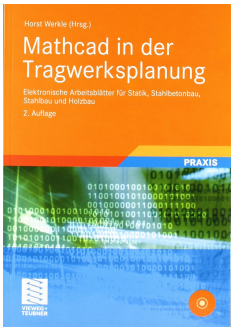
- keine Maßeinheitenverwaltung (Ergebnisse können manuell mit Text ergänzt werden)
- keine CAS-Funktionen

1.8. Literatur

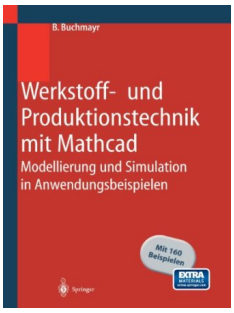
Es gibt viele Bücher zum Einsatz von Mathcad und dialogorientierten Computeralgebra-Systemen für technische Berechnungen.



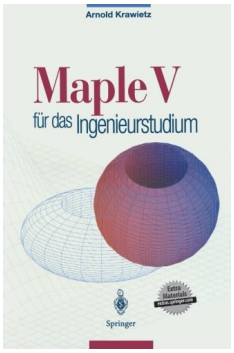
Henning/Jahr/Mrowka (2004) Entstanden an der Fachhochschule Düsseldorf zu einem Wahlpflichtkurs „Angewandte Mechanik”.



Werkle (2012) Der Herausgeber unterrichtet Baustatik an der HTWG Konstanz.



Buchmayr (2002) Numerische Verfahren bis hin zu einem FEM-Programm für Scheiben und einem Werkstoffdatenbank-System, Materialdatenanpassung, Prozessmodelle



Krawietz (1997) Mathematisch sehr weitreichende Behandlung mit Beispielen aus der Mechanik. Der Autor hat an der Technischen Fachhochschule Berlin Mechanik unterrichtet.

2. Erste Schritte

Dieses Kapitel wurde bereits im Unterricht an der Fachhochschule Brandenburg erprobt und kann in ein bis zwei Stunden durchgearbeitet werden.

2.1. Download und Installation

Diesem Handbuch liegt die FHB-Distribution von SMath Studio mit Maxima zugrunde. Vorteile gegenüber der offiziellen SMath-Distribution sind:

- Vorinstallation der meisten externen Plugins
- Vorinstallation des interaktiven Handbuchs (ca. 500 bilinguale (deutsch/englisch) verlinkte Hilfeseiten enthält.
- Einsatzbereite Maxima-Installation mit Anbindung an SMath Studio (für Computer-Algebra-Fähigkeiten und flexiblere Grafik)

Schritte für die Installation:

- Entpacken Sie die Datei in einem beliebigen Speicherort.
- Optional: Erklären Sie die ausführbare Datei *SMathStudio_Desktop.exe* zum Standardprogramm für .sm Dateien. Dann können Sie im Dateimanager SMath-Dateien mit Doppelklick öffnen.
  - Rechtsklick auf *welcome.sm* (oder irgend eine andere .sm-Datei)
  - Öffnen mit> Standardprogramm auswählen
  - Durchsuchen und *SMathStudio_Desktop.exe* im Installationsverzeichnis auswählen.
  - Wenn Sie SMath Studio auf einem USB-Stick installiert haben, müssen Sie diese Zuordnung auf dem jeweiligen Rechner neu ausführen.
- Optional: Wählen Sie Ihr Lieblings-Zeichenprogramm (z.B. Paint) als Standardprogramm für .png-Bilddateien. Dann können Sie aus SMath-Studio heraus Bilder erzeugen und editieren.
- Stellen Sie die Sprache der Benutzeroberfläche auf deutsch (voreingestellt ist englisch):

  Hauptmenü> Tools> Options> Interface language „Deutsch" (Hauptmenü> Extras> Einstellungen> Sprache)

2.2. Speichern und Rückgängig machen

SMath speichert Rechenblätter als xml-Dateien mit der Endung .sm. Legen Sie zu Beginn der Arbeit mit **Hauptmenü> Datei> Speichern unter** den Dateinamen fest und speichern Sie regelmäßig (am bequemsten mit Strg - S , das heißt: Strg und S gleichzeitig drücken).

Rückgängig machen per Strg-Z und Wiederherstellen per Strg-Y funktionieren in den meisten Fällen, aber maximal bis zum letzten Speicherpunkt.

Häufig ist es einfacher, eine Eingabe durch Strg-Z rückgängig zu machen, als das letzte eingegebene Zeichen zu löschen, besonders bei der Eingabe von Formeln.

2.3. SMath als Taschenrechner

Wenn Sie einfach lostippen, entsteht an der Stelle der Einfügemarke + ein Formelbereich. In Formelbereichen können Sie Berechnungen durchführen und Funktionen und Variablen definieren.

Beispiele :

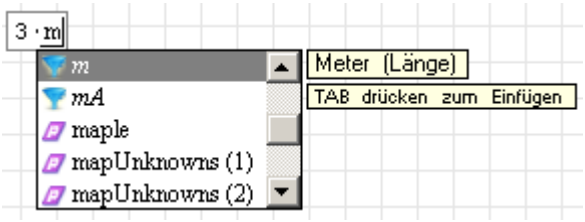
Tastatureingabe	Ergebnis
3+3=	3+3=6
3^3=	$3^3=27$
cos(p Strg-G =	$\cos(\pi)=-1$
\3=	$\sqrt{3}=1,73$

Die Zahl der angezeigten Nachkommastellen ist einstellungsabhängig, darauf kommen wir später noch zurück.

In den Platzhalter (■) hinter dem Ergebnis kann eine Maßeinheit oder sonst ein Faktor, den man im Ergebnis haben will, eingetippt werden.

SMath kann mit Maßeinheiten rechnen:

3m



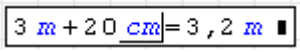
Sobald Sie einen Buchstaben in einem Formelbereich tippen, prüft SMath, ob es einen solchen Namen schon kennt. Es erscheint dann die dynamische Auswahl. Sie können weitere Buchstaben eintippen oder mit den Pfeiltasten aus der Liste auswählen. Mit ⇆ (Tabulator) wird bestätigt.

⇆

Maßeinheiten werden kursiv und blau dargestellt. Intern steht vor dem Einheitennamen ein Hochkomma (’). Das können Sie auch bei der Eingabe benutzen, um Maßeinheiten zu kennzeichnen.

Hier die Summe zweier Längen:

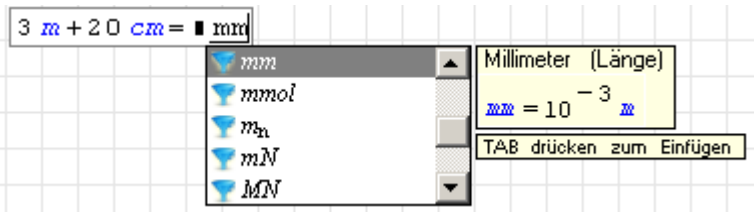
3m ⇆ +20cm ⇆ =



Die interne Darstellung diese Ausdrucks ist `3*'m+20*'cm=3.2*'m@#`

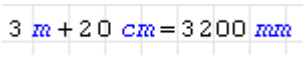
Um das Ergebnis in mm anzuzeigen, stellen Sie die Eingabemarke auf den Platzhalter hinter dem Ergebnis und tippen

mm



⇆

Die Berechnung erfolgt erst nach Verlassen des Formelbereichs (↵ drücken oder irgendwo anders im Rechenblatt klicken, dann verschwindet der Rahmen um die Formel):



Man kann Maßeinheiten auch über das Menü oder die Werkzeugleiste einfügen, siehe Abschnitt 6

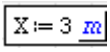
2.4. Bezeichner

Funktionen oder Variablen können einen Bezeichner (Namen) bekommen, der mit dem Zuweisungsoperator := definiert wird. Dieser ist einfach als Doppelpunkt einzutippen.

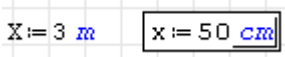
Groß- und Kleinschreibung zählt! `X` und `x` sind verschiedene Bezeichner!

Definitionen sind immer unterhalb und rechts ihres Bereiches bekannt (dabei zählt die linke obere Ecke als Position des Bereichs).

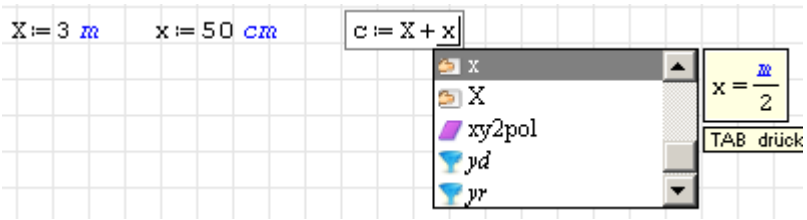
X:3m ⇆



x:50cm ⇆



c:a+b



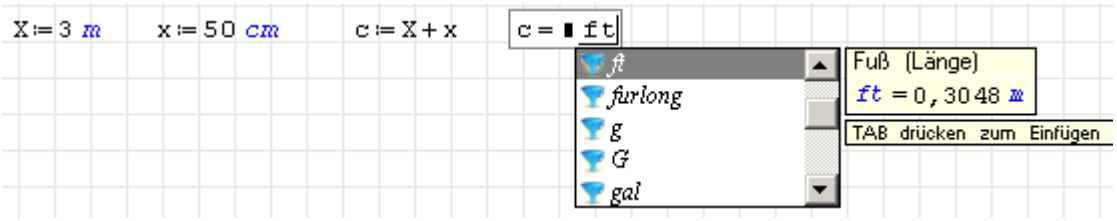
Die dynamische Auswahl kennt alle bereits definierten Bezeichner (Variablen, Funktionen, Maßeinheiten). Der Wert wird in einer symbolisch vereinfachten Form angezeigt. So kann man erkennen, was genau SMath als Definition gespeichert hat.

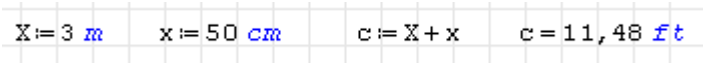
Den Zahlenwert einer Variablen zeigt man mit „=" an:

```
c=
```

Und auch hier kann wieder die Einheit geändert werden:

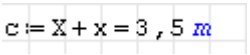
```
ft
```





Zugewiesene Werte können direkt angezeigt werden:

```
c:X+x=
```

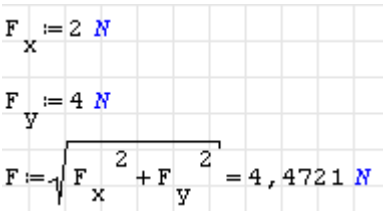


Bezeichner können auch mit einem tiefgestellten Namenszusatz (Textindex) versehen werden. Die Tiefstellung wird mit einem Punkt (.) eingeleitet, der als unsichtbares Zeichen Bestandteil des Namens bleibt. Die folgenden Eingaben sind jeweils in einem neuen Bereich zu machen.

```
F.x:2N
```

```
F.y:4N
```

```
F:\F.x^2 Leert. +F.y^2=
```

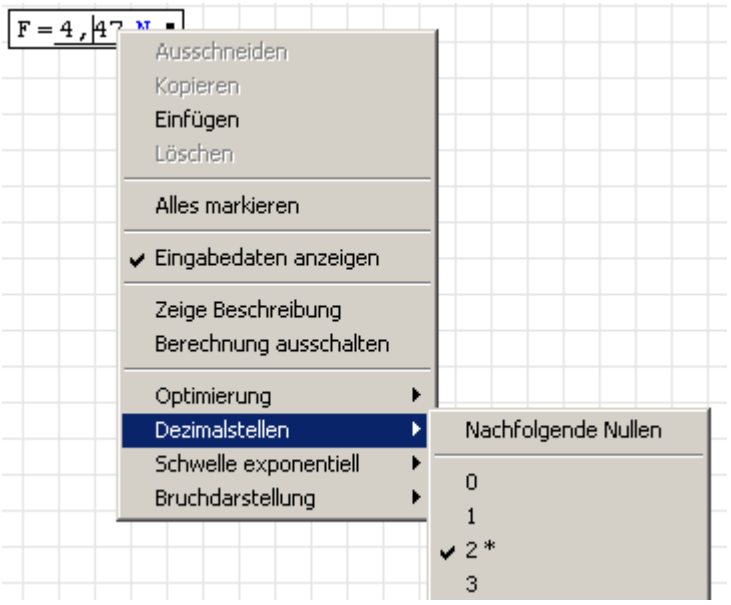


Verwenden Sie möglichst aussagefähige Variablennamen. Griechische Buchstaben erzeugen Sie, indem Sie nach einem lateinischen Buchstaben Strg-G drücken.

2.5. Ergebnisdarstellung und Runden

Die Anzahl angezeigter Nachkommastellen kann unter **Hauptmenü> Extras> Optionen> Berechnung** voreingestellt werden.

Alternativ kann die Einstellung für jedes Ergebnis im Kontextmenü (rechte Maustaste) geändert werden:



Man erkennt den aktuell eingestellten Wert (√) und den voreingestellten Wert (*)

Hier wurde der Wert „Dezimalstellen" auf 5 geändert:

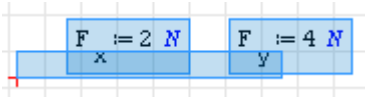
$F = 4,47214\ N$

Damit nachfolgende Nullen angezeigt werden, ist diese Option im Kontextmenü zu aktivieren.

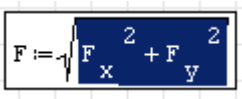
2.6. Kopieren und Verschieben

Sie können die Bereiche frei auf dem Rechenblatt anordnen. Sie werden immer in der Reihenfolge von links nach rechts und von oben nach unten ausgewertet.

Zum Kopieren, Verschieben oder Löschen ganzer Formeln zieht man mit der Maus ein Rechteckfenster über die zu markierenden Bereiche. Sie erscheinen dann in Hellblau:



Will man Teile einer Formel kopieren oder löschen, klickt man zunächst in die Formel und zieht mit der Maus eine Markierung auf:



Mit Strg - C kopiert man den markierten Bereich. Er kann auch außerhalb von SMath als Text eingefügt werden, so sieht man, wie er intern dargestellt wird:

```
F.x^2+F.y^2
```

2.7. Textbereiche

Das Rechenblatt kann mit Kommentaren versehen werden. Textbereiche erzeugt man durch Eintippen von Gänsefüßchen ".

```
"
```

Man kann einfach lostippen, am ersten Leerzeichen (nach den ersten Buchstaben) merkt SMath, dass man keine Formel, sondern Text schreibt. Das erkennt man an der dynamischen Sprachanzeige:

```
Textbereiche␣
```

Textbereiche | Deutsch

Textbereiche können mehrzeilig sein, der Zeilenumbruch erfolgt mit Strg - ↵ oder ⇑ - ↵ manuell. Einen automatischen Umbruch gibt es leider noch nicht.

```
Manueller␣ Strg - ↵  Zeilenumbruch
```

manueller
Zeilenumbruch

Textbereiche können nur als Ganzes formatiert werden. Am schnellsten erreichbar sind die Formatierungen **fett**, unterstrichen und *kursiv*, die mit Strg -Befehlen ein- und ausgeschaltet werden:

Normal **Strg-b** Strg-u *Strg-i*

Hinweis: Wenn das nicht funktioniert, haben Sie möglicherweise einen Formelbereich erzeugt. Wenn Sie ein Wort, z.B. Strg tippen und dann ein Minus, dann denkt SMath, dass Sie eine Formel schreiben wollen. Dann nutzt auch ein Leerzeichen nicht mehr. Dann müssen Sie einen neuen Bereich anfangen, hier z.B. mit

```
"Strg-b
```

Strg-b

Zum Vergleich der Formelbereich, der Unterschied ist schwer zu erkennen, das Minus ist länger und hat etwas mehr Platz. Hier fügen wir erst einen Textbereich ein und dann eine Formel:

```
"Strg-b ↵ Strg-b
```

Strg-b
Strg−b

Ergebnisse (und überhaupt alle Bereiche) kann man mit  (Werkzeugleiste) einrahmen. Dieser Rahmen bleibt auch, wenn man den Bereich verlässt.

$F = 4,47\ N$

Weitere Formatierungsmöglichkeiten finden sich in der Werkzeugleiste am oberen Rand des Rechenblatts.

- Schriftfarbe 
- Schriftgröße 
- Hintergrundfarbe 

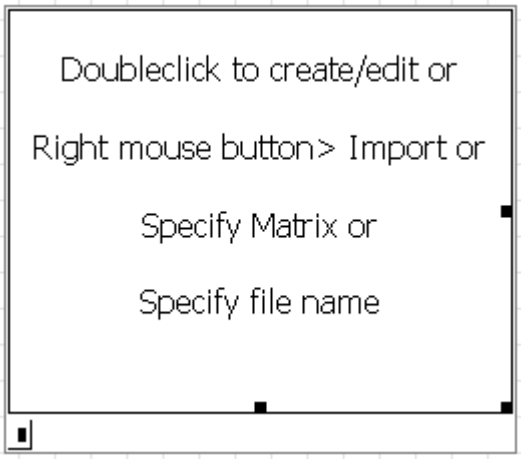
Diese Optionen sollten vorsichtig benutzt werden, denn das Rechenblatt ist schnell durch inkonsistente Formatierung verhunzt.

2.8. Bilder einfügen

Häufig ist es sinnvoll, Bilder zur Erläuterung der Rechnung einzufügen. Sie können Bilder aus der Zwischenablage einfügen oder aus SMath heraus erzeugen. Das geht so:

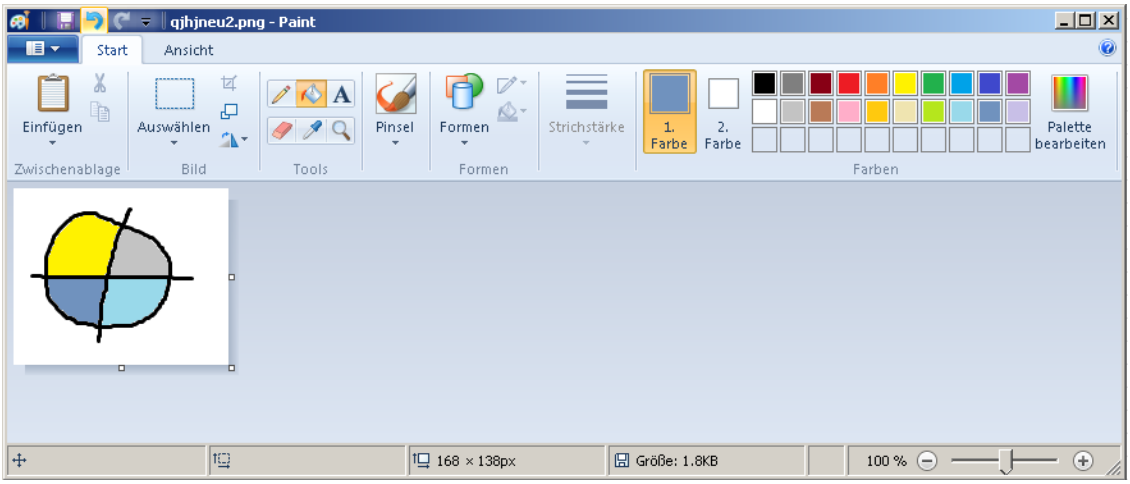
Fügen Sie zunächst einen Image-Bereich ein:

Hauptmenü> Einfügen> Image



Doppelklick auf den Bereich:

Es öffnet sich das Standardprogramm für die png-Bearbeitung. Ziehen Sie zunächst die Zeichenfläche auf die gewünschte Größe und zeichen Sie das Bild.



Schließen Sie das Programm und bestätigen Sie die Speicherung.

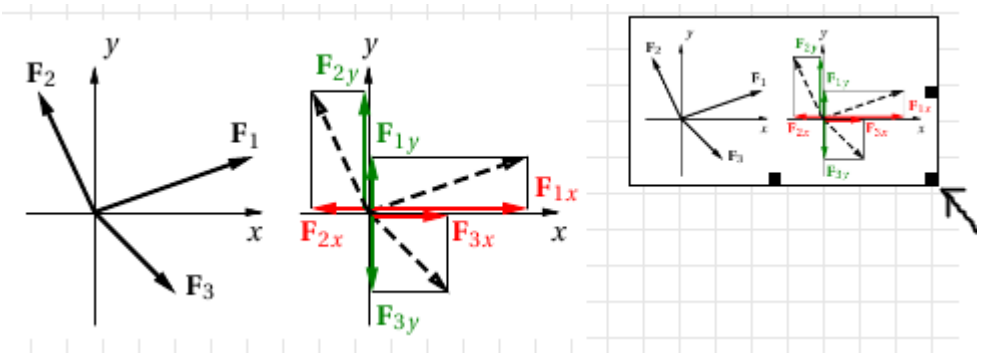
Sie können dann das Bild an den Randmarken vergrößern oder verkleinern.

Im Kontextmenü können Sie

- das Bild auf die tatsächliche Größe zurücksetzen („Reset to original size”),
- den Rand ausschalten („Switch border on/off”)
- den Platzhalter verbergen („Eingabedaten anzeigen”)

So erzeugte Bilder können durch Doppelklick erneut editiert werden.

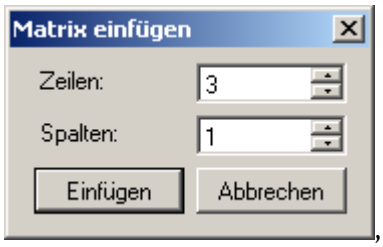
Man kann auch Bilder direkt per Strg - V aus der Zwischenablage einfügen (dabei entsteht ein sogenannter Bildbereich (*picture region*). Solche Bilder sind für die spätere Bearbeitung innerhalb von SMath Studio nicht geeignet. Man kann aber auf diese Weise sehr einfach externe Bilder oder Screenshots einfügen.



An den Randmarken ■ kann man mit der Maus ziehen, um die Größe zu ändern.

2.9. Vektoren und Matrizen

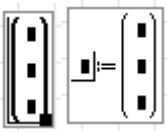
SMath kennt einige Vektor- und Matrixoperationen. Der Befehl [(::)] aus der Matrix-Palette der Seitenleiste öffnet den Dialog



hier schon auf 3 Zeilen und 1 Spalte eingestellt. Alternativ ruft man den Dialog mit Strg - M auf. Einfügen liefert einen passenden Platzhalter

Stellt man die Einfügemarke auf die linke Klammer, kann man vor der Matrix etwas einfügen, z.B. den Zuweisungsoperator (Doppelpunkt)

:



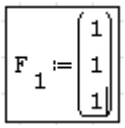
Diese Platzhalter werden nun ausgefüllt. Dabei kann man mit den Pfeiltasten oder mit der Maus zwischen den Platzhaltern navigieren. Der mit dem Punkt . erzeugte Index ist ein Textindex.

```
F.1
```

1

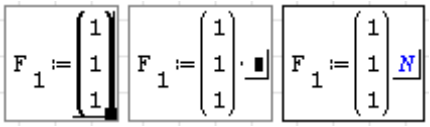
1

1

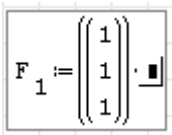


Rechte Klammer anklicken und noch eine Einheit hinzufügen:

```
*N
```



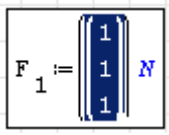
Es kann passieren, dass SMath dabei eine Klammer um die Matrix setzt:



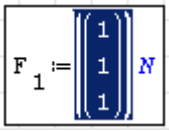
Die schadet nicht, ist aber auch nicht erforderlich, daher soll sie beseitigt werden.

Überflüssige Klammern löschen: Klammern können nur gemeinsam mit ihrem Inhalt gelöscht werden. Daher muss man vorher den Inhalt kopieren.

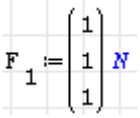
1. Klammerinhalt markieren (nur den Inhalt, nicht die äußere Klammer)



2. Kopiereren mit Strg - C
3. Markierung auf ganze Klammer erweitern, z.B. mit → ←

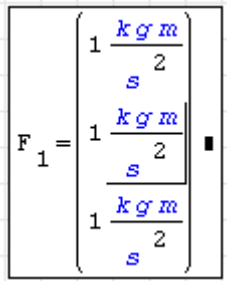


4. Inhalt wieder einfügen mit Strg - V



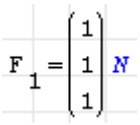
Bei der Anzeige kann eine Maßeinheit für die ganze Matrix gewählt werden:

F.1=



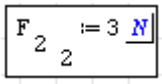
Dafür wählt man den Platzhalter an und tippt

N ⇆



Man kann Vektoren und Matrizen auch erzeugen, indem man einzelnen Elementen Werte zuweist. Elemente werden mit Adress-Indizes angesprochen, die mittels [erzeugt werden

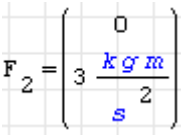
F.2[2 Leert. :3N ⇆



Adress-Indizes (mit [erzeugt) unterscheiden sich in der Formatierung von Textindizes (mit . erzeugt) durch ein vorangestelltes Leerzeichen, also einen größeren Abstand.

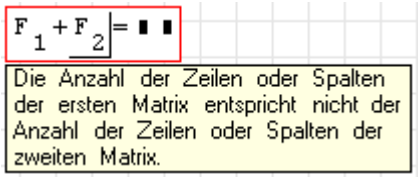
Verwendet man nur einen Index, wird ein Spaltenvektor erzeugt. Die erzeugten Matrizen oder Vektoren sind gerade so groß, wie die höchsten vorkommenden Indizes.

F.2=



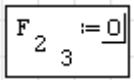
Die Addition von $\underline{F}_1$ und $\underline{F}_2$ scheitert daher:

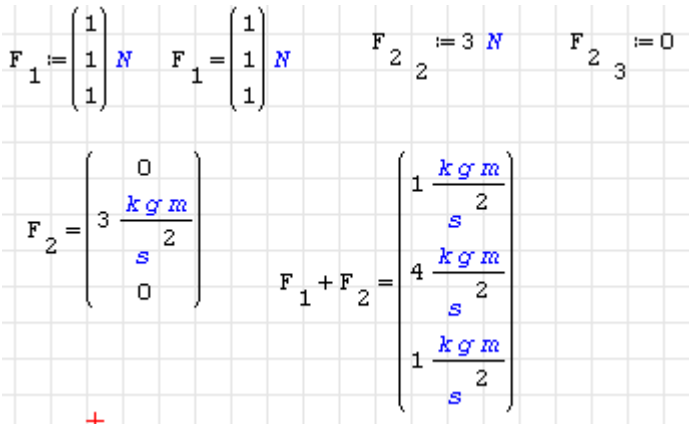
F.1+F.2=



Definiert man noch das dritte Vektorelement, dann geht es:

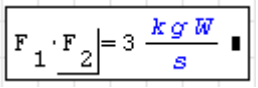
F.2[3 Leert. :0





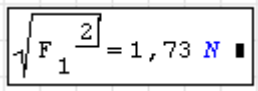
Das Skalarprodukt zweier Vektoren schreibt man als gewöhnliches Produkt:

```
F.1*F.2=
```



Der Betrag eines Vektors ist die Wurzel aus dem Skalarprodukt mit sich selbst:

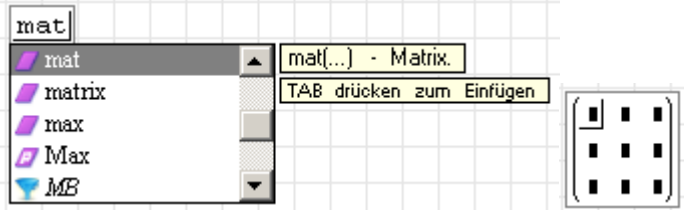
```
\F.1^2=
```



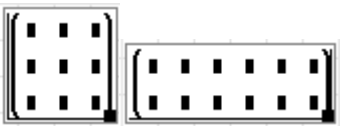
Man kann auch die Größe einer Matrix interaktiv ändern. Das geht manchmal schneller, als den Dialog mit Zeilen und Spalten auszufüllen. Beachten Sie den feinen Unterschied zwischen den Tastatureingaben. Einmal erhalten Sie einen 2x2-Platzhalter, das andere mal einen 3x3-Platzhalter

```
mat
```


```
mat(
```



Klicken Sie nun auf die linke Klammer und ziehen Sie die Matrix am schwarzen Quadrat rechts unten mit der Maus in die richtige Größe



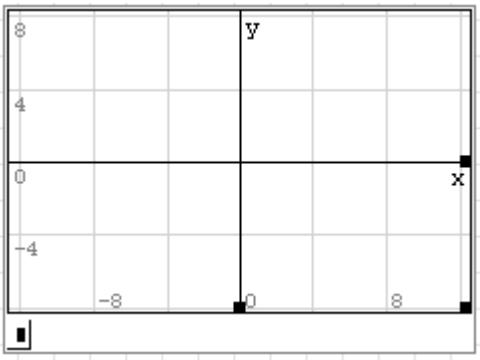
2.10. Diagramme

Wenn es nur mal eben um die schnelle Darstellung einer Funktion geht, dann kann man den eingebauten 2D-Diagrammbereich verwenden. Für bessere Qualität und mehr Möglichkeiten können die Funktionen `Draw2D()` und `Draw3D()` benutzt werden, siehe dazu Abschnitt 5.1. In dieser Einführung behandeln wir nur den internen 2D-Diagrammbereich.

Hinweis: Ohne Mausrad wird man nicht wirklich zufrieden sein, denn dies ist der einfachste Weg zur Einstellung der Diagrammgrenzen.

Mit @ oder über **Hauptmenü> Einfügen> Grafik> Zweidimensional** fügt man einen 2D-Diagrammbereich ein:

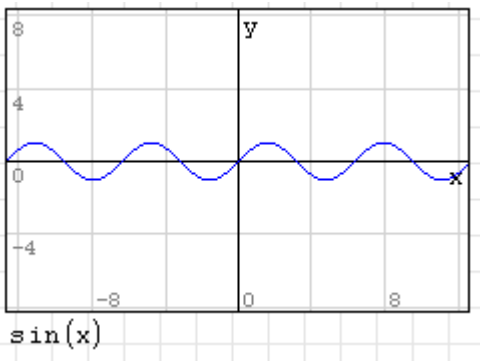
```
@
```



In den Platzhalter unten links schreibt man den zu plottenden Ausdruck. An den anderen Platzhaltern kann man mit der Maus ziehen, um die Diagrammgröße zu ändern.

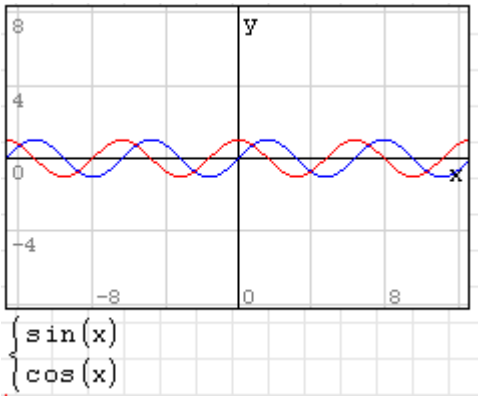
Bei Funktionsplots ist die unabhängige Variable immer x.

```
sin(x↵
```



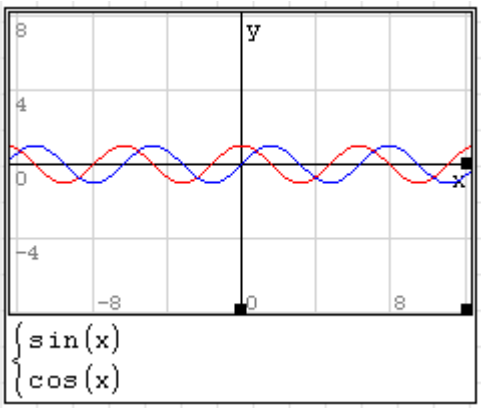
Will man mehrere Funktionen plotten, dann muss man sie in einer Liste („System") gruppieren. Das ist etwas ähnliches wie ein Vektor und hat nur links eine geschweifte Klammer. Man bekommt sie durch Eintippen von `sys` ⇆ oder mit dem Symbol { in der Funktionen-Palette der Seitenleiste. Hier ein Beispiel:

```
@sys⇆ sin(x → → cos(x ↵
```



Bereichseinstellung in Diagrammen:

Das Diagramm muss zunächst angeklickt werden, so dass die Randmarken zu sehen sind.



- Ausschnitt verschieben: Klicken und Ziehen der Diagrammfläche
- Beide Achsen zoomen: Mausrad
- nur x-Achse zoomen: ⇑ -Mausrad
- nur y-Achse zoomen: Strg -Mausrad

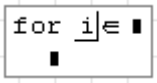
Weitere Funktionen

Neben Funktionen lassen sich auch noch Linienzüge (Polygone) und Text im Diagramm darstellen. Grafikobjekte können auch animiert werden. Darauf wird im Abschnitt 5.2 eingegangen.

2.11. Etwas Programmierung

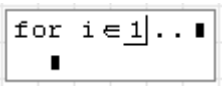
Für die Datenverarbeitung und deren grafische Aufbereitung bieten sich häufig Schleifen an. Man erzeugt sie mit „for” aus dem Panel „Programmierung” oder mit der Tastatur:

`for` ⇆ `i`

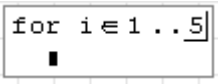


Es entstehen drei Platzhalter, den ersten haben wir bereits mit dem Schleifenzähler i ausgefüllt. Der Zähler soll nacheinander Werte von 1 bis 5 annehmen. Dafür verwenden wir die Funktion `range()` auf dem zweiten Platzhalter.

→ `rang` ⇆ `1`

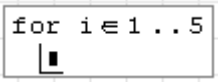


→ `5`



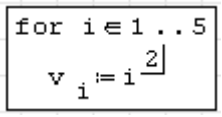
Der dritte Platzhalter ist der Schleifenkörper.

→ →

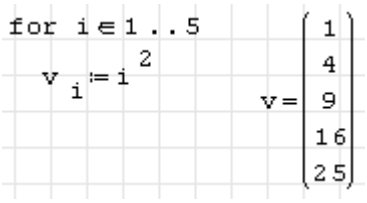


Dort wollen wir nun einen Vektor erzeugen, dessen i-tes Element die Zahl i^2 enthält. Der Adress-Index `i` bei `v` wird mit der eckigen Klammer auf erzeugt und unterscheidet sich von beschreibenden Indizes durch den zusätzlichen Zwischenraum:

`v[i` Leert. `:i^2`



Das Ergebnis entspricht den Erwartungen.

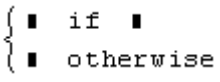
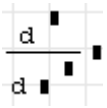
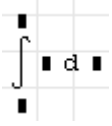


Nach Durcharbeiten der „Ersten Schritte" sollten Sie in der Lage sein, das Rechenblatt zur Interpolation in Abschnitt 10.3 nachzuvollziehen.

2.12. Überlebensregeln, Tips und Tricks

- In Bezeichnern wird Groß- und Kleinschreibung unterschieden!
- Benutzen Sie `i` nur dann als Schleifenzähler (auch in Summen- oder Produktoperatoren), wenn Sie nicht mit komplexen Zahlen rechnen wollen. Dadurch ist `i` nicht mehr als imaginäre Einheit verwendbar.

- Schreiben Sie logische Operationen (=, ≠, ∧, ∨, ⊕) mit ihren Operanden immer in Klammern, also $(a \wedge b)$. Diese Operatoren haben unerwartet niedrige Priorität, insbesondere niedriger als der Zuweisungsoperator `:=`.
- Beachten Sie den Unterschied zwischen beschreibenden Indizes (`F.x`) und Adress-Indizes (`v[i]`)
- Definieren Sie stückweise stetige Funktionen durch Multiplikation mit logischen Ausdrücken oder mit der `cases()` Funktion, nicht jedoch mit der `if()`-Anweisung.
- *Maßeinheiten* mit der Tastatur:
  - Eingabe eines Hochkommas und des Namens
  - Eingabe des Namens und Bestätigung mit ⇆ in der Vorschlagsliste
- Unnötige Klammern entfernen:
  1. Inneres markieren, kopieren,
  2. Klammer mit Inhalt markieren, einfügen (durch Zwischenablage ersetzen)
- Interaktives Handbuch nutzen
  - Aufruf über Werkzeugleiste 
  - Verweisen mit Strg-Klick folgen
- Wichtige Operatoren und ihre Tastatureingabe:

Operator	Bedeutung	Eingabe
	Substitution	`at(;`
	Fallunterscheidung	`cases(;;`
	Elementadressierung in Feldern	`[`
	mit zwei Indizes	`[;`
	Differentiation	`diff(;`
	Differentiation	`diff(;;`
	Integration	`int(`
	Summe	`sum(`
		`sum(;;;`

▮..▮	Wertebereich (Vektor)	`range(;`
▮;▮..▮		`range(;;`

3. Programmbedienung

Dieses Kapitel beschreibt die Benutzeroberfläche mit Menüstruktur, die verschiedenen Objekte auf dem Rechenblatt und deren Editiermöglichkeiten sowie die nutzerspezifischen Einstellungen im Programm.

3.1. Benutzeroberfläche

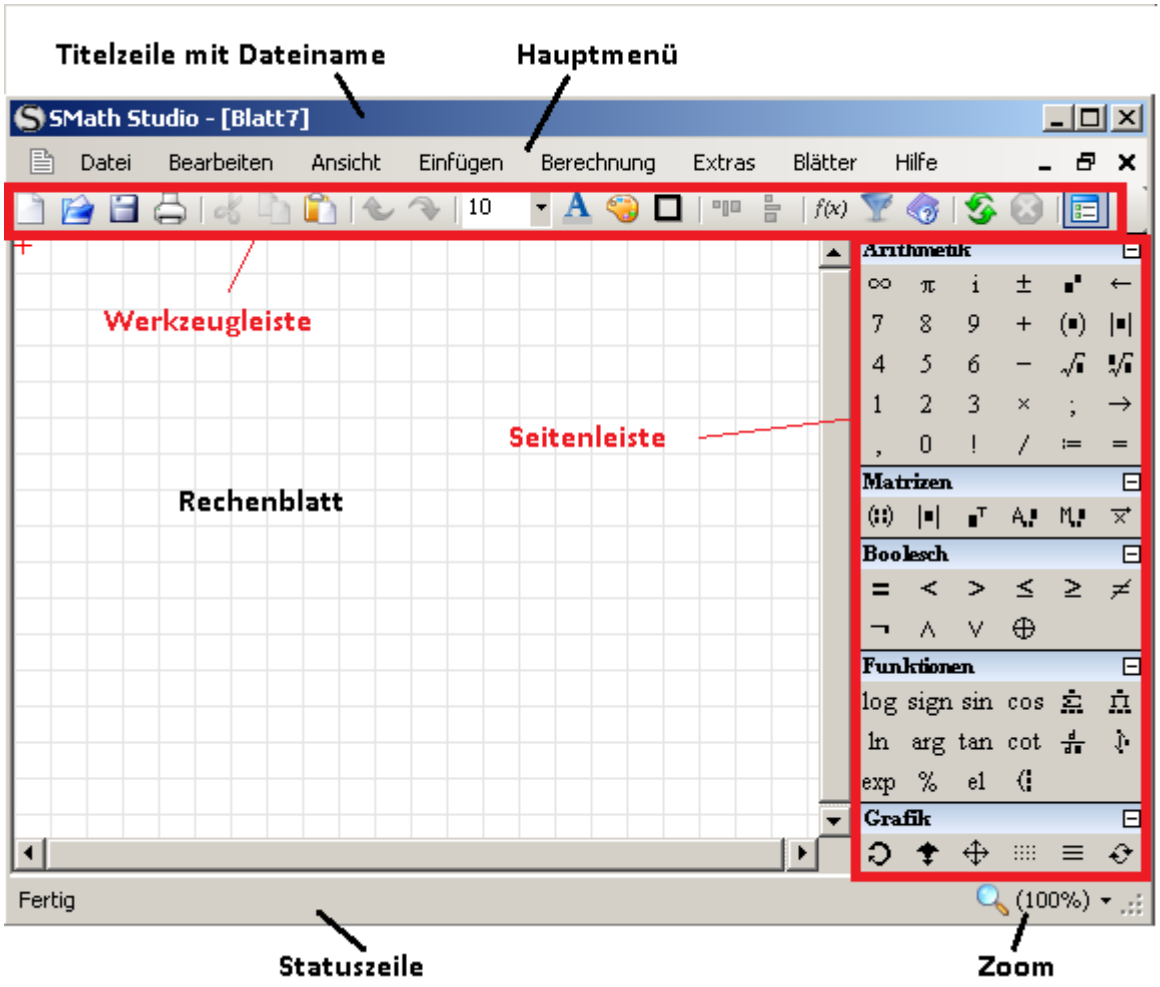


Abbildung 3.1.: Elemente der Benutzeroberfläche

Die Benutzeroberfläche besteht aus

- Titelzeile mit Namen des aktuellen Rechenblatts
- Hauptmenü (nicht konfigurierbar)
- Werkzeugleiste (nicht konfigurierbar)

- Seitenleiste mit Symbol- und Funktionspaletten. Diese kann über den Knopf  ein- und ausgeschaltet werden. Die einzelnen Paletten könnnen mit dem Knopf  eingeklappt werden.
- Statuszeile mit Zoomsteuerung.
- Rechenblatt, eventuell mehrere davon, je nach Fenstereinstellung. Rechenblätter werden im Abschnitt 3.2 detailliert behandelt.

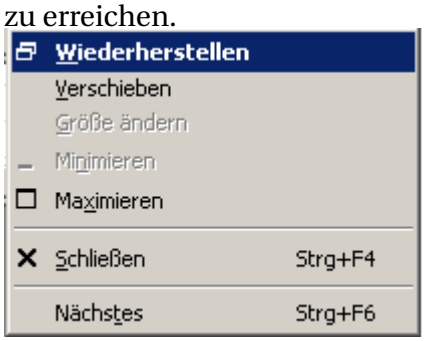
3.1.1. Titelzeile

Die Titelzeile zeigt den Namen des aktiven Rechenblatts an, wenn dieses maximiert ist. Wenn er mit einem Stern versehen ist, dann ist der aktuelle Stand nicht gespeichert. Mitunter wird der Programmname „SMath Studio" von Version zu Version leicht modifiziert.

3.1.2. Hauptmenü

Das Hauptmenü befindet sich unterhalb der Titelzeile und hat die folgenden Einträge:

Blatt-Menü  Funktionen des Fenstermanagers für das aktuelle Rechenblatt, falls dieses maximiert ist. Ansonsten sind diese Funktionen durch die Titelzeile der Einzelblätter zu erreichen.



Wiederherstellen Alle geöffneten Blätter in separaten Fenstern anzeigen.

Minimieren Das aktuell angezeigte Blatt minimieren

Maximieren Minimiertes Blatt wieder aufklappen

Schließen Strg - F4 Blatt schließen

Nächstes Strg - F6 Umschalten auf das nächste offene Blatt

Datei

Blatt erstellen Strg - N Neues Blatt öffnen

Öffnen Strg - O Vorhandenes Blatt öffnen. Alternativ kann man auch eine SMath Datei (.sm) auf das SMath-Fenster ziehen.

Blatt schließen Strg-F4 aktuelles Blatt schließen

Speichern Strg-S aktuelles Blatt speichern. Damit wird die Befehlshistorie gelöscht, Rückgängig machen geht nur bis zum letzten Speicherpunkt.

Speichern unter Aktuelles Blatt unter wählbarem Namen und in wählbaren Formaten speichern. Hier verbergen sich also auch diverse Export-Funktionen.

Starte Sitzung Gemeinsame Bearbeitung eines Arbeitsblatts starten.

An Sitzung teilnehmen In bestehende Sitzung einschalten.

Blatteinstellungen Einstellungen zum Druckbild (siehe Abschnitt 3.15.1)

Drucken Strg-P

Druckvorschau Vorschau des Druckbildes.

Eigenschaften Einstellungen zu den Dateieigenschaften (Metadaten, siehe Abschnitt 3.15.2)

Zuletzt geöffnete Dateien Liste der zuletzt geöffneten Dateien.

Schließen Alt-F4 Alle offenen Blätter und das Programm schließen. Gegebenenfalls wird Speichern angeboten.

Bearbeiten Dieses Menü enthält die Standardfunktionen zum Blatt-Bearbeiten. Ist ein Bereich komplett (hellblau) markiert, dann erscheinen die Optionen des Kontextmenüs auch hier im Hauptmenü. Auf diese Optionen wird bei der Beschreibung der Bereiche eingegangen.

Rückgängig Strg-Z Macht vorangegangene Aktionen rückgängig. Mitunter gilt das nur für die Modifikationen am aktuell geänderten Bereich (Formel). Maximal möglich bis zum letzten Speicherpunkt.

Wiederherstellen Strg-Y Rückgängig rückgängig machen.

Ausschneiden Strg-X

Kopieren Strg-C

Einfügen Strg-V

Löschen Entf

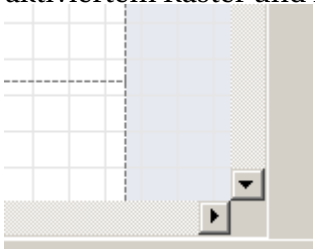
Alles markieren Strg-A

Aktualisieren

Ansicht In diesem Menü können Anzeigeelemente ein- und ausgeschaltet werden.

Raster Schalter für das Kästchenraster auf dem Blatthintergrund

Druckbereich Schalter für die Anzeige des rechten Randes und des Seitenumbruchs auf dem Blatthintergrund. Die Voreinstellung kann unter **Hauptmenü> Einstellungen>** Register „Interface": Druckbereich geändert werden. Hier ein Bild mit aktiviertem Raster und Druckbereich



Ausgabefenster Schalter für das Ausgabefenster der Debug-Funktion `trace()`

Dynamische Auswahl Schalter für die Anzeige der dynamischen Auswahl. Das ist eine dynamisch erzeugte Liste aller bekannten Bezeichner, die zum gerade eingetippten Wort passen. Die Abschaltung kann bei Demonstrationen am Beamer hilfreich sein, wenn man Dinge mit der Maus zeigen will, ohne dass die dynamische Auswahl aufpoppt.

Einfügen Hier können SMath-Objekte in das Rechenblatt eingefügt werden.

Matrix Strg-M Aufruf des Dialogs „Matrix einfügen". Das Arbeiten mit Matrizen wird in Abschnitt 4.9 erläutert.

Funktion Strg-E Aufruf des Dialogs „Einfügen - Funktion"

Operator Aufruf des Dialogs „Einfügen - Operator" zum Einfügen arithmetischer und boolscher Operatoren

Einheit Strg-W Aufruf des Dialogs „Einheit einfügen"

Hintergrund Aufruf des Dialogs „Hintergrund". Damit kann man ein Hintergrundbild laden, z.B. für Formblätter.

Bereich Fügt zwei Begrenzungslinien ein, zwischen denen das Rechenblatt zusammengeklappt (kollabiert) werden kann.

Trennlinie Strg-↵ Fügt eine horizontale Trennlinie ein. Diese erzeugt im Ausdruck einen Seitenumbruch und in exportierten Anwendungen einen neuen Tab.

Die folgenden Bereiche werden durch Plugins bereitgestellt, je nach Installation können diese Bereiche auch fehlen.

Bild Bild erstellen oder aus Datei einfügen. Dient Illustrationszwecken.

Bild erstellen Strg-T Fügt einen Bereich ein, in dem mit der Maus freihand gezeichnet werden kann. Löschen/Radieren ist nicht möglich, daher hat diese Funktion wenig Wert. Verwenden Sie statt dessen den Image-Bereich.

aus Datei Rasterbilder in ve rschiedenen Formaten (PNG, JPG, Icons,...) können aus der Zwischenablage oder durch Ziehen mit der Maus eingefügt werden. Der Inhalt ist statisch, wird also nicht aktualisiert, wenn der Dateiinhalt sich ändert.

Checkbox (Plugin *Checkbox Region*) Schaltfläche, die eine Variable durch Mausklick zwischen Null und Eins umschaltet.

CheckBoxList (Plugin *CheckboxList Region*) Anordnung mehrerer An/Aus-Schaltflächen

Combobox (Plugin *Combobox Region*) Auswahlliste, der Wert einer Variable kann aus einer Liste von Werten ausgewählt werden.

ComboBoxList (Plugin *ComboBoxList Region*) Auswahlliste, der Wert einer Variable kann aus einer Liste von Werten ausgewählt werden.

Development Tools Werkzeuge für Entwickler

Math⇒Terms Übersetzt SMath-Ausdrücke in C#-Code für Plugins

Math's visual debug Übersetzt SMath-Ausdrücke in einen Strukturbaum

Network analyzer Zeigt Verbindungsdetails für Internet-Adressen an

System information Zeigt SMath-relevante Systeminformationen an.

Diagramm Diagramm erstellen

Zweidimensional (Tastatur @) Fügt einen 2D-Diagrammbereich ein.

Dreidimensional Fügt einen 3D-Diagrammbereich ein.

Formatted label (Plugin *CFLabel Region*) Bedingt formatierter Text. Text, Textfarbe und Hintergrundfarbe können abhängig von Berechnungsergebnissen gesetzt werden.

Hyperlink Strg-H (Plugin *Hyperlink Region*) Fügt einen Hyperlink-Bereich ein (siehe Seite 54)

Image (Plugin *ImageRegion*) Fügt einen Rasterbildbereich zur Anzeige von Bilddateien oder Datenmatrizen ein. Alternativ kann das Bild mit dem Standardprogramm für png-Dateien editiert werden.

Modeller (Plugin *Modeller Region*) Bereich für interaktive parametrische Grafik

Maxima (Plugin *Maxima*) Steuerung der Maxima-Schnittstelle

Log Fenster zur Sichtung des Sitzungs- und Übersetzungsprotokolls

Debug Fenster zur interaktiven Sichtung und Steuerung des Übersetzungsprozesses

Settings Fenster für Einstellungen der Schnittstelle.

Numeric up-down (Plugin *Numeric UpDown Region*) Komfortables Eingabefeld für Zahlenwerte, die Werte können mit der Maus oder den Pfeiltasten geändert werden.

RadioButtonList (Plugin *RadioButtonList Region*) Anordnung mehrerer An/Aus-Schaltflächen

Slider Schieberegler zur interaktiven Einstellung von Werten

Snapshot Strg-F Bereich für den automatisierten Export von Ausschnitten des Rechenblatts als Bilddatei (beispielsweise für die Erstellung von Bildern für Handbücher)

Table (Plugin *Table Region*) Tabelle zur Anzeige von Daten

Textbereich (Tastatur ") Startet einen Textbereich.

X-Y Plot Erweiterter 2D-Diagrammbereich, siehe Abschnitt F.3)

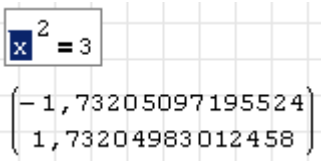
Bereichs-Sperre Damit kann ein kollabiertes Gebiet mit einem Passwort gegen Aufklappen geschützt werden.

Sperren Ruft den Dialog „Erstelle Passwort” auf.

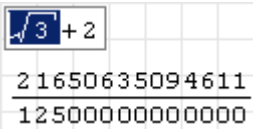
Entsperren Fordert zur Passworteingabe zum dauerhaften Entsperren des Gebiets auf.

Berechnung

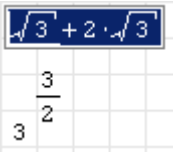
Lösen Löst eine Gleichung numerisch nach einer Variable auf. Die Variable muss vorher markiert werden.



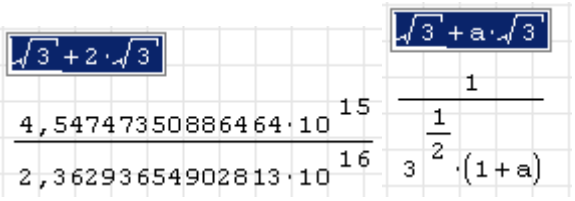
Berechnen Wertet einen markierten (Teil-)Ausdruck aus.



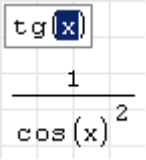
Vereinfachen Wertet einen markierten (Teil-)Ausdruck symbolisch aus.



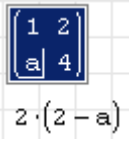
Kehrwert Berechnet den Kehrwert des markierten (Teil-)Ausdrucks symbolisch



Differenzieren Leitet den Ausdruck nach der markierten Variable ab



Determinante Berechnet die Determinante einer markierten Matrix



Automatische Berechnung Schalter für die automatische Berechnung nach jeder Änderung. Abschalten kann sinnvoll sein, wenn die Rechenzeit zu lang ist und mehrere Änderungen gemacht werden müssen, bis eine neue Rechnung überhaupt sinnvoll ist.

Blatt berechnen Strg - F9 Manueller Start der Neuberechnung, wenn die automatische Berechnung abgeschaltet ist.

Prozess unterbrechen Rechnung anhalten (wenn sie zu lange dauert).

Extras

Plugins Öffnet die Seite „Plugins" der Erweiterungs-Verwaltung, dort können Plugins (Anwendungserweiterungen) aus dem Internet geladen oder aktualisiert werden.

Code-Baustein-Verwaltung Öffnet die Seite „Code-Bausteine" der Erweiterungs-Verwaltung, dort können Code-Bausteine aus dem Internet geladen oder vorhandene Bausteine ins aktuelle Rechenblatt eingefügt werden.

Einstellungen Öffnet den Dialog „Einstellungen” für Berechnung und Benutzeroberfläche, siehe Abschnitt 3.15.3

Blätter Bietet Zugriff auf die geöffneten Blätter und darüber hinaus folgende Funktionen:

Blatt erstellen Strg-N Neues Rechenblatt erstellen.

Blatt schließen Strg-F4 Aktuelles Blatt schließen.

Hilfe

Referenzmaterial Eine kleine mathematische Formelsammlung.

Beispiele Öffnet die Seite „Beispiele” der Erweiterungs-Verwaltung. Darin kann man Beispiele aus dem Internet laden oder installierte Beispiele auswählen und als Rechenblatt öffnen.

Nach Aktualisierungen suchen Öffnet die Seite „SMath Studio” der Erweiterungsverwaltung und bietet eventuell vorhandene neuere Programmversionen zur Installation an.

Spenden Sie erhalten die Kontodaten des Entwicklers Andrey Ivashov. Unterstützt werden PayPal, Webmoney und Jandex-Dengi.

SMath Studio Live Öffnet eine Sitzung in der Web-Version von SMath. Diese hat gegenüber der Programmversion einen eingeschränkten Funktionsumfang.

SMath Info Versionsinformation wird angezeigt.

3.1.3. Werkzeugleiste


Neues Blatt, Öffnen, Speichern, Drucken


Ausschneiden, Kopieren, Einfügen


Rückgängig, Wiederherstellen

Schriftgröße, Schriftfarbe, Hintergrundfarbe, Rahmen


Vertikal ausrichten, horizontal ausrichten

Funktion einfügen, Einheit einfügen, Formelsammlung

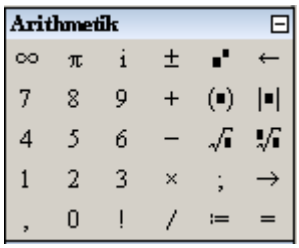

Neu berechnen, Berechnung stoppen


Seitenleiste aus/an

3.1.4. Seitenleiste

Die Seitenleiste bietet den Schnellzugriff auf Symbole, Operatoren und Funktionen, ohne dass man deren Tastaturkürzel kennen muss. Die Seitenleiste kann über die Taste  der Werkzeugleiste an- und ausgeschaltet werden.

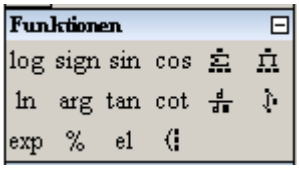
Das **Arithmetik-Panel** enthält Symbole und Funktionen für die Grundrechenoperationen und einige weitere arithmetische Operationen.



Das **Matrizen-Panel** enthält Funktionen für das Rechnen mit Matrizen.

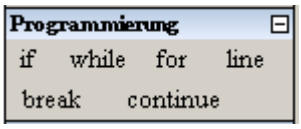
Das **Boolesch-Panel** enthält logische (boolsche) Operationen.

Das **Funktionen-Panel** enthält diverse Funktionen.

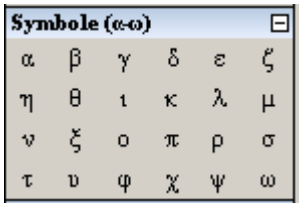


Das **Grafik-Panel** enthält Funktionen zur Steuerung der 2D- und 3D-Grafikanzeige.

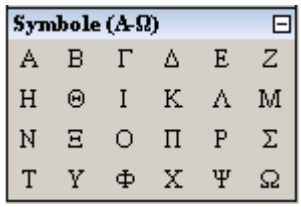
Das **Programmierung-Panel** enthält Funktionen für die Ablaufsteuerung in Funktionsdefinitionen (Programmen)



Das **Panel „Symbole** $\alpha - \omega$**"** enthält die griechischen Kleinbuchstaben.

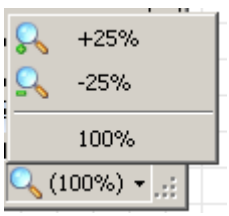


Das **Panel „Symbole** $A - \Omega$**"** enthält die griechischen Großbuchstaben.



3.1.5. Statuszeile

Die Statuszeile zeigt die aktuelle Rechenzeit an und bietet rechts Zugriff auf die Zoom-Einstellung des Rechenblatts.



3.2. Das Rechenblatt

Jede SMath-Datei enthält ein Rechenblatt, also ein zweidimensionales Gebiet, auf dem Bereiche unter- und nebeneinander frei angeordnet werden können. Es gibt standardmäßig folgende Bereichstypen:

- Formel (Abschnitt 3.4)
- Text (Abschnitt 3.5)
- 2D-Diagramm (Abschnitt 5.2)
- 3D-Diagramm (Abschnitt F.2)
- Bild (Rasterbild, statisch)
- Blattbereich (zur Gruppierung anderer Bereiche, einklappbar, Abschnitt 3.13)
- Trennlinien (Erzeugen einen Seitenumbruch, Abschnitt 3.13)

Darüber hinaus können externe Plugins weitere Bereichstypen beisteuern. Beispiele sind:

- Image (dynamisches Rasterbild aus Datei oder Datenmatrix oder mit dem Windows-Standardprogramm für png-Dateien editierbares Rasterbild Plugin *ImageRegion*, Abschnitt 3.7)
- Steuerelemente wie Schieberegler, Schaltknöpfe und Auswahllisten

Bereiche bearbeiten Bereiche können auf dem Rechenblatt beliebig, auch übereinander, angeordnet werden.

Bevor man sie verschieben, löschen oder kopieren kann, müssen sie markiert werden. Das geschieht durch Ziehen mit der Maus, dabei werden die Bereiche hellblau eingefärbt.

Ausnahme: Bei gedrückter Strg -Taste kann man mit der linken Maustaste einen noch nicht markierten Bereich kopieren und anderswo auf das Rechenblatt ziehen. Die Kopie ist anschließend hellblau markiert.

Die Auswahl kann bei gehaltener Umschalttaste ⇑ erweitert oder reduziert werden.

Anschließend kann man die markierten Bereiche

- Verschieben (durch Ziehen mit der Maus oder mit den Pfeiltasten)
- Löschen (mit „Löschen" im Kontextmenü oder mit der Taste Entf)
- Ausschneiden, also Löschen und in die Zwischenablage übernehmen (mit „Ausschneiden" im Kontextmenü oder Strg - X)
- Kopieren (mit „Kopieren" im Kontextmenü oder mit Strg - C)
- Duplizieren (durch Ziehen bei gedrückter Strg -Taste)

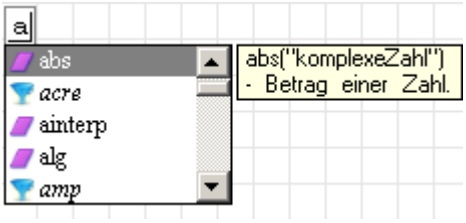
Ausgeschnittene oder kopierte Bereiche können an der Einfügemarke ins Rechenblatt eingefügt werden (Strg - V oder mit „Einfügen" aus dem Kontextmenü).

Die Befehle „Ausschneiden", „Löschen", „Einfügen" und „Kopieren" finden sich auch unter **Hauptmenü> Bearbeiten**.

3.3. Dynamische Auswahl

Eine Besonderheit von SMath Studio ist die Dynamische Auswahl. Wann immer Sie im Rechenblatt eine Formel eintippen, klappt ein dynamisches Fenster auf, welches Ihnen mögliche Ergänzungen und deren Beschreibung oder Definition anzeigt. Sie können in der Liste mit den Pfeiltasten auswählen und die Auswahl mit ⇆ bestätigen.

Die dynamische Auswahl kennt nicht nur die eingebauten Funktionen, Maßeinheiten und Konstanten, sondern auch die von Ihnen selbst definierten, sowie alle Funktionen und Konstanten aus eventuell installierten Anwendungserweiterungen (Plugins).



Die Symbole in der Liste haben folgende Bedeutung:

- Eingebaute Konstante oder Anweisung (Eulersche Zahl e, imaginäre Einheit i, aber auch Programmanweisungen wie `break` und `continue`)
- selbiges, aber vom Benutzer überschrieben (tun Sie das nur, wenn Sie wissen, was Sie tun)
- Eingebaute Funktion
- Variable oder Funktion aus einem Plugin (extern oder intern)
- Benutzerdefinierte Variable oder Funktion
- Eingebaute Maßeinheit oder Konstante
- Benutzerdefinierte Maßeinheit oder Konstante
- Code-Baustein

Bei selbstdefinierten Bezeichnern können Sie eine Beschreibung angeben, die in der dynamischen Auswahl erscheint, siehe Abschnitt 3.12.

3.4. Formelbereich

Formelbereiche entstehen an der Cursorposition durch Tastatureingabe von Bezeichnern (Variablennamen oder Funktionsnamen) oder Operatoren mit der Tastatur oder aus Menüs. Der Inhalt von Formelbereichen wird mathematisch formatiert und mathematisch ausgewertet.

Formelbereiche können als Ganzes einheitlich formatiert werden (der Bereich muss dazu als Ganzes hellblau markiert sein):

- Schriftgröße (Werkzeugleiste > Auswahlbox)
- Rahmen an/aus (Werkzeugleiste > „Die Grenze des Elements")
- Hintergrundfarbe (Werkzeugleiste > „Hintergrundfarbe")
- Textfarbe (Werkzeugleiste > „Textfarbe")

3.4.1. Markieren in Formelbereichen

Die Wirkung mancher Aktionen hängt davon ab, welche Ausdrücke in der Formel markiert sind. Die Markierung wird mit der Maus vor oder nach einem Bezeichner gesetzt und kann mit der Leertaste erweitert werden. Die Markierung besteht aus den Teilen

- Unterstreichung: Sie kennzeichnet den Wirkungsbereich der Markierung
- Senkrechte Einfügemarke: Sie kennzeichnet den Ort der nächsten Eingabe

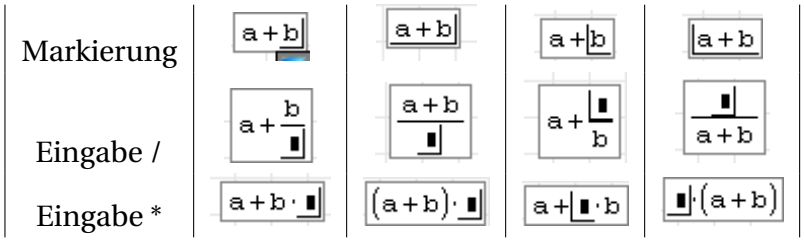


Abbildung 3.2.: Wirkung von Eingaben in Formeln abhängig von der Markierung

3.4.2. Größe von Matrizen, Anweisungsblöcken und Listen ändern

Symbol anklicken, erscheinenden Marker rechts unten ziehen. Dadurch entstehen neue leere Platzhalter unten oder rechts. Verkleinern geht auch, aber nur durch Abschneiden unten oder rechts. Streichen von einzelnen Zeilen oder Spalten ist nicht durch Editieren, sondern nur durch entsprechende Indexoperationen möglich.

In einspaltigen Objekten (Spaltenmatrix, Anweisungsblock, Listen) kann man Elemente mit Backspace löschen und neue hinzufügen, indem man das Argumenttrennzeichen (Semikolon in unserer Standardeinstellung) an der entsprechenden Stelle eintippt.

3.4.3. Griechische Buchstaben

Griechische Buchstaben können in Formel- und Textbereichen eingegeben werden durch

- Buchstabe tippen, dann Strg-G . Die Zuordnung zeigt Tabelle 3.2.
- Auswahl aus den Seitenpanels „Symbole (A-Ω)" oder „Symbole (α-ω)"

Name	groß		klein		Name	groß		klein	
Alpha	A	A	a	α	Ny	N	N	n	ν
Beta	B	B	b	β	Xi	X	Ξ	x	ξ
Gamma	G	Γ	g	γ	Omikron	O	O	o	o
Delta	D	Δ	d	δ	Pi	P	Π	p	π
Epsilon	E	E	e	ε	Rho	R	P	r	ρ
Zeta	Z	Z	z	ζ	Sigma	S	Σ	s	σ
Eta	H	H	h	η	Tau	T	T	t	τ
Theta	Q	Θ	q	ϑ	Ypsilon	U	Y	u	υ
			J	ϑ	Phi	F	Φ	f	φ
Iota	I	I	i	ι				j	ϕ
Kappa	K	K	k	κ	Chi	C	X	c	χ
Lambda	L	Λ	l	λ	Psi	Y	Ψ	y	ψ
My	M	M	m	μ	Omega	W	Ω	w	ω

Tabelle 3.2.: Griechische Buchstaben und die zugehörigen lateinischen Buchstaben, aus denen sie mit Strg-G entstehenden. Rot: Lateinische und griechische Version nicht unterscheidbar.

3.4.4. Maßeinheiten

Maßeinheiten und andere vordefinierte Konstanten werden in Formelbereichen kursiv blau dargestellt. Sie werden durch Variablen gleichen Namens nicht überschrieben. Intern werden sie durch ein vorangestelltes Hochkomma (') markiert. Maßeinheiten können eingegeben werden durch

- Symbol tippen, dynamische Auswahl korrigieren (↑ oder ↓) und mit ⇆ bestätigen. Wird die Auswahl nicht mit ⇆ bestätigt, ist der eingetippte Bezeichner eine normale Variable. So kann man die Variable m definieren, ohne die Einheit m unbrauchbar zu machen.
- Hochkomma tippen, dann Einheitsname. Die dynamische Auswahl erscheint, muss aber nicht bestätigt werden.
- **Hauptmenü > Einfügen > Einheit** > Dialog „Einheit einfügen"
- Strg-W ruft den Dialog „Einheit einfügen" ebenfalls auf.

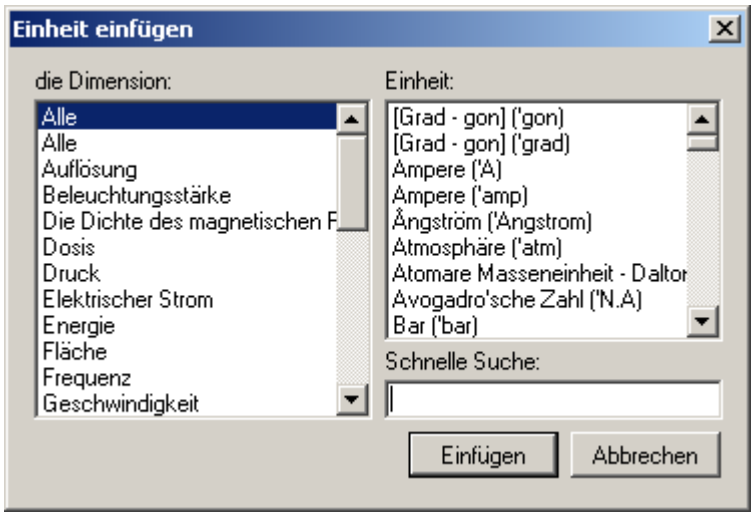
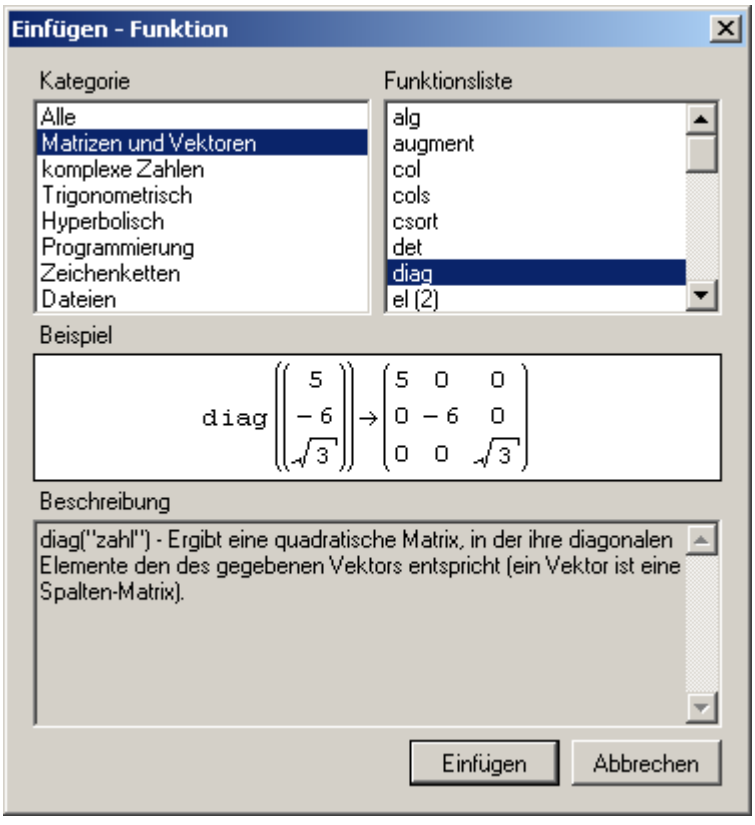


Abbildung 3.3.: Dialog „Einheit einfügen”

3.4.5. Funktionen



Es gibt vordefinierte Funktionen und benutzerdefinierte Funktionen. Der Funktionsaufruf besteht aus dem Funktionsnamen, gefolgt von den durch Kommas getrennten Argumenten in einer Klammer.

Funktionen können mit dem Dialog „Einfügen - Funktion” ausgewählt werden. Dieser Dialog wird aufgerufen durch

- **Hauptmenü> Einfügen> Funktion**
- Werkzeugleiste> 
- Strg - E

Winkelfunktionen erwarten ihr Argument im Bogenmaß. Mit der Maßeinheit ° kann man sich aber behelfen.

$$\sin\left(\frac{\pi}{2}\right)=1$$

$$\sin(90\,°)=1$$

Beim Eintippen des Namens einer Funktion in einem Formelbereich bietet die dynamische Auswahl eine Liste passender bekannter Funktionsnamen an.

3.4.6. Internes Format

Der Inhalt jedes Formelbereichs wird intern als eine Textzeile dargestellt (eindimensionale oder serialisierte Form). Diese Darstellung wird in der Zwischenablage und beim Import oder Export symbolischer Ausdrücke verwendet.

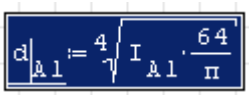
In den .sm-Dateien wird das Format nicht verwendet, dort kommt eine XML-Darstellung in umgekehrter polnischer Notation zur Anwendung (UPN).

Das interne Format entspricht weitgehend der Zeichenfolge, mit der Ausdrücke von der Tastatur eingegeben werden.

Operatoren werden durch die zugehörigen internen Funktionen dargestellt (siehe auch Tabelle **??**)

Man kann das interne Format eines Formelbereiches oder eines Teils davon ermitteln, indem man ihn markiert und in die Zwischenablage kopiert. Fügt man den Inhalt in einen existierenden Textbereich ein, erhält man die interne Darstellung.

Ein Beispiel: Ein markierter Ausdruck



wird mit Strg - C in die Zwischenablage kopiert. Er ist dort als

```
d.Al:nthroot(I.Al*64/π,4)
```

gespeichert.

3.5. Textbereich

Ein Textbereich enthält Text, der nicht mathematisch formatiert und nicht mathematisch ausgewertet wird.

Textbereiche entstehen, wenn Sie

- " (Anführungszeichen) am Cursor im Rechenblatt eintippen
- Text eintippen, der Leerzeichen enthält (bevor Rechenoperatoren auftreten, sonst wird der Bereich endgültig als Formel betrachtet)
- **Hauptmenü> Einfügen> Textbereich** aufrufen

Wenn Sie Objekte aus Mathematik-Bereichen einfügen, ist die Anzeige unterschiedlich. Manche Operatoren erscheinen formatiert (z.B. die Relationszeichen), andere erscheinen in ihrer Eingabetextversion oder als unbrauchbares Ersatzsymbol. Griechische Buchstaben erscheinen korrekt und können wie in Mathe-Bereichen durch nachfolgendes Strg - G erzeugt werden.

Textbereiche können als Ganzes einheitlich formatiert werden:

- Schriftgröße (Werkzeugleiste > Auswahlbox)
- Schriftstil, z.B. **fett** (Strg - B), *kursiv* (Strg - I) oder unterstrichen (Strg - U)
- Rahmen an/aus (Werkzeugleiste> „Element umrahmen ein/aus")
- Hintergrundfarbe (Werkzeugleiste> „Hintergrundfarbe")
- Textfarbe (Werkzeugleiste> „Textfarbe")

3.5.1. Zeilenumbruch

Es gibt keinen automatischen Zeilenumbruch. Eine neue Zeile wird angefangen mit

- Strg - ↵ oder ⇑ - ↵ , neue Zeile im gleichen Textbereich, oder
- ⇑ , neue Zeile in neuem Textblock, linksbündig ausgerichtet.

3.5.2. Schriftart in Textbereichen ändern

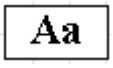
SMath Studio bringt von Haus aus gegenwärtig (März 2014) keine Möglichkeit mit, die Schriftart von Textbereichen zu ändern. Das Plugin *Text Region Utilities* bietet aber Alternativen an. Es erzeugt im Menü „Einfügen" neue Einträge:

Add a new text choosing font Neuen Textbereich mit wählbarer Schriftart einfügen. Angeboten werden alle im System vorhandenen Schriftarten. Zum Gitterraster passen optimal die 11-Punkte Schriften:

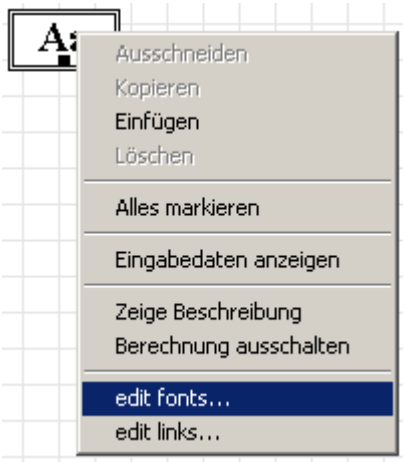
Es wäre logisch und einfach, in der Werkzeugleiste oder im Kontextmenü von Textbereichen einen Eintrag zur Auswahl der Schriftart zu haben. Leider ist dies im Basisprogramm noch nicht realisiert und es gibt auch keine Schnittstelle, dies über ein Plugin zu ergänzen. Das hat zunächst die Folge, das man die Schriftart bestehender Bereiche nicht ändern kann.

Im Plugin *Text Region Utilities* wird daher ein neuer Bereichstyp (Toolbox) angeboten, der nach Neuberechnung des Dokuments alle Texte kennt und deren Änderung über sein Kontextmenü anbietet. Er wird folgendermaßen angewendet:

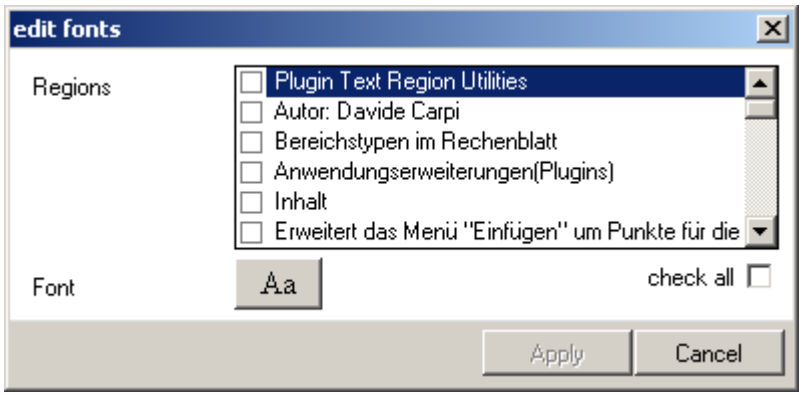
1. Einfügen des Toolbox-Bereichs mit **Einfügen> Textbereich (Utilities)> insert a toolbox**

2. Neuberechnen des Dokuments mit 
3. Kontextmenü der Toolbox öffnen (rechte Maustaste) und den Eintrag „edit fonts" auswählen:

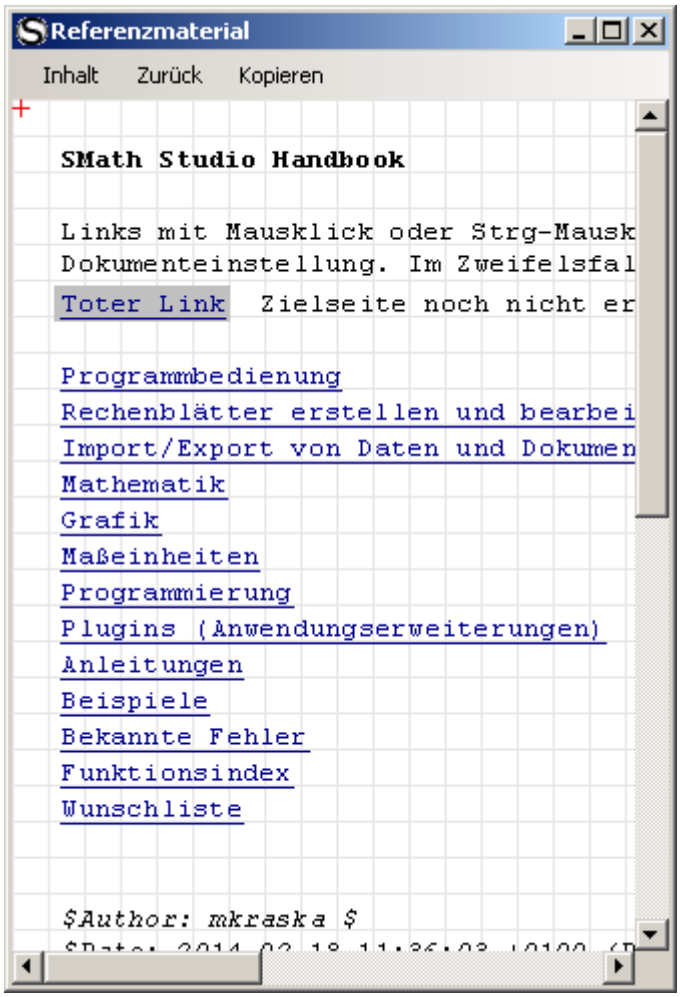


4. Im Dialog „edit fonts" sind alle vorhandenen Texte aufgelistet. Sie können einzeln oder insgesamt ausgewählt werden um dann ihre Schriftart zu modifizieren.



5. Mit dem Knopf Font [Aa] ruft man dann den Schriftauswahldialog auf. Dort wird mit OK bestätigt und dann im Dialog „edit fonts" mit [Apply] angewendet (erst dann sieht man, ob die Einstellung sinnvoll war)

3.5.3. Textbereiche mit Verweis (Hyperlink)

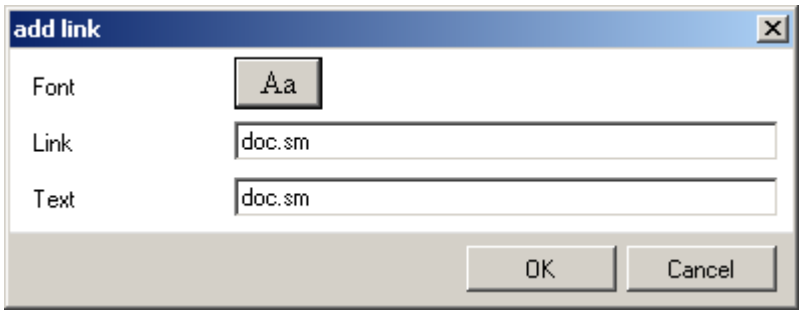


Textbereiche können als Verweis (Hyperlink) auf andere SMath-Dokumente dienen. Das interaktive Handbuch in SMath Studio macht davon umfangreich Gebrauch. Verweise sind wie auch im Internet üblich, durch Unterstreichung und blaue Schrift gekennzeichnet. Aktiviert werden Verweise durch Strg +Klick. Das aktuelle Dokument wird dann durch das Verweisziel ersetzt, eventuelle Änderungen werden ohne Nachfrage verworfen.

Soll auf ein anderes Ziel als eine SMath-Datei verwiesen werden, kann man den Hyperlink-Bereich (siehe Abschnitt 3.9) verwenden.

Textbereich mit Verweis einfügen

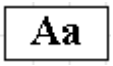
Dafür bietet das Plugin *Text Region Utilities* einen extra Punkt: **Einfügen> Textbereich (Utilities)>add a new link to a worksheet**. Dies öffnet einen Dialog, in welchem das Verweisziel und der sichtbare Text eingegeben werden können.



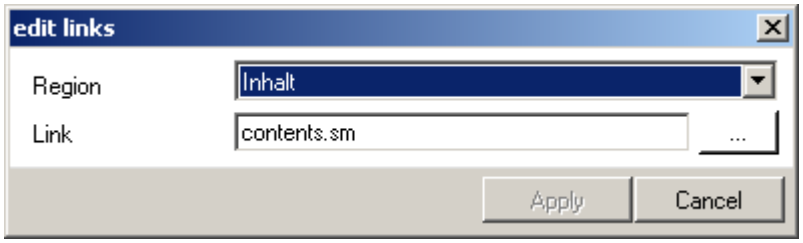
Verweisziel für bestehenden Textbereich angeben

Wie die Schriftart kann auch das Verweisziel nicht direkt für bestehende Textbereiche geändert werden. Daher wird auch hier der Umweg über den Toolbox-Bereich genommen.

1. Einfügen des Toolbox-Bereichs mit **Einfügen> Textbereich (Utilities)> insert a toolbox**

2. Neuberechnen des Dokuments mit 
3. Kontextmenü der Toolbox öffnen (rechte Maustaste) und den Eintrag „edit links” auswählen:
4. Der Dialog „edit links” bietet eine Auswahlliste mit allen bestehenden Textbereichen. Im Eingabefeld „Link” kann das Verweisziel angegeben werden (muss eine SMath-Datei, .sm sein). Mit dem Knopf rechts daneben kann das Verweisziel interaktiv ausgewählt werden.



3.6. Diagrammbereich

2D-Diagrammbereiche werden erzeugt mit

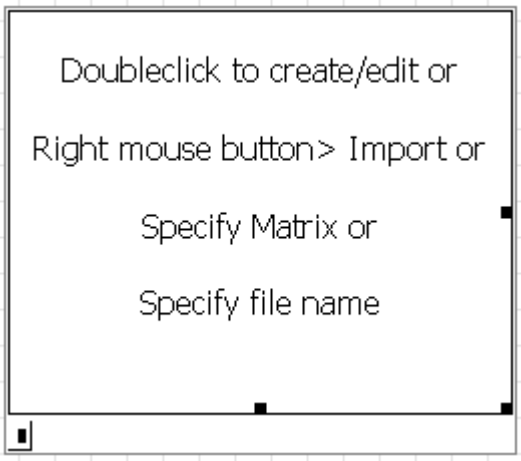
- @
- **Hauptmenü> Einfügen> Diagramm> Zweidimensional**

3D-Diagrammbereiche werden mit **Hauptmenü> Einfügen> Diagramm> Zweidimensional** erzeugt.

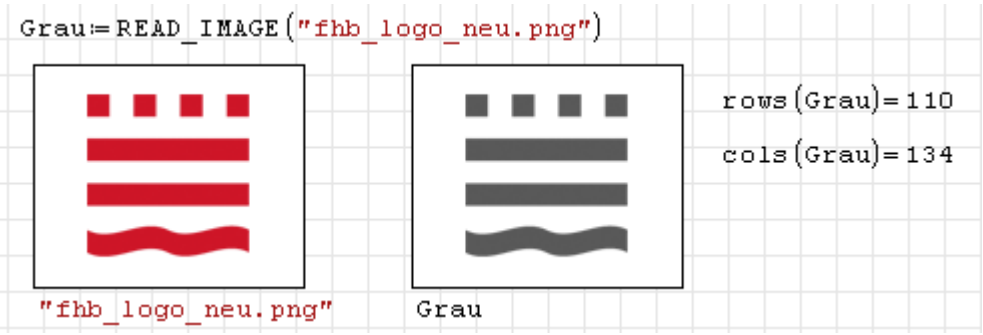
Details zur Erzeugung von Diagrammen sind im Kapitel 5 zu finden.

3.7. Image-Bereich

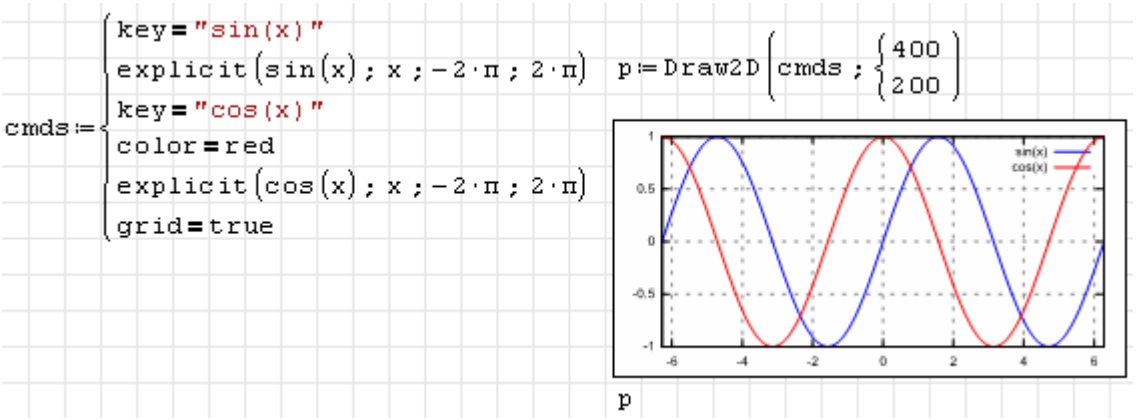
Dieser wird vom Plugin *ImageRegion* bereitgestellt und mit **Hauptmenü> Einfügen> Image** eingefügt.



In den Platzhalter kann ein Dateiname oder eine Matrix geschrieben werden.



Die Funktionen `Draw2D()` und `Draw3D()` aus dem *Maxima*-Plugin erzeugen Bilddateien, die mit dem Image-Bereich angezeigt werden können. Näheres zu diesen Diagrammfunktionen in Abschnitt 5.1.



3.8. Tabellenbereich

Achtung! Dieser Bereichstyp befindet sich in der Entwicklung. Es ist nicht sicher, dass Tabellen von späteren Versionen noch gelesen werden können.

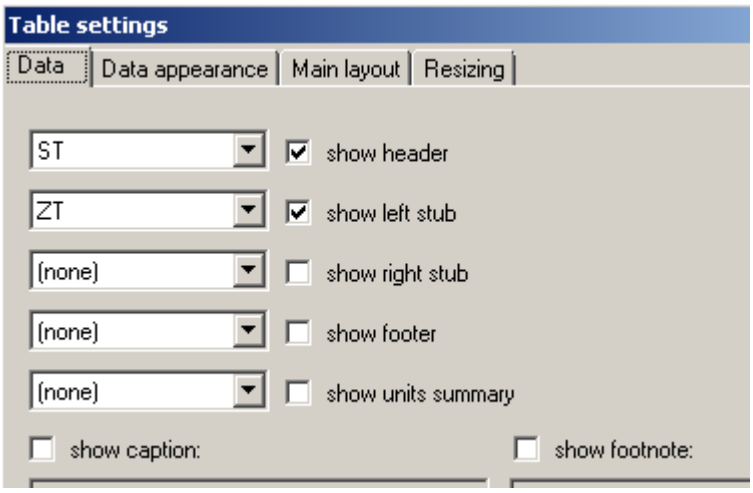
Das Plugin *TableRegion* stellt einen Tabellenbereich bereit, der für die Anzeige von Text, Zahlen oder SMath-Formeln genutzt werden kann. Eingefügt wird er mit **Hauptmenü> Einfügen> Table** oder mit ⇑ - Strg - T

In den Platzhalter trägt man eine Datenmatrix ein, für die Zeilentitel und Spaltentitel kann man Vektoren und Matrizen definieren, die dann im Einstellungsdialog ausgewählt werde können.

$$ZT := \begin{pmatrix} "Test1" \\ "Test2" \\ "Test3" \end{pmatrix} \quad ST := \begin{pmatrix} "S:R.e" & "S:A.G" \\ "S:'MPa" & "\%" \end{pmatrix} \quad Daten := \begin{pmatrix} 234 & 12 \\ 236 & 13 \\ 324 & 22 \end{pmatrix}$$

	R_e	A_G
	MPa	%
Test1	234	12
Test2	236	13
Test3	324	22

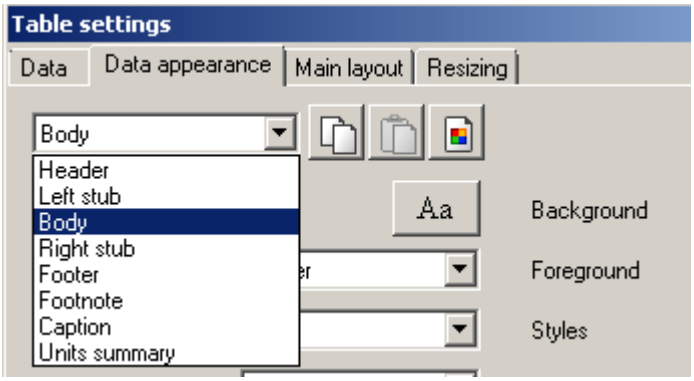
Doppelklick öffnet den Einstellungsdialog, der auf vier Seiten umfangreiche Anpassungen ermöglicht.



Auf der Seite „Data" kann man folgende Einstellungen vornehmen (bei einer Datenmatrix mit *m* Zeilen und *n* Spalten):

- *header*: Spaltentitel (oben), Matrix mit *n* Spalten und mindestens einer Zeile (im Beispiel 2 Zeilen)
- *left stub*: Zeilentitel (links), Matrix mit *m* Zeilen und mindestens einer Zeile
- *right stub*: Zeilentitel (rechts), Matrix mit *m* Zeilen und mindestens einer Zeile
- *footer*: Spaltentitel (unten), Matrix mit *n* Spalten und mindestens einer Zeile
- *units summary*: Maßeinheiten für die Zeilen oder Spalten (als Spaltenvektor mit *m* Elementen bzw. als Zeilenvektor mit *n* Elementen anzugeben).

Die Seite „Data appearance” erlaubt die Einstellung des Zahlenformats, der Schriftart und der Farben. Die Einstellungen beziehen sich auf den jeweils in der Liste ausgewählten Tabellenteil:



Im Kontextmenü (rechte Maustaste) des Tabellenbereichs sind folgende Einträge interessant:

- *Display input data*: Platzhalter anzeigen
- *Export> Save as*: Speichern als Grafikdatei oder HTML
- *Export> Copy to Clipboard*: Kopieren in die Zwischenablage, von dort als Tabelle oder Text in Word oder OpenOffice einfügbar

Im Tabellenbereich können auch symbolische Ausdrücke angezeigt werden, man gibt diese als Zeichenketten an, die mit `S:` beginnen, also z.B. `"S:a^b"`. Hier ein Beispiel, das die Leistungsfähigkeit des Tabellenbereichs demonstriert, dabei sind auch Schriftarten verstellt worden.

misc :=

1	π	i	sin(|x|)
0	{1, e}	0	[1 2; 3 (A B_A C^2); 5 6]
∞	0	1	1 m
0	"x⁰¹²³⁴ΣΒΤΒΦ"	0	x m
0	"S:x^n+x^(n-1)+…+x"	0	"S:B.A+2"

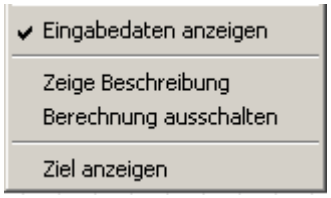
1	3,14159265358979	i	sin(|x|)
0	1 / 2,71828182845905	0	[1 2; 3 (A B_A C^2); 5 6]
∞	0	1	1 m
0	x⁰¹²³⁴ΣΒΤΒΦ	0	x m
0	x^n+x^(n-1)+…+x	0	B_A+2

Table 10

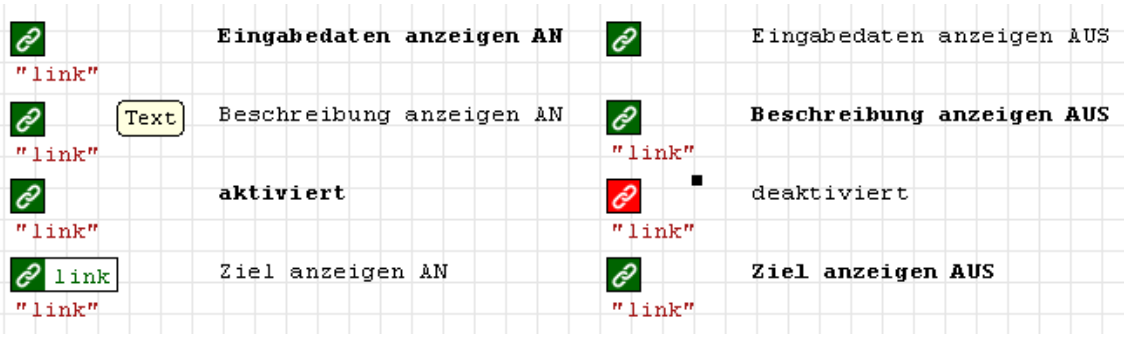
3.9. Hyperlink-Bereich

Dieser Bereich wird vom Plugin *Hyperlink Region* bereitgestellt. Eingefügt wird er mit **Hauptmenü> Einfügen> Hyperlink** oder mit Strg - H .

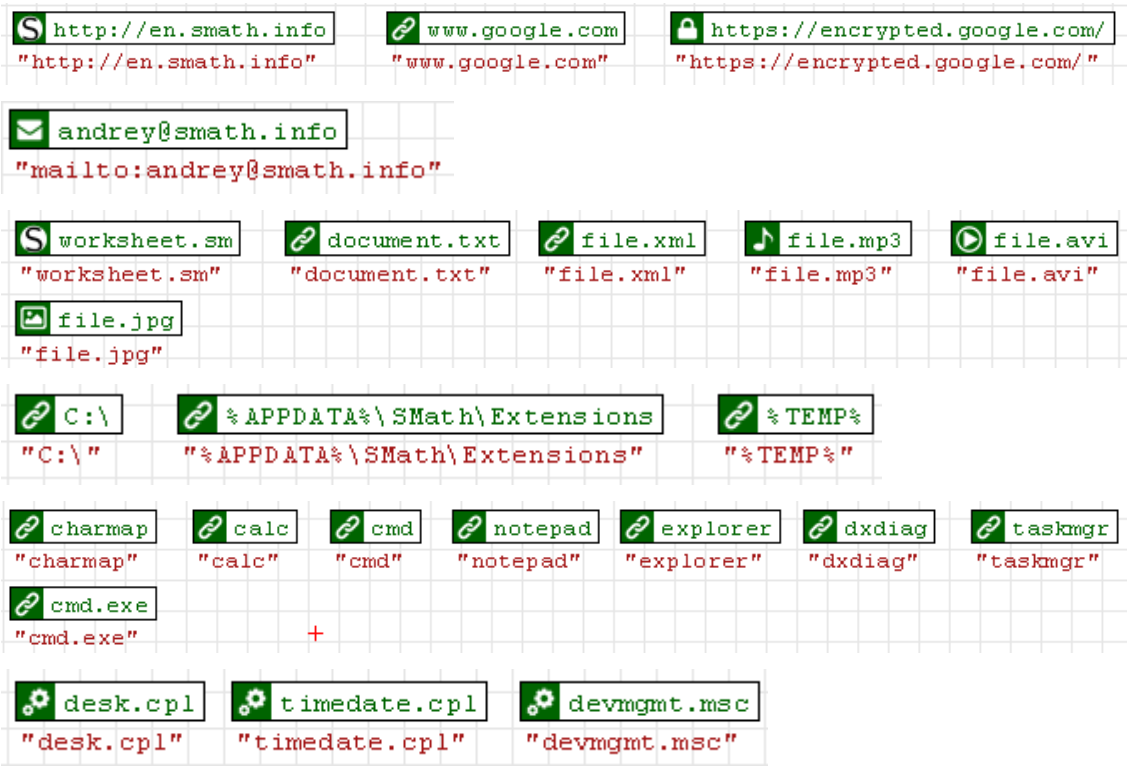
In den Platzhalter kann man das Verweisziel als Zeichenkette einfügen. Das Erscheinungsbild kann im Kontextmenü mit vier Optionen beeinflusst werden:



Im folgenden Beispiel sind die Standardeinstellungen hervorgehoben:



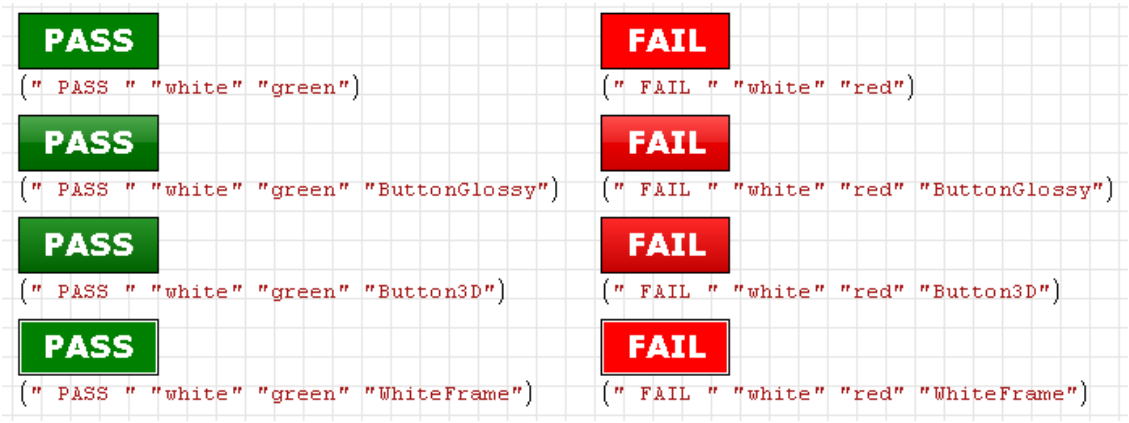
Hyperlink-Bereiche können auf die verschiedensten Ziele verweisen, z.B. Dateien, Verzeichnisse, Internet-Seiten oder Programme. Je nach Typ passt sich das Symbol im Hyperlink-Bereich an. Hier einige Beispiele:



Dateien und Programme werden mit dem Windows-Befehl *run* gestartet. Relative Pfadangaben beziehen sich auf das aktuelle Verzeichnis, das nicht unbedingt das Verzeichnis ist, in dem sich das aktuelle Dokument befindet.

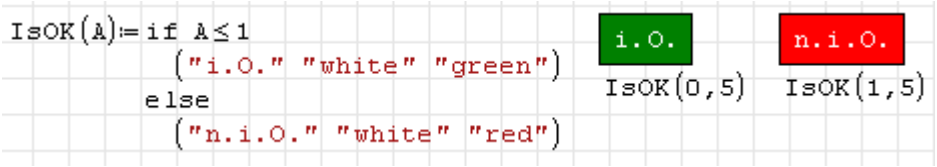
3.10. Bedingt formatierte Anzeige

Dieser Bereich wird vom Plugin *Conditionally Formatted Label* bereitgestellt. Das ist eine Anzeige, bei der Text, Textfarbe und Hintergrundfarbe durch den Inhalt der Variable im Platzhalter gesteuert werden kann. Eingefügt wird er über **Einfügen> Formatted label**.



Doppelklick auf den Bereich öffnet einen Dialog zur Auswahl der Schriftart und -größe.

Im folgenden Beispiel werden die Hintergrundfarbe und der Text abhängig von einem Zahlenwert gesetzt.



3.11. Interaktive Bereiche

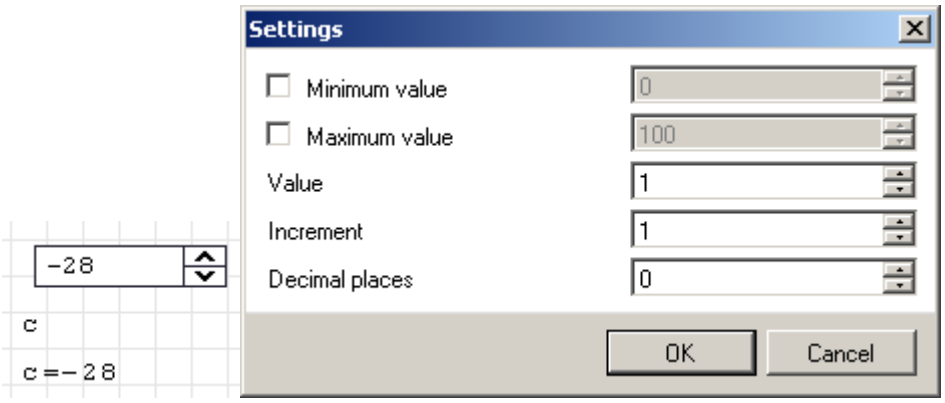
Es gibt besondere Bereiche für die interaktive Veränderung von Variablenwerten. Das geht natürlich zunächst immer mit einer Zuweisung in einem Formelbereich. In manchen Situation möchte man es allerdings bequemer haben oder dem Anwender eines Rechenblatts oder einer daraus erzeugten ausführbaren Anwendung einen bestimmten Wertebereich oder einzelne Werte zur Auswahl anbieten.

SMath Studio verfügt dafür über verschiedene Bereiche. Diese wurden von Davide Carpi bereitgestellt. Per Doppelklick erhält man ein Fenster mit Einstellungen.

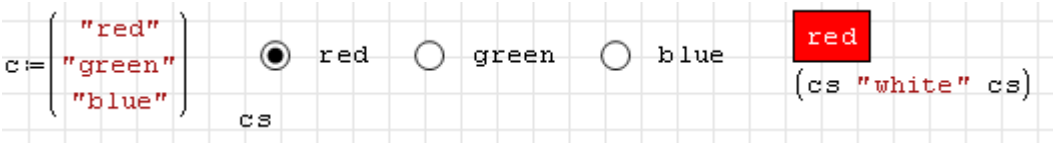
CheckBoxList Anordnung mehrere Schaltflächen, deren Anklicken den Wert des entsprechenden Matrixeintrags im Ergebnis zwischen 0 und 1 umschaltet.

ComboBoxList Auswahlbox, aus einem gegebenen Vektor kann ein Element ausgewählt werden. Durch Doppelklick in die aktive Zone bekommt man einen Einstellungs-Dialog.

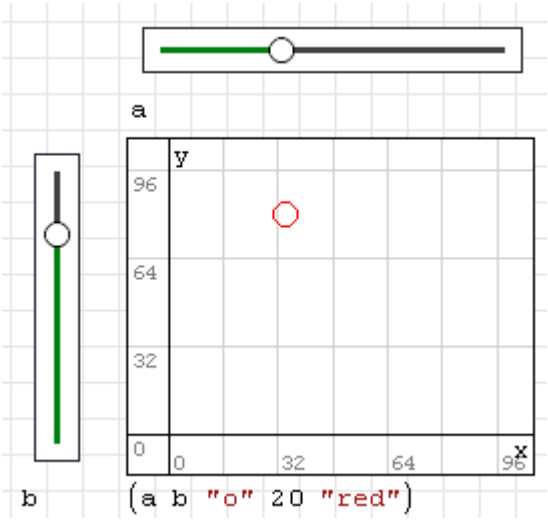
Numeric-up-down Zahleneingabefeld mit Schaltflächen zum schrittweisen Erhöhen oder Vermindern des Zahlenwerts oder dessen Änderung mit dem Mausrad. Man kann dabei Bereichsgrenzen und Schrittweite angeben.

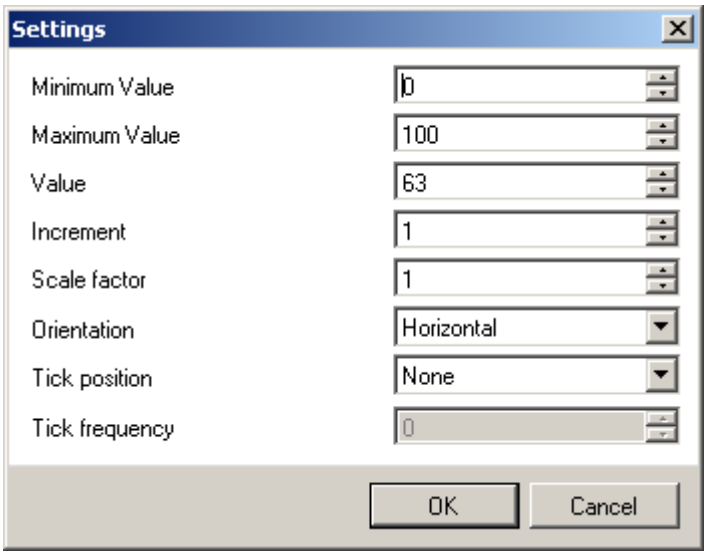


RadioButtonList Bereich, bei dem durch Drücken eines Knopfes genau eine der gegebenen Möglichkeiten ausgewählt wird. Die Möglichkeiten werden als Liste oder Matrix vorgegeben. Bei einer Matrix wird die erste Spalte für die Beschriftung verwendet. Die Ergebnisvariable im Platzhalter kann über ein Einstellungsfenster auf den Index, die Datenzeile oder einen bestimmten Spalteneintrag der Auswahl konfiguriert werden.



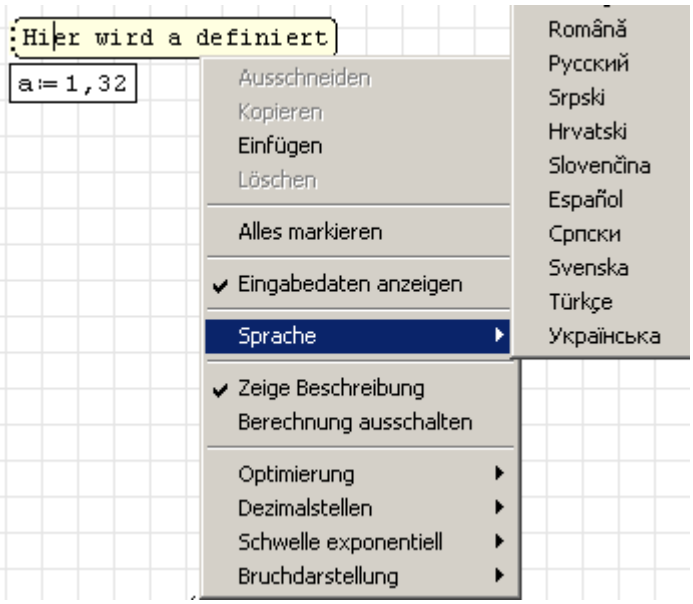
Slider Schieberegler zur Einstellung von Zahlenwerten durch Klicken oder Ziehen mit der Maus oder Drehen am Mausrad. Die Einstellungen findet man im Kontextmenü unter „Settings".



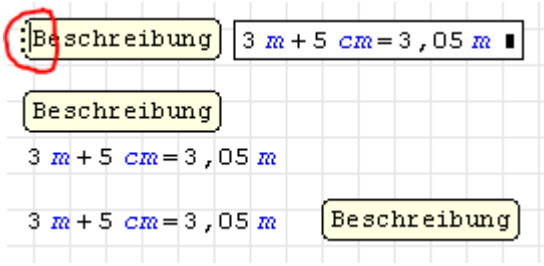


3.12. Beschreibungstexte

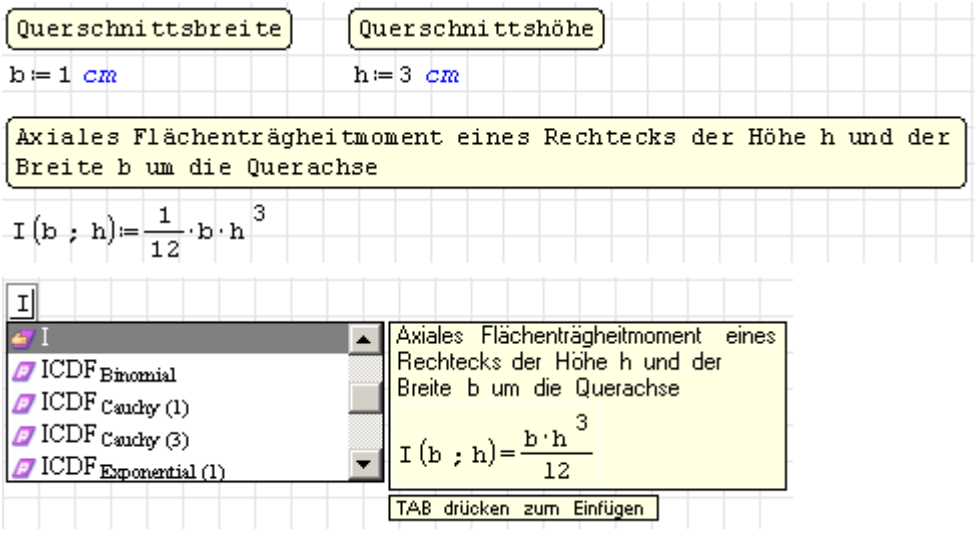
SMath besitzt einen Mechanismus, mit dem Beschreibungstexte für Bereiche mehrsprachig angegeben werden können. Die Beschreibung wird im Kontextmenü mit der Einstellung „Zeige Beschreibung” aktiviert. Die Sprache ist auf die Sprache der Benutzeroberfläche voreingestellt. Sie kann im Kontextmenü umgestellt werden, so dass für jede unterstützte Sprache ein Text eingegeben werden kann. Auf diese Weise kann man mehrsprachige Dokumente erzeugen, die in der jeweils vom Benutzer eingestellten Sprache angezeigt werden.



Beschreibungstexte können mit der Maus an verschiedene Seiten des jeweiligen Bereichs gezogen werden. Fassen Sie beim Ziehen an dem rot markierten, gepunkteten Rand an.



Werden in einem Formelbereich Variablen oder Funktionen definiert, so sind die Beschreibungstexte der dynamischen Auswahl bekannt (spätestens nach einmaliger Neuberechnung des Rechenblatts):



Beschreibungstexte an Definitionen können auch mit der Funktion `description()` abgefragt werden. Bei Funktionen muss die korrekte Anzahl von Argumenten angegeben werden, da sich die Definitionen je nach deren Anzahl unterscheiden können.

```
description(b)="Querschnittsbreite"
description(h)="Querschnittshöhe"
description(I(b ; h))="Axiales Flächenträgheitmoment eines Rechtecks
```

Beschreibungstexte werden im SMath Viewer (Kapitel 8) für Formelbereiche als Beschriftungstext benutzt, der den Formelinhalt ersetzt.

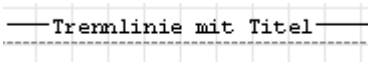
3.13. Trennlinien und Blattbereiche

Trennlinien und Blattbereiche dienen der Strukturierung des Rechenblatts. Beim Export in den SMath Viewer haben sie besondere Formatierungswirkung (Kapitel 8).

Trennlinien werden eingefügt mit **Hauptmenü> Einfügen> Trennlinie** oder mit Strg -↵ .

Im Rechenblatt erzeugen sie einen Seitenumbruch, der nur in der Druckansicht von Bedeutung ist. Im Viewer erzeugen sie eine neue Registerkarte.

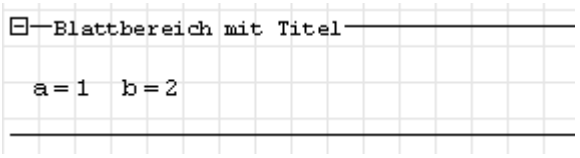
Trennlinien können mit einem Titel versehen werden, der im Viewer als Titel für die Registerkarte benutzt wird.



Trennlinien können wie alle anderen Bereiche auch nach Anklicken mit der Maus oder nach Markieren mit der Maus oder den Pfeiltasten verschoben werden.

Blattbereiche gruppieren den Inhalt zwischen einer oberen und einer unteren Trennlinie. Die obere Trennlinie hat eine Schaltbox zum Einklappen (Verbergen) des Blattbereichs und kann einen Titeltext erhalten. Die untere Trennlinie kann nicht beschriftet werden und erzeugt im Unterschied zur normalen Trennlinie keinen Seitenumbruch.

Ein Blattbereich wird eingefügt mit **Hauptmenü> Einfügen> Bereich**.



obere Trennlinie

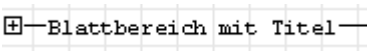
untere Trennlinie

Die Trennlinien können dann nach Bedarf verschoben werden, um den gewünschten Bereich einzugrenzen.

Blattbereiche können auch verschachtelt werden. Unabhängig davon, in welcher Reihenfolge sie erzeugt wurden, bilden immer die nächstgelegenen oberen und und unteren Trennlinien einer Bereichsebene.

Beim Löschen einer oberen oder unteren Trennlinie wird immer auch die andere zugehörige Trennlinie entfernt, so dass die Struktur stets eindeutig bleibt.

Ein Blattbereich kann durch Anklicken der Box - eingeklappt werden. Dann sieht man nur noch die obere Trennlinie mit dem Titel und und einer Box +, mit der man den Bereich wieder aufklappen kann.



Im Viewer werden eingeklappte Blattbereiche nicht angezeigt, aufgeklappte Bereiche werden mit einer Box umrahmt.

3.14. Mehrsprachige Arbeitsblätter

SMath Studio ist von vornherein auf die Unterstützung verschiedener Sprachen angelegt. Das betrifft nicht nur die Benutzeroberfläche, sondern auch die Arbeitsblätter selbst.

Für einige Elemente können alternative Sprachvarianten angegeben werden. Je nach dem, welche Sprache der Benutzer für die Oberfläche eingestellt hat, bekommt er die jeweilige Sprachvariante zu sehen, wenn sie vorhanden ist.

Alternative Sprachvarianten können angegeben werden für

- Dateieigenschaften
- Beschreibungstexte an Formel- und Diagrammbereichen
- Trennlinien und Überschriften von Blattbereichen

Die Spracheinstellung ist bei Beschreibungstexten, Trennlinien und Blattbereichsüberschriften über das Kontextmenü erreichbar. In nebenstehendem Bild ist die aktuelle Einstellung (Deutsch) ausgegraut und die weiteren verfügbaren Beschreibungen (Englisch, Russisch) sind angehakt.

Die Angaben zu den Dateieigenschaften (Titel, Autor, Beschreibung, Firma, Stichworte) können in mehreren Sprachen angegeben werden. Der Dialog wird aufgerufen mit **Hauptmenü> Datei> Eigenschaften**. Dort kann man unter „Sprache" auswählen, in welcher Sprache man die Angaben machen will. Theoretisch könnte man dies für alle verfügbaren Sprachen parallel tun.

Einige der Angaben wirken sich auf den Export des Arbeitsblatts als ausführbares Programm aus (SMath Viewer, siehe Kapitel 8)

3.15. Einstellungen

Die Einstellungen sind auf mehrere Dialoge verteilt.

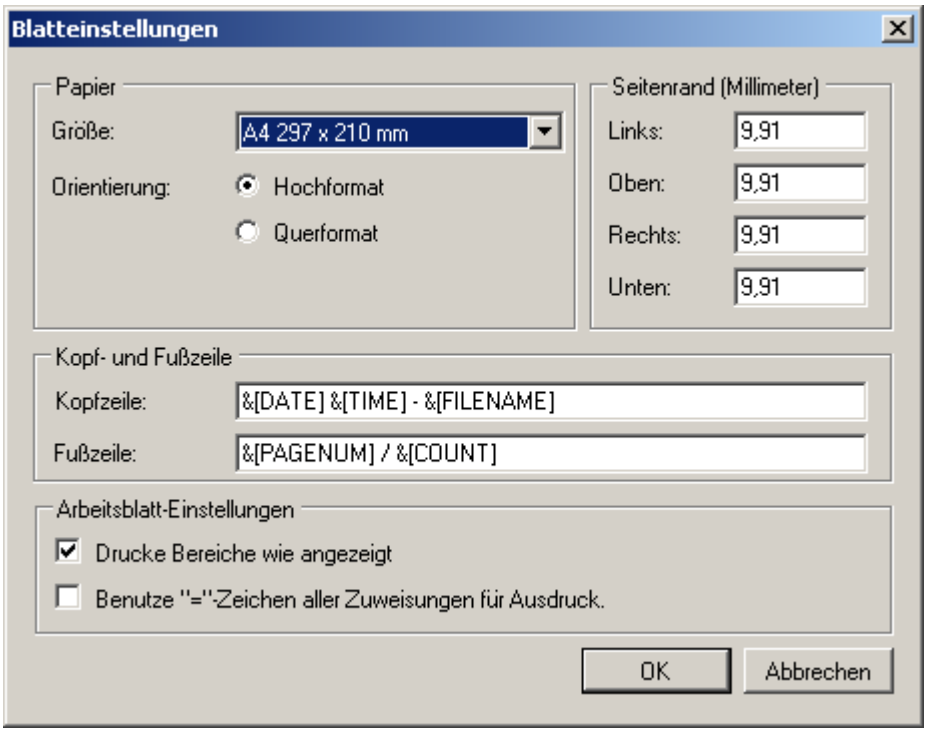
Blatteinstellungen Format, Ränder, Kopf- und Fußzeile, Abbildung, Abschnitt 3.15.1

Dateieigenschaften Standard-Dialog, nicht zwingend erforderlich, Abschnitt 3.15.2

Einstellungen Einstellungen zur Berechnung und zur Benutzeroberfläche, Abschnitt 3.15.3

3.15.1. Dialog „Blatteinstellungen"

Aufruf: **Hauptmenü> Datei> Blatteinstellungen**



Platzhalter für die Kopf- und Fußzeile:

Platzhalter	Inhalt
&[DATE]	Datum
&[TIME]	Uhrzeit
&[FILENAME]	Dateiname
&[PAGENUM]	aktuelle Seite
&[COUNT]	Seitenzahl

Tabelle 3.3.: Platzhalter für Kopf- und Fußzeile

Weitere Optionen:

Drucke Bereiche wie angezeigt Probieren!

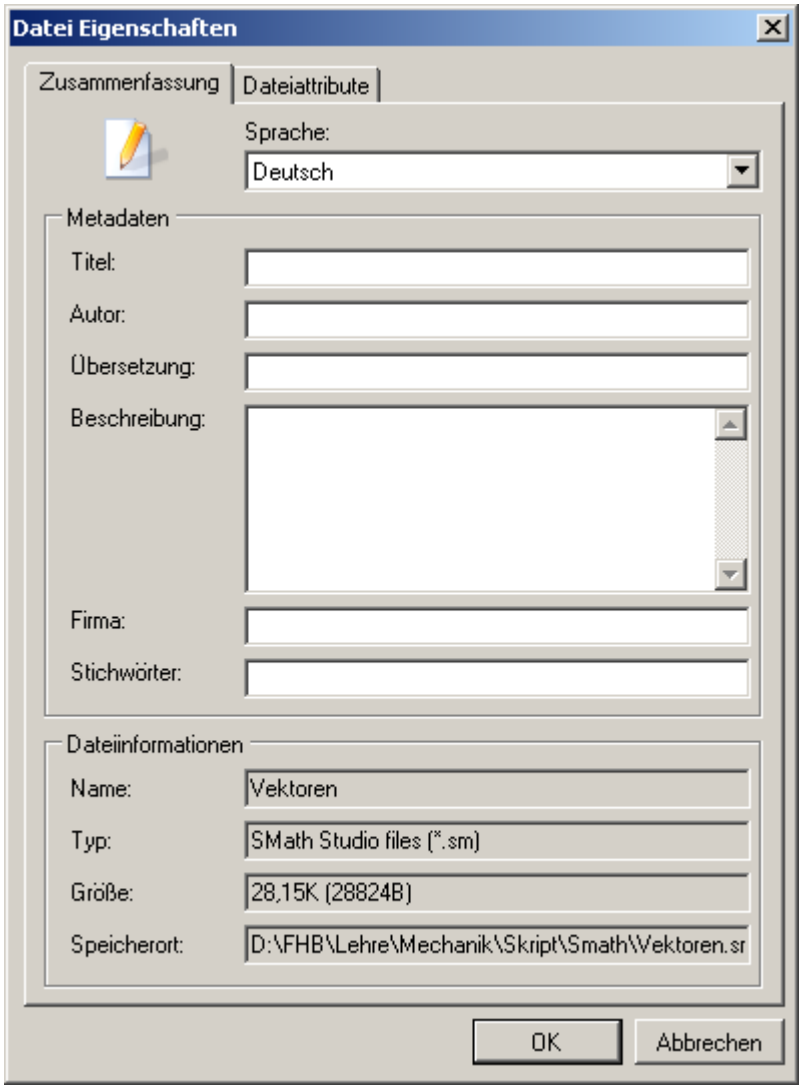
Benutze „="-Zeichen aller Zuweisungen für Ausdruck Standardmäßig ist := der Zuweisungsoperator. Bei der Druckausgabe kann man aber dass normale Gleichheitszeichen benuztzen. Davon ist abzuraten, denn damit wird eine Abweichung von Erscheinungsbild und Funktion erzeugt.

3.15.2. Dialog „Datei Eigenschaften"

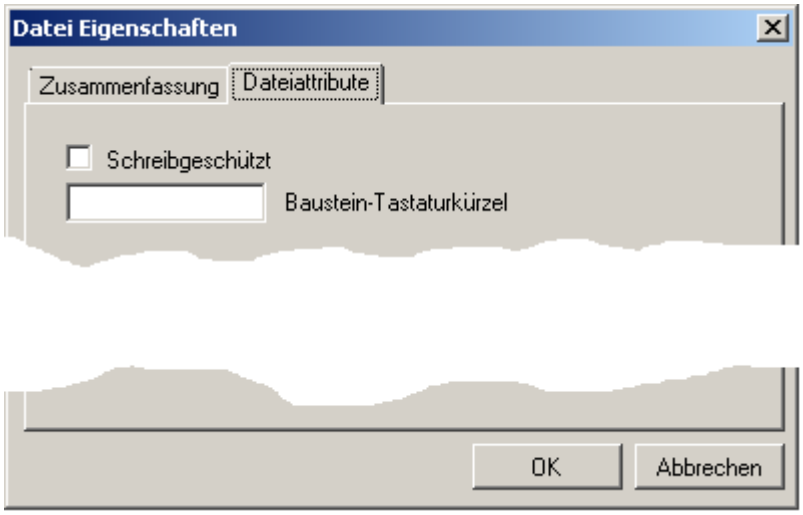
Aufruf: **Hauptmenü> Datei> Eigenschaften**

3.15.2.1. Einstellungsseite „Zusammenfassung"

Die Einträge sind alle selbsterklärend. Die Felder sind wichtig, wenn die Datei als Code-Baustein (snippet) oder Beispiel verwendet werden soll. Dann werden diese Daten im entsprechenden Auswahldialog angezeigt. Beim Export als ausführbare Anwendung (siehe Kapitel 8) werden die englischsprachigen Angabe zu Titel, Autor und Firma im Dateimanager angezeigt.



3.15.2.2. Einstellungsseite „Dateiattribute"

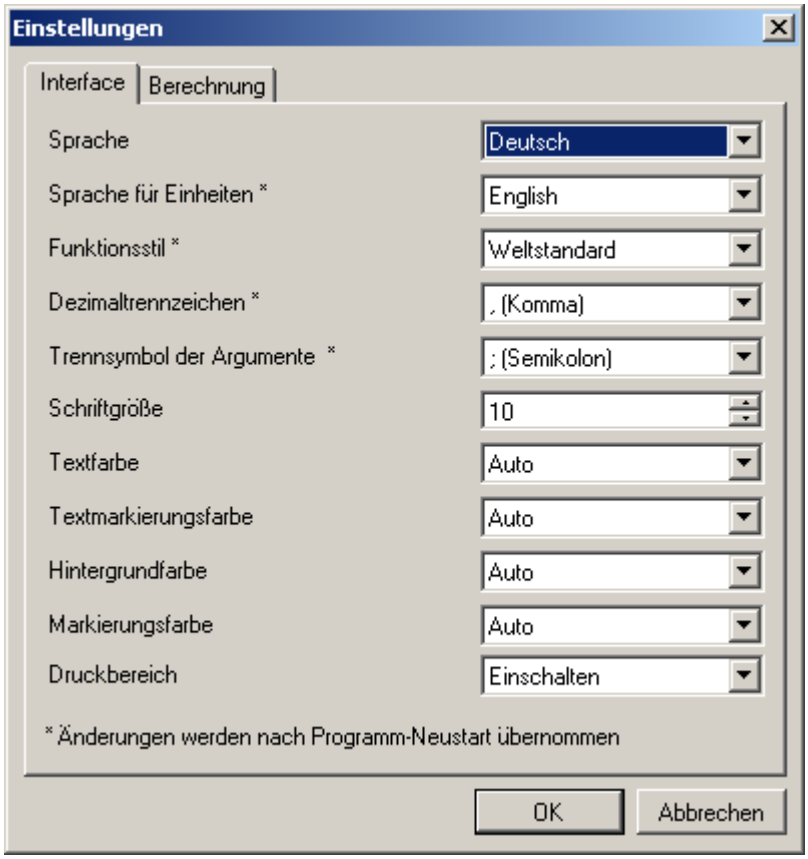


Baustein-Tastaturkürzel Name, unter dem das Rechenblatt in ein anderes Rechenblatt eingefügt werden kann (Code-Baustein, siehe Abschnitt 7.5). Dafür muss die Datei im Verzeichnis „Snippets" des Installationsverzeichnis liegen.

3.15.3. Dialog „Einstellungen"

Aufruf: **Hauptmenü> Extras> Einstellungen**

3.15.3.1. Einstellungsseite „Interface"



Sprache Sprache der Bedienoberfläche. Eine breite Auswahl wird angeboten. Die vorliegende Anleitung setzt „Deutsch" voraus.

Sprache für Einheiten Darstellung der Maßeinheiten. Neben dem Standard „English" gibt es dänische und russische Versionen (bei letzterer kommen dann kyrillische Buchstaben vor). Für deutsche Benutzer ist „English" angemessen.

Funktionsstil Steuert die Namen der Arkus-Funktionen (Umkehrfunktionen der Winkelfunktionen). „Weltstandard" bedeutet `asin`, `acos`, `atan` usw. „Europa" bedeutet `arcsin`, `arccos`, `arctan` usw.

Dezimaltrennzeichen Trennzeichen vor den Nachkommastellen von Dezimalbrüchen. Für deutsche Benutzer ist „Komma" angemessen. Alternativen sind „Punkt", wie es im Englischen üblich ist, oder „Semikolon", was eine nirgendwo gebräuchliche Variante ist.

Trennzeichen für Argumente Trennzeichen zwischen den Argumenten von Funktionen mit mehreren Parametern. Wenn das Dezimaltrennzeichen auf „Komma" steht, sollte hier „Semikolon" verwendet werden.

Schriftgröße Voreinstellung der Schriftgröße für neu eingefügte Text-oder Mathebereiche. Die Voreinstellung ist 10 Punkt. Die dargestellte Größe kann auch durch Mausrad-Zoom (nach Klick auf den Hintergrund des Rechenblatts) verstellt werden.

Textfarbe Voreinstellung für neu eingefügte Text- oder Mathebereiche. Es stehen vordefinierte Grundfarben und die Voreinstellung „Auto" zur Verfügung. Es gibt keinen guten Grund, hier rumzuspielen.

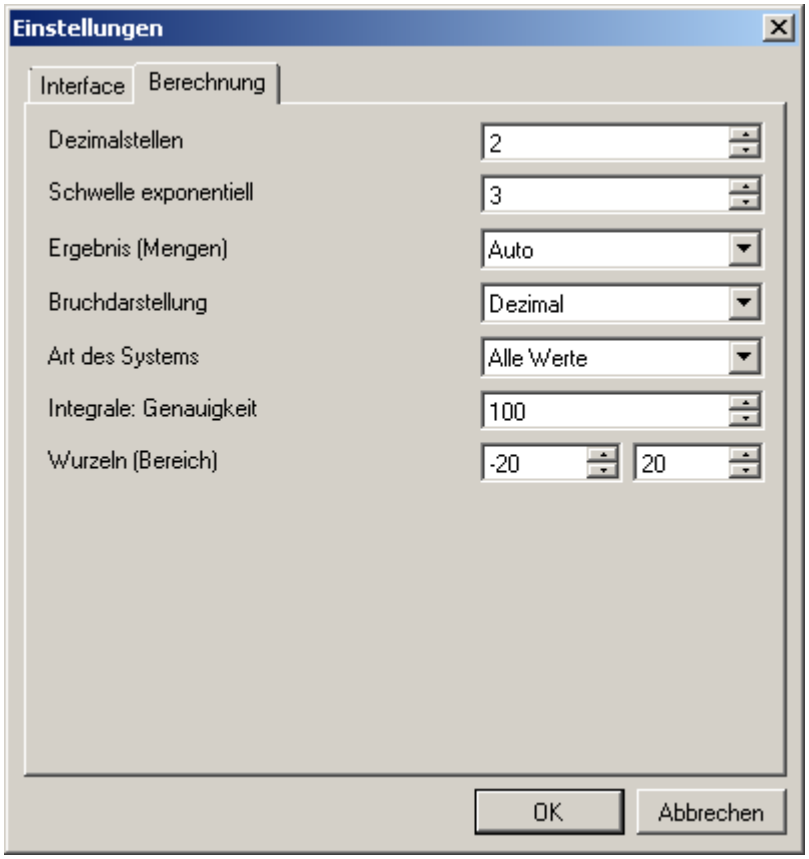
Textmarkierungsfarbe Voreinstellung für die Farbe, in der markierter Text in neu eingefügten Textbereichen erscheint. Es stehen vordefinierte Grundfarben und die Voreinstellung „Auto" zur Verfügung. Es gibt keinen guten Grund, hier rumzuspielen.

Hintergrundfarbe Voreinstellung für die Hintergrundfarbe neu eingefügter Text- oder Mathebereiche. Es stehen vordefinierte Grundfarben und die Voreinstellung „Auto" zur Verfügung. Es gibt keinen guten Grund, hier rumzuspielen.

Markierungsfarbe Voreinstellung für die Markierungsfarbe in neu eingefügten Text- oder Mathebereichen. Es stehen vordefinierte Grundfarben und die Voreinstellung „Auto" zur Verfügung. Es gibt keinen guten Grund, hier rumzuspielen.

Druckbereich Voreinstellung für die Anzeige von Seitenumbruch und Rand auf dem Rechenblatt. Diese kann für die laufende Sitzung mit **Hauptmenü> Anzeige> Druckbereich** geändert werden. Geänderte Voreinstellungen werden nach Neustart des Programms übernommen.

3.15.3.2. Einstellungsseite „Berechnung"



Dezimalstellen Voreinstellung für die angezeigten Nachkommastellen (0...15). Kann im Kontextmenü für markierte Mathebereiche geändert werden. Gegebenenfalls wird wissenschaftlich gerundet. Nullen am Ende des Nachkommateils werden nur angezeigt, wenn dies im Kontextmenü des jeweiligen Formelbereichs angewählt wird.

Schwelle exponentiell Voreinstellung für den Übergang auf abgetrennte Zehnerpotenzen (0...15). Gilt für negative und positive Exponenten. Kann im Kontextmenü geändert werden.

Ergebnis (Mengen) „komplex", „reell", „imaginär", „Auto", „Standard"

Bruchdarstellung Dezimal, gewöhnlich, Auto

Art des Systems „Alle Werte", „wiederholte Werte nur einmal"

Integrale: Genauigkeit Intervallzahl bei numerischer Integration, in 50er Schritten einstellbar.

Wurzeln (Bereich) Intervall, in welchem mit der Funktion `solve()` nach Nullstellen gesucht wird.

3.16. Die Zwischenablage

3.16.1. Aus SMath in die Zwischenablage kopieren

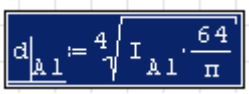
Ausdrücke in Formelbereichen und Text in Textbereichen können markiert und mit Strg-C in die Zwischenablage übernommen werden. Der Inhalt steht auch Anwendungen außerhalb SMath zur Verfügung.

Benutzt wird ein internes Textformat, das weitgehend den Tastatureingaben für den Ausdruck entspricht. Siehe dazu auch Abschnitt 3.4.6)

3.16.2. Übernahme von Ausdrücken in Forumbeiträge

Eine Anwendung ist die Übernahme von Ausdrücken aus SMath in Forumbeiträge. Wenn man sie im Forum-Beitragseditor zwischen die Marken `[MATH]` und `[/MATH]` setzt, dann werden sie wie in SMath formatiert angezeigt. Bei deutschen Einstellungen für Dezimaltrennzeichen und Argumenttrennzeichen (Komma und Semikolon) muss das Argument `lang=ENG` hinzugefügt werden.

Ein Beispiel: In SMath wird der Ausdruck

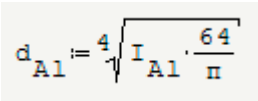


markiert und mit Strg - C in die Zwischenablage kopiert.

Im Forum-Editor fügt man mittels [BB/] die Marken `[MATH][/MATH]` ein, ergänzt `lang=ENG` und fügt den Inhalt der Zwischenablage zwischen den Marken ein, so dass dann dasteht:

```
[MATH lang=ENG]d.Al:nthroot(I.Al*64/π,4)[/MATH]
```

Das wird dann im Forum (mit „Preview" zu prüfen) so angezeigt:



3.16.3. Einfügen aus der Zwischenablage

Aus der Zwischenablage können folgende Arten von Daten eingefügt werden:

- Bilder: Rastergrafiken werden als Bildbereich eingefügt, der anschließend in der Größe geändert und verschoben werden kann.
- Text: Gültige SMath-Ausdrücke im internen SMath-Format können eingefügt werden:
  - in Formelbereiche (wenn die Einfügemarke darin steht) oder
  - als Formelbereiche (wenn die Einfügemarke auf dem Blatt steht).
- Enthält die Zwischenablage Leerzeichen, wird der Inhalt als Textbereich eingefügt. Befindet man sich in einem Formelbereich, wird das Einfügekommando ignoriert.

Das Einfügen von Text in Formelbereiche, etwa um eine durch Anführungszeichen "" erzeugte Zeichenkette mit Inhalt zu füllen, ist nicht möglich. Das ist insbesondere bei Dateioperationen lästig, wenn man Dateinamen einfügen will.

Mit folgendem Trick kann man sich helfen:

1. Einfügen des Textes als Textbereich
2. Ergänzung der Anführungszeichen
3. Kopieren des Textes mit Anführungszeichen
4. Einfügen in einen Formelbereich, der Ausdruck wird als Zeichenkette erkannt.

3.17. Verzeichnisse

In SMath Studio spielen drei Verzeichnisse eine Rolle, die jeweils mit einer Funktion angezeigt werden können:

```
Aktuelles Verzeichnis
   CurrentDirectory("")="D:\Programme\SMath\SMath Studio\"
Dokumentverzeichnis
   DocumentDirectory("")="D:\FHB\Software\SMath\SMath Skript\Bilder\"
Einstellungsverzeichnis
   SettingsDirectory("")="C:\Users\Kraska\AppData\Roaming\SMath\"
```

Außerdem ist noch das Installationsverzeichnis wichtig, welches man sich allerdings nicht anzeigen lassen kann.

Arbeitet man mit externen Dateien, dann ist es in der Regel sinnvoll, das aktuelle Verzeichnis auf das Dokumentverzeichnis zu setzen. So beziehen sich relative Pfadnamen auf das Verzeichnis, wo das aktuell offene Rechenblatt liegt:

```
dir:=CurrentDirectory(DocumentDirectory(""))
dir="D:\FHB\Software\SMath\SMath Skript\Bilder\"
```

Installationsverzeichnis

Das Installationsverzeichnis ist das Verzeichnis, wo SMath installiert wurde, z.B. *D:\Programme\SMath*. Dort liegt die ausführbare Datei *SmathStudio_Desktop.exe*. Wird SMath Studio über das Startmenü gestartet, dann ist dieses Verzeichnis auch das aktuelle Verzeichnis (*current directory*).

Im Installationsverzeichnis befinden sich weitere Unterverzeichnisse:

book Hier liegt das im Standardumfang mitgelieferte Referenzmaterial (Mathematik-Formelsammlung).

entries Enthält momentan nur die Maßeinheitendefinition.

examples Mitgelieferte Beispieldateien.

extensions In der portablen Version legt die Erweiterungsverwaltung (siehe Kapitel 9) hier die aus der Online-Galerie heruntergeladenen Erweiterungen (Plugins, Code-Bausteine, Beispiele) ab. Hier befinden sich auch die vorinstallierten Plugins und das interaktive Handbuch.

lang Texte für die verschiedenen Sprachversionen der Benutzeroberfläche.

plugins Hier liegen die dynamisch gelinkte Bibliotheken (.dll) des Standard-Programms.

snippets Hier liegen die als .sm Dateien bereitgestellten Bausteine (Snippets), siehe Kapitel 7.5.

Maxima... Hier liegt in der FHB-Distribution das Maxima-Installationsverzeichnis.

Einstellungsverzeichnis

Dies ist das Verzeichnis, in welchem SMath benutzerspezifische Daten ablegt, darunter auch die Datei *settings.inf* mit den Programmeinstellungen. Der Erweiterungsmanager legt hier die Verzeichnisstruktur *Extensions* an, in der die heruntergeladenen Erweiterungen abgelegt werden.

Dieses Verzeichnis wird zum Beispiel von den Funktionen `wfile()` (Seite 329), `rfile()` (Seite 329) oder `dfile()` (Seite 269) benutzt.

Man kann das Verzeichnis mit der Funktion `SettingsDirectory()` aus dem Plugin *customFunctions* (siehe Abschnitt 9.4.3) ermitteln:

```
SettingsDirectory(■)="D:\FHB\Software\SMath\2014 test SMath and Maxima\"
SettingsDirectory(0)="D:\FHB\Software\SMath\2014 test SMath and Maxima\"
```

Bei der Installer-Version liegt dieses Verzeichnis in der Windows-Anwenderdatenstruktur, bei der portablen Version ist das Einstellungsverzeichnis mit dem Installationsverzeichnis identisch.

Aktuelles Verzeichnis

Einige Dateifunktionen wie `ImportData()` (siehe Seite 298) oder `exportData.CSV()` benötigen Dateinamen oder Pfade, die entweder relativ oder absolut sein können.

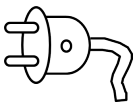
Relative Namen beziehen sich auf das aktuelle Verzeichnis, absolute Namen enthalten den vollständigen Pfad.

Das aktuelle Verzeichnis ist das Verzeichnis, wo SMath gestartet wurde. Wird das Programm ohne zu öffnende Datei gestartet, dann ist das Installationsverzeichnis, also da, wo die ausführbare Datei liegt, das aktuelle Verzeichnis.

Wurde eine .sm-Datei als registrierter Dateityp im Dateimanager gestartet, dann wird das Verzeichnis, wo die .sm-Datei liegt, das aktuelle Verzeichnis.

Das aktuelle Verzeichnis ändert sich nicht, wenn eine Datei an anderer Stelle gespeichert wird.

Mit der Funktion `CurrentDirectory()` aus dem Plugin *MathcadFileAccess* (siehe Abschnitt 9.4.9) kann das aktuelle Verzeichnis ermittelt und gesetzt werden:



```
CWD:=CurrentDirectory("")

CurrentDirectory("")="D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"

CWD="D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"

CurrentDirectory("d:\tmp")="d:\tmp\"

CurrentDirectory(CWD)="D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"
```

Dokumentverzeichnis

Das Dokumentverzeichnis ist das Verzeichnis, in welchem das aktuell offene Dokument liegt. Man kann es mit der Funktion `DocumentDirectory()` angzeigen. Mit der Kombinati-

on `CurrentDirectory(Documentdirectory(""))` kann man sicherstellen, dass das aktuelle Verzeichnis auf das Dokumentverzeichnis gesetzt ist.

Damit das funktioniert, muss allerdings das aktuelle Rechenblatt bereits gespeichert und nach dem Speichern mindestens einmal ausgeführt worden sein. Auch nach dem Wechseln in ein anderes Rechenblatt muss man dieses zunächst ausführen (neu berechnen), damit das Dokumentverzeichnis richtig gesetzt ist.

Systemverzeichnisse

Unter Windows können einige Systemverzeichnisse mit der Funktion `GetFolderPath()` aus dem Plugin *MathcadFileAccess* (siehe Abschnitt 9.4.9) bestimmt werden. Die Funktion muss mit einer Kennzahl aufgerufen werden und liefert dann den zugehörigen Pfad als Zeichenkette:

```
PROGRAMS := 2    PERSONAL := 5    FAVORITES := 6    STARTUP := 7    RECENT := 8    SENDTO := 9
STARTMENU := 11    MUSIC := 13    DESKTOP := 16    TEMPLATES := 21    APPDATA := 26
LOCALAPPDATA := 28    INTERNETCASH := 32    COOKIES := 33    HISTORY := 34
COMMONAPPDATA := 35    SYSTEM := 37    PROGRAMFILES := 38    PICTURES := 39
COMMONPROGRAMFILES := 43
```

Hier einige Beispiele:

```
GetFolderPath(PERSONAL) = "C:\Users\Kraska\Documents"
GetFolderPath(FAVORITES) = "C:\Users\Kraska\Favorites"
GetFolderPath(COOKIES) = "C:\Users\Kraska\AppData\Roaming\Microsoft\Windows\Cookies"
GetFolderPath(HISTORY) = "C:\Users\Kraska\AppData\Local\Microsoft\Windows\History"
GetFolderPath(COMMONAPPDATA) = "C:\ProgramData"
GetFolderPath(SYSTEM) = "C:\Windows\system32"
```

3.18. Import/Export

SMath Studio verfügt über umfangreiche Möglichkeiten zum Import oder Export von Rechenblättern, Daten und Bildern. Diese Funktionen sind an verschiedenen Stellen verfügbar:

- Im Hauptmenü unter **Datei> Speichern unter**. Dort kann das ganze Rechenblatt in verschiedenen Formaten exportiert werden.
- Als Schreib- oder Lesefunktionen für bestimmte Dateitypen (Text, Tabellenkalkulation, Bilder, Audio, allgemeine Binärdateien).
- Der Image-Bereich kann externe Bild-Dateien anzeigen.
- Der Snapshot-Bereich kann Ausschnitte des Rechenblatts als Bilddatei exportieren.

Das aktuelle Rechenblatt kann über **Hauptmenü> Datei> Speichern unter** in verschiedenen Formaten abgespeichert/exportiert werden.

Format	Endung	Export	Import	Bemerkung
SMath Studio XML-Format	sm	++	++	
komprimiertes SMath Studio XML-Format	smz	++	++	
MathCad XML-Format	xmcd	o	o	Plots können nicht exportiert werden
PDF	pdf	++		mit Druckoptionen
HTML	htm	++		Formeln und Bilder als Grafik
Rastergrafik	bmp, png, gif, jpg	++		Ein einziges Bild ohne Seitenumbruch
XeLaTeX	tex	++		mit XeLaTeX zu compilieren Erfordert Plugin *DataExchange* (Seite 215)
OpenOffice Text	odt	+		Erfordert Plugin *DataExchange* (Seite 215)
Ausführbare Datei	.exe	+		Siehe Kapitel 8. Das Maxima-Plugin wird nicht unterstützt.

Tabelle 3.4.: Unterstützte Export- und Importformate für SMath-Dokumente

Das Plugin *DataExchange* (siehe Abschnitt 9.4.4) stellt Exportformate für Formeln und Ergebnisse bereit.

Format	Endung	Exportfunktion	Importfunktion
Text mit Spaltentrennzeichen	csv	`exportData.CSV()` 275	`importData()` 298
MS Excel	xlsx	`exportData.XLSX()` 278	`importData.XLSX()` 299
OpenOffice Calc	ods	`exportData.ODS()` 277	`importData.ODS()` 298
OpenOffice Formel	odf	`exportData.ODF()` 276	

Tabelle 3.5.: Export- und Importformate für Ergebnisse und Formeln.

Der odf- oder LaTeX-Export ist hilfreich, wenn man Formeln in Texte übernehmen will.

Im Plugin MathcadFileAccess (Abschnitt 9.4.9) finden sich Funktionen für das Lesen und Schreiben von Binärdaten sowie für das Lesen von Rasterbilddaten.

Format	Endung	Exportfunktion	Importfunktion
Bilddaten	png, tif, bmp...		READ_IMAGE() 326
			READ_RED() 326
			READ_GREEN() 326
			READ_BLUE() 325
			READRGB() 326
	bmp		READ_BMP() 327
Audiodaten	wav	WRITEWAV() 343	READWAV() 328
Binärdateien (Matrizen)		WRITEBIN() 343	READBIN() 327

Tabelle 3.6.: Mathcad-kompatible Datenaustauschfunktionen.

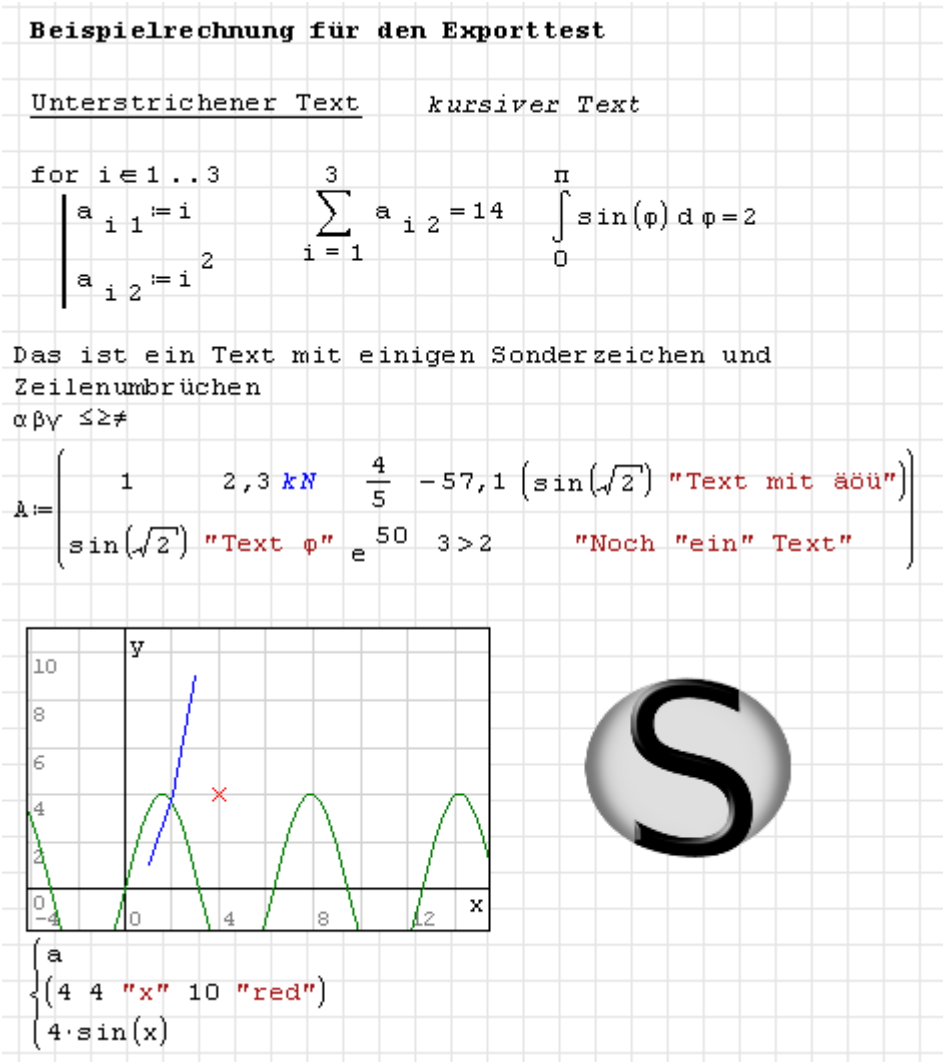
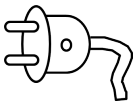


Abbildung 3.4.: Rechenblatt zur Demonstration der Exportfunktionen

3.18.1. Schreiben und Lesen von XLSX- und ODS-Tabellen



Diese Funktionen sind mit dem Plugin *DataExchange* (siehe Abschnitt 9.4.4) verfügbar.

Die Exportfunktionen `exportData.XLSX()` und `exportData.ODS()` schreiben den Inhalt des ersten Arguments in eine Datei des passenden Typs. Typischerweise ist dieses Argument eine Matrix, die je nach Inhalt der Einträge unterschiedliche Exportergebnisse erzeugt:

- Zahlenkonstanten und Zeichenketten (z.B. `4,5`, `"ein Beispiel"`) werden als numerische bzw. Textzelle direkt übernommen.
- Ausdrücke werden zunächst numerisch ausgewertet. Ist das Ergebnis eine Zahl, wird diese exportiert (z.B. `2,0943951023932`).
- Kann das Objekt nicht zu einer Zahl oder Zeichenkette ausgewertet werden, wird die interne symbolische Darstellung des Ausdrucks als Formel exportiert (z.B. `mat(1,2,2,1)`).

Die Importfunktionen `importData.XLSX()` und `importData.ODS()` lesen ein anzugebendes Blatt aus einer anzugebenden Datei. Optional kann eine Zelladresse oder ein Adressbereich angegeben werden.

Die Adressen sind als Zeichenkette aus Spalten-Kennbuchstabe und Zeilennummer anzugeben, also z.B. `"A1"`, `"B10"`.

Die Zellen werden je nach Inhalt unterschiedlich importiert:

- Zahlenkonstanten werden als solche übernommen.
- Textzellen werden als Zeichenketten importiert.
- numerisch auswertbare Formelzellen werden als Zahlen importiert.
- Formelzellen der Form `="mat(1,2,2,1)"` werden als symbolische Ausdrücke importiert.

Symbolische SMath-Ausdrücke in Rechentabellen

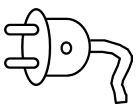
Symbolische Ausdrücke werden in ODS- oder XLSX-Dateien als Formelzelle in der Form `="mat(1,2,2,1)"` gespeichert oder erwartet, also mit Gleichheitszeichen und eingeschlossen in Anführungszeichen.

Dabei werden die Standardeinstellungen für Dezimaltrenner (Punkt) und Argumenttrenner (Komma) verwendet. Die in diesem Handbuch empfohlene Einstellung (Komma und Semikolon) wird ignoriert.

In symbolischen Ausdrücken enthaltene Zeichenketten werden durch doppelte Hochkommas (') anstelle von Gänsefüßchen eingeschlossen, da letztere bereits den Ausdruck selbst markieren.

3.18.2. Speichern als XeLaTeX

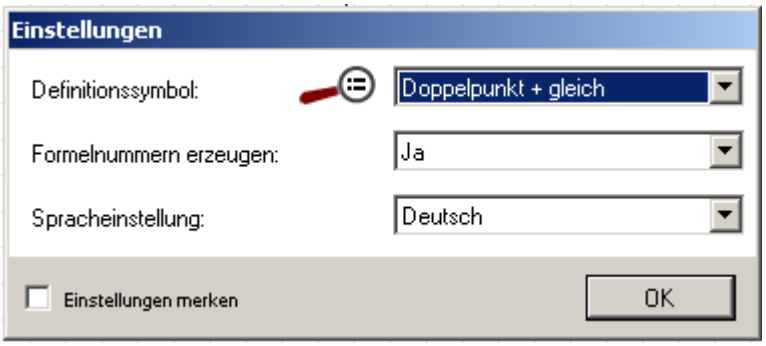
Hauptmenü> Datei> Speichern unter: Dateityp wählen:



Es erscheint ein Dialog mit Einstellungen für

- Darstellung des Zuweisungsoperators (:= in SMath), Optionen sind: :=, = oder $\stackrel{\text{def}}{=}$
- Numerierte oder nicht numerierte Formeln
- Spracheinstellung in XeLaTeX

Die aktuellen Einstellungen können als Standard gesetzt werden. Nach Drücken von OK wird die Datei exportiert.



Für die Weiterverarbeitung der Datei benötigt man eine LaTeX-Installation, z.B. MiKTeX unter Windows. Die Kompilation funktioniert nur mit einer wirklich aktuellen Installation (mindestens XeLaTeX 0.9998). Zudem müssen die Pakete XITS und Libertine nachinstalliert werden. (Stand Februar 2013).

XeLaTeX ist ein LaTeX-Kompiler mit kompletter Unicode-Unterstützung und weiteren Fähigkeiten, die ihn befähigen, Texte in beliebigen Sprachen zu übersetzen. Er produziert PDF-Dateien.

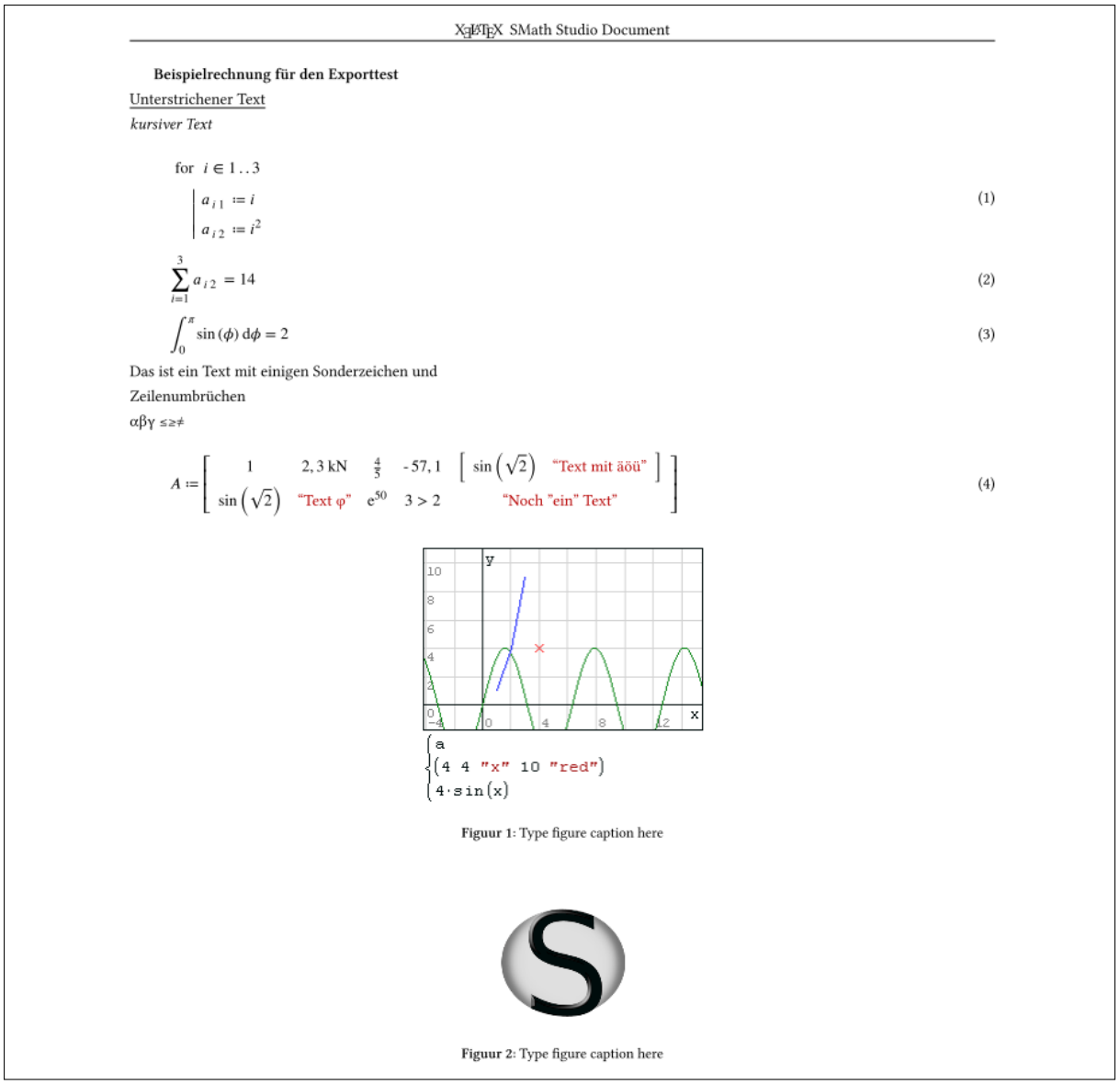
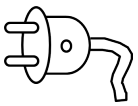


Abbildung 3.5.: Kompilierter XeLaTeX-Export (pdf)

3.18.3. Speichern als OpenDocument Text

Hauptmenü> Datei> Speichern unter: Dateityp wählen:

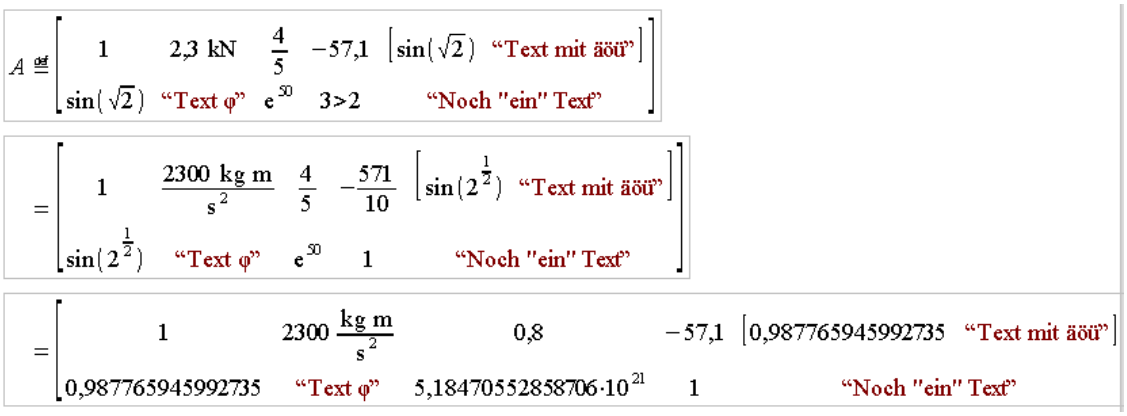


Es erscheint ein Dialog mit Einstellungen:

- Formeln können optional als Grafik exportiert werden. Das ist dann sinnvoll, wenn man sie nicht weiterbearbeiten
- Text kann optional als Grafik exportiert werden
- Ausdrücke können optional ausgewertet werden.

Wenn „Definitionen auswerten" eingestellt ist, dann werden drei Versionen jeder Definition ausgegeben:

1. Definition, wie sie im Rechenblatt steht,
2. symbolisch vereinfachte Version,
3. numerisch ausgewertete Version.

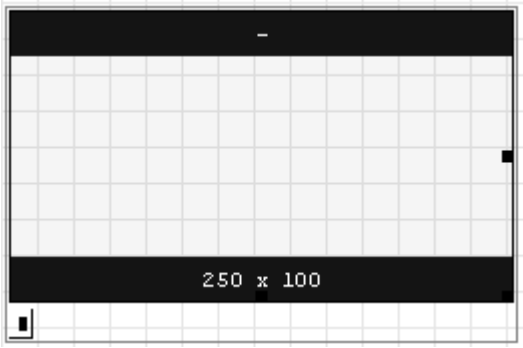


Ansonsten wird nur die eigentliche Definition ausgegeben, so wie in SMath angezeigt.

3.18.4. Snapshot-Bereich für Bildschirmfotos (Screenshots)

Wenn SMath-Rechnungen in Arbeiten dokumentiert werden sollen, die nicht selbst mit SMath Studio erstellt werden, dann kann man dies mit Screenshots tun. Das vorliegende Handbuch enthält hunderte davon.

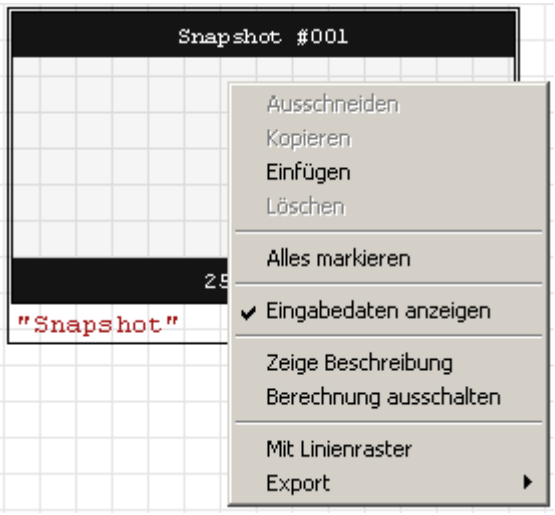
Ein Snapshot-Bereich wird eingefügt mit **Einfügen> Snapshot** oder Strg - F



Ein solcher Bereich kann den Inhalt des Rechenblatts zwischen den zwei schwarzen Balken als Bilddatei exportieren.

Im Platzhalter ist ein Zeichenkette anzugeben, aus dem dann die Dateinamen beim Export gebildet werden. Position und Größe des Bereichs werden wie bei Diagrammen angepasst. Um den Bereich als ganzes zu verschieben, wird er blau markiert. Zur Größenänderung muss der Bereich aktiviert werden (zwischen die Streifen klicken) und kann dann an den Randmarken links und unten gezogen werden.

Im unteren Balken wird die Bildgröße in Pixeln angezeigt, der obere Balken zeigt den Dateinamen des Bildes. Mehrere Snapshot-Bereiche mit gleichem Text im Platzhalter bilden einen Satz, innerhalb derer die Bilder eine dreistellige Nummer erhalten, die Bestandteil des Dateinamens wird (*name_xxx.png*).



Im Kontextmenü (rechte Maustaste) stehen die folgenden Optionen und Funktionen zur Verfügung:

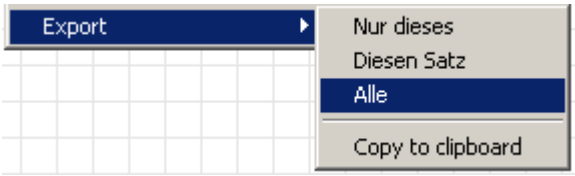
Eingabedaten anzeigen: Damit wird der Platzhalter für den Dateinamen ein- oder ausgeblendet.

Zeige Beschreibung: Wie jeder Bereich kann auch der Snapshot-Bereich mit einer Beschreibung versehen werden (siehe Abschnitt 3.12).

Berechnung ein-/ausschalten: Damit wird der Bereich aktiviert oder deaktiviert. Ein deaktivierter Bereich wird bei der Erzeugung der Dateinamen nicht mitgezählt. Aktivieren und Deaktivieren von Bereichen ändert also die Dateinamen der erzeugten Screenshots.

Mit Linienraster: Das Linienraster des Rechenblatts kann beim Screenshot abgeschaltet werden.

Export: Hier wird der eigentliche Export veranlasst. Es stehen folgende Möglichkeiten zur Verfügung:



Nur dieses: Nur das aktuelle Bild exportieren

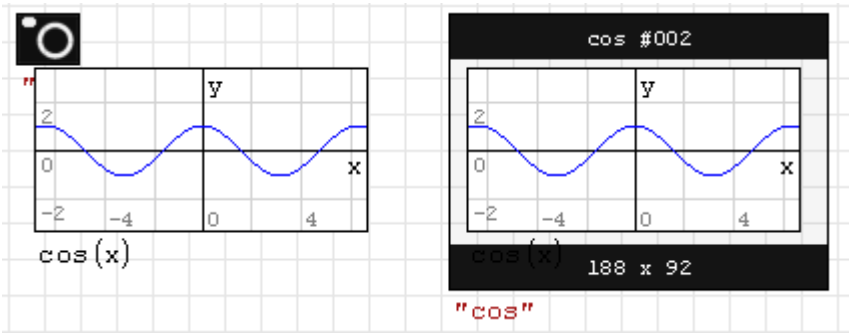
Diesen Satz: Alle Bilder mit dem gleichen Namen exportieren. Die Bilder werden fortlaufend numeriert: *name_xxx.png*

Alle: Alle Snapshot-Bereiche des Rechenblatts exportieren,

Zwischenablage: Bild in die Zwischenablage übernehmen (alternativ zum Windows Snipping Tool)

Beim Export öffnet sich ein Dialog zur Auswahl des Exportverzeichnisses, dieses ist voreingestellt auf das Dokumentverzeichnis (da, wo das aktuelle Rechenblatt liegt). Daher wird in der Regel einfach eine Bestätigung reichen.

Um die Bearbeitung der unter dem Snapshot-Bereich liegenden Bereiche zu ermöglichen, kann der Bereich durch Doppelklick geschrumpft und wieder erweitert werden.



4. Mathematik

4.1. Begriffe

Adressindex Tiefgestellter Index für die Angabe von Zeile und eventuell Spalte bei vektor- oder matrixwertigen Bezeichnern und Ausdrücken. Er wird durch Eingabe von `[` (eckige Klammer auf) erzeugt und ist durch einen Zwischenraum abgetrennt.

Ausdruck Besteht aus Zahlen, Variablen, Operatoren, Funktionen und ihren Argumenten und kann zu einem Ergebnis ausgewertet werden.

Bezeichner Name für eine Variable oder eine Funktion.

Funktion Berechnet aus ihren Argumenten einen Rückgabewert. Die Argumente stehen in Klammern, getrennt durch das Argumenttrennzeichen (in der deutschen Standardeinstellung das Semikolon)

Operator Spezielles Symbol für eine Funktion, wie z.B. +, -, *, /, $\sum$

Textindex Tiefgestellter Bestandteil eines Bezeichners, der zur näheren Erläuterung dient. Die Tiefstellung wird durch Einfügen eines Punkts erzeugt. Zwischen Bezeichner und beschreibendem Index ist kein Zwischenraum.

Variable Enthält einen Ausdruck, der unter einem Namen (Bezeichner) angesprochen werden kann.

Zahlen Ganze Zahlen bestehen aus einer Folge von Ziffern. Dezimalbrüche bestehen aus einer Folge von Ziffern und dem Dezimaltrennzeichen (in der deutschen Standardeinstellung das Komma). Fließkommazahlen werden in SMath als Produkt eines Dezimalbruchs mit einer Zehnerpotenz dargestellt.

4.2. Datentypen

Es gibt skalare Größen, die einen einzigen Ausdruck als Wert haben können oder Felder, die mehrere, durch Adressindizes ansprechbare Ausdrücke aufnehmen können.

Es gibt drei Typen von Feldern:

- **Matrizen**, das sind zweidimensionale Felder, die mit einem oder zwei Indizes angesprochen werden können und die in runden Klammern angezeigt werden. Interne Darstellung: `mat()`
- **Listen**, das sind eindimensionale Felder, die mit einer geschweiften Klammer von links angezeigt werden. Interne Darstellung: `sys()`
- **Anweisungsblöcke**, das sind eindimensionale Felder deren Ausdrücke bei Aufruf zeilenweise ausgeführt werden. Sie werden links durch eine vertikale Linie begrenzt. Interne Darstellung: `line()`

Ausdrücke sind Kombinationen aus Bezeichnern, Funktionen, Operatoren, Zahlenkonstanten, Zeichenketten und Maßeinheiten.

- **Bezeichner** sind Zeichenfolgen, die nicht mit einer Ziffer beginnen und keine reservierte Operationszeichen enthalten. Sie dienen als Namen von Variablen und Funktionen.
- **Funktionen** sind Bezeichner mit nachfolgenden in Klammern gesetzten Argumenten. Die Argumente werden durch das Argumenttrennzeichen getrennt.
- **Operatoren** sind spezielle Symbole, die zugehörige Ausdrücke (die Operanden) mathematisch verknüpfen. Man kann sie als Funktionen mit speziellem Anzeigeformat auffassen.
- **Zahlenkonstanten** sind Folgen von Ziffern, die ein Dezimaltrennzeichen enthalten können. Das Vorzeichen ist in SMath nicht Bestandteil der Zahlenkonstante, sondern ein unärer Operator (Zeichen, das für eine Funktion mit genau einem Argument steht).
- **Zeichenketten** sind beliebige, in Anführungszeichen eingeschlossene Zeichenfolgen. Sie werden mit den Anführungszeichen in rotbrauner Farbe dargestellt. Funktionen für Zeichenketten werden im Kapitel „Programmierung", Abschnitt 7.4 erläutert.
- **Maßeinheiten** sind spezielle Bezeichner, die mit einem Apostroph beginnen. Das Apostroph wird in der Anzeige unterdrückt und der Bezeichner wird blau und kursiv dargestellt. Multiplikationszeichen vor Maßeinheiten werden unterdrückt.

Argumenttrennzeichen in Funktionen und Dezimaltrennzeichen in Zahlenkonstanten sind von der individuellen Programmeinstellung abhängig (**Hauptmenü> Extras> Einstellungen> Interface**).

In **Deutschland** ist das Komma als Dezimaltrennzeichen üblich. Daher kann es nicht als Argumenttrennzeichen verwendet werden, wie es in vielen Programmiersprachen üblich ist. Man weicht daher auf das Semikolon aus.

Die **englischsprachige** Standardeinstellung ist der Punkt als Dezimaltrennzeichen (Dezimalpunkt) und das Komma als Argumenttrennzeichen.

Logische (Boolesche) Werte werden durch Zahlenwerte dargestellt:

- Null für Falsch
- Ungleich Null (typischerweise Eins) für Wahr

Zehnerpotenzen werden nicht als Bestandteil der Zahlenkonstante, sondern als die entsprechende Rechenoperation dargestellt und eingegeben.

Komplexe Zahlen werden als Summe aus dem Realteil (Zahlenkonstante) und dem Imaginärteil (Produkt aus imaginärer Einheit i und einer Zahlenkonstante) dargestellt.

4.3. Arithmetik - Rechnen mit Zahlen

SMath kann die üblichen Operationen mit ganzen Zahlen ausführen.

8+7= $8+7=15$

25-16= $25-16=9$

34*49= $34\cdot 49=1666$

1000/8= $\frac{1000}{8}=125$

2^16= $2^{16}=65536$

Werden die Ergebnisse zu groß oder kommen gebrochene Werte heraus, werden sie als Fließkommazahl angezeigt, also als Produkt aus Dezimalbruch und Zehnerpotenz.

2^40= $2^{40}=1,0995\cdot 10^{12}$

10!= $10!=3,6288\cdot 10^{6}$

1/23= $\frac{1}{23}=0,0435$

log(2;10= $\log_{10}(2)=0,301$

Wann eine Zahl zu groß ist und wieviele Nachkommastellen dann angezeigt werden, kann man im Kontextmenü (rechte Maustaste) unter den Punkten „Schwelle exponentiell" und „Dezimalstellen" einstellen. Genauer wird das im Abschnitt 4.5 erläutert.

Aber auch für die Fließkomma-Darstellung kann eine Zahl zu groß werden. Die Grenze kann man durch Ausprobieren ermitteln:

$2^{1023}=8,9885\cdot 10^{307}$

$2^{1024}=$ ▮▮

Der Wert ist oberhalb der maximalen Grenze für positive Zahlen.

Das entspricht einer internen Darstellung als Double-Precision Fließkommazahl mit 64 Bit breiter Kodierung. Dabei stehen 1 Bit für das Vorzeichen, 11 Bits für den Exponenten und 52 Bits für die Mantisse zur Verfügung.

Das bedeutet, dass SMath nicht zwischen den Zahlen 2^{52} und $2^{52}+1$ unterscheiden kann, die Zahlen also nur auf 15 Dezimalstellen genau sein können.

$2^{52}=4503599627370500$

$2^{52}+1=4503599627370500$

$\log_{10}\left(2^{52}\right)=15,6536$

Ab einem Kontostand von zweistelligen Billionenbeträgen fallen also Cent-Werte unter den Tisch.

Die Begrenzungen der Zahlendarstellung gelten auch bei symbolischer Auswertung, auf die im Kontextmenü umgestellt werden kann oder die man mit Strg-. anstelle des Gleichheitszeichens anfordern kann.

40!= (numerische Auswertung)

$40!=8{,}1592\cdot 10^{47}$

40! Strg - . (symbolische Auswertung)

$40!=815915283247898000000000000000000000000000000$

Will man mit beliebig langen exakten Ganzzahlen rechnen, kann man dies mit Maxima erledigen:

Maxima(40! Strg - .

$(40!)=815915283247897734345611269596115894272000000000$

Wird nach einer Zahl ein Buchstabe eingegeben, versteht SMath das automatisch als Multiplikation und fügt einen Punkt ein. Zwischen Zahlen und Einheiten wird kein Punkt eingefügt.

2x	$2\cdot x$
3sin ⇆ 3p Strg - G	$3\cdot\sin(3\cdot\pi)$
10N ⇆	$10\,N$

4.4. Symbolisches Rechnen mit Variablen

SMath kann mit Variablen symbolisch rechnen. Das geht immer dann, wenn der Variablen kein Zahlenwert zugewiesen wurde. Die folgenden Beispiele müssen also in einem neuen Rechenblatt nachvollzogen werden.

Die folgende Definition weist der Variable $symb$ den Wert $a+b$ zu. Da aber weder a noch b vorher mit einer Zahl belegt wurden, sind $symb$, a und b symbolische Variable.

$symb:=a+b$

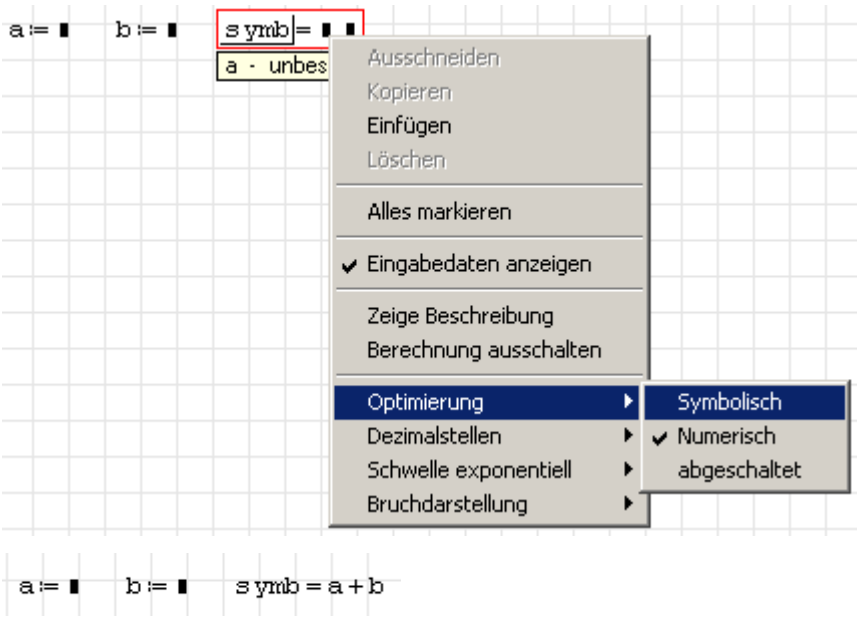
a und b sind undefiniert, während $symb$ den eben definierten Ausdruck $a+b$ zum Inhalt hat. Das Gleichheitszeichen = eignet sich nur zur Anzeige definierter Variablen. Bei undefinierten Variablen erzeugt es einen Zuweisungsoperator :=, so als hätte man : gedrückt.

$a:=\blacksquare$ $b:=\blacksquare$ $symb=\blacksquare\blacksquare$

a - unbestimmt.

Standardmäßig wird bei der Anzeige versucht, einen numerischen Wert für den Ausdruck zu ermitteln. Da im vorliegenden Fall aber a und b undefiniert sind, kann auch $a+b$ nicht berechnet werden. Die Fehlermeldung weist auf das erste Problem hin, über das die Berechnung stolpert.

Anstelle des Werts kann aber auch der symbolische Ausdruck, mit dem die Variable definiert wurde, angezeigt werden. Dies geschieht, indem man die Anzeige im Kontextmenü (rechte Maustaste) auf „Optimierung symbolisch" umstellt:



Der symbolische Auswertungsoperator $\rightarrow$, den Sie im Arithmetikpanel der Seitenleiste finden, zeigt sofort das symbolische Ergebnis an (Sie können auch das Tastenkürzel Strg - . verwenden).

$$a = a$$

$$b = b$$

$$symb = a + b$$

In früheren SMath-Versionen wurde dieser Operator mit einem Pfeil (→) dargestellt, so dass man ein symbolisches von einem numerisch ausgewerteten Ergebnis unterscheiden konnte. Gegenwärtig wird in beiden Fällen das Gleichheitszeichen benutzt. Das ist besonders lästig, weil der symbolische Auswertungsoperator die Kontexmenü-Einstellung „Numerisch" ignoriert. Von seiner Benutzung ist daher abzuraten. Statt dessen sollte immer das normale Gleichheitszeichen mit der passenden Einstellung im Kontextmenü gewählt werden.

Werden nun a oder b definiert, so wird dies bei der nächsten Anzeige von $symb$ berücksichtigt. Zunächst die Definition

$$a := xx \qquad b := yy$$

und nun die Anzeige (symbolisch, denn Zahlenwerte bekommt man natürlich nach wie vor nicht):

$$a = xx \qquad b = yy \qquad symb = xx + yy$$

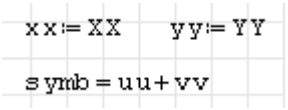
Der für $symb$ gespeicherte Ausdruck lautet nach wie vor $a + b$. Allerdings wird bei der Auswertung alles eingesetzt, was SMath über a und b weiß, auch wenn diese Information erst nach der Definition von $symb$ hinzugefügt wurde. Entscheidend ist, was zur Auswertungszeit bekannt ist.

Werden a und b geändert, so ändert sich auch das symbolische Auswertungsergebnis von $symb$:

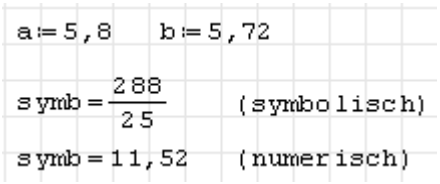
$$a := uu \qquad b := vv$$

$$symb = uu + vv$$

Die Variablen xx und yy sind nun für den Ausdruck $symb = a + b$ irrelevant, da sie nicht mehr mit a und b zusammenhängen. Daher wirken sich Änderungen an xx und yy nicht mehr aus:



Wenn Zahlenwerte bekannt sind, dann werden diese bei symbolischer Auswertung als Quotient und bei numerischer Auswertung als Gleitkommazahl angezeigt:



Symbolische Optimierung bei Definitionen

Die Optimierungseinstellung (Kontextmenü „Optimierung" mit den Werten „Symbolisch", „Numerisch" oder „Abgeschaltet") hat auch bei Definitionen Einfluss.

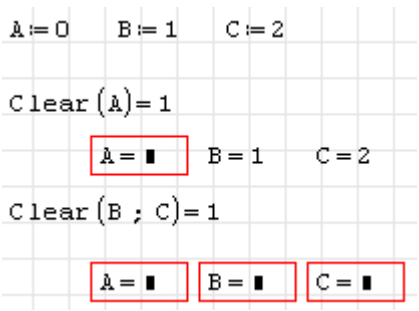
Symbolisch (Standardeinstellung bei Definitionen) Der Ausdruck auf der rechten Seite wird symbolisch vereinfacht, allerdings ohne existierende Variablenbelegungen außerhalb der Definition einzusetzen. Es gibt Fälle, in denen die symbolische Optimierung Fehler in die Definition bringt. Beispielsweise werden Vereinfachungsregeln für die Multiplikation von Zahlen angewendet, die nicht gültig sind, wenn die Variablen vektor- oder matrixwertig sind. Zur Definitionszeit ist dies normalerweise nicht bekannt.

Numerisch Das Ergebnis des Ausdrucks auf der rechten Seite wird berechnet. Wenn dies nicht gelingt, weil einer der beteiligten Werte nicht definiert ist, dann wird der Ausdruck unverändert als Definition gespeichert.

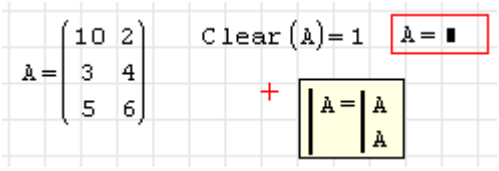
Abgeschaltet Der Ausdruck auf der rechten Seite wird unverändert gespeichert. Dies müsste angesichts der unsicheren symbolischen Vereinfachung die Voreinstellung sein.

4.4.1. Variablen löschen

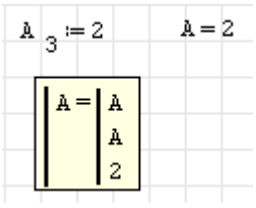
SMath kennt standardmäßig keine Möglichkeit, eine einmal definierte Variable wieder zu löschen, z.B. um sie weiter unten im Rechenblatt neu verwenden zu können. Das Plugin *Custom Functions* (Seite 214) enthält die Funktion `Clear()`, (Seite 267), mit der man Variablen löschen kann:



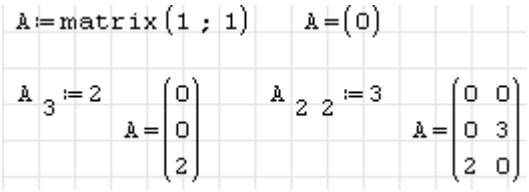
Beachten Sie, dass die Funktion ausgewertet werden muss, damit sie tatsächlich ausgeführt wird. Die Löschung ist nicht ganz vollständig, denn die Variable enthält einen Anweisungsblock mit sich selbst als Rückgabewert (man sieht das, wenn man die Maus über das Argument `A` in `Clear(A)` stellt:



Sie kann daher nicht wie jungfräuliche Variablen elementweise als Matrix definiert werden.



Vorher muss man die Variable explizit als Matrix deklarieren, in dem man ihr eine solche als Wert zuweist. Dann kann man diese Matrix wie gewohnt durch Elementzuweisung dymanisch erweitern.



4.4.2. Vorsicht mit symbolischer Auswertung!

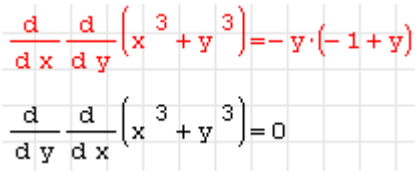
An dieser Stelle werden Beispiele für Fehler in der symbolischen Auswertung angeführt. In der Regel sind diese Probleme durch numerische Auswertung vermeidbar. Sie illustrieren die Überlebensregel

> „Äußerste Vorsicht bei symbolischer Auswertung! Im Zweifelsfall verzichten Sie auf die Benutzung dieses Leistungsmerkmals."

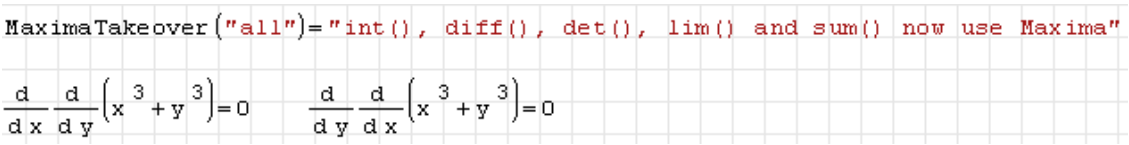
Die jeweils falschen Ergebnisse sind rot gekennzeichnet.

Beispiel 1

Bei der Bildung **gemischter zweiter Ableitungen** ist das Ergebnis von der Reihenfolge abhängig und kann daher falsch sein (Forum-Beitrag):



Dieses Problem wird durch Verwendung des Maxima-Plugins vermieden:



Beispiel 2

Unzulässige Vereinfachung von Ausdrücken, wenn die Variablen vektorwertig sind. Die folgenden Vereinfachungen sind für skalare Zahlen korrekt:

$$(a\cdot b)\cdot c = a\cdot b\cdot c \qquad a\cdot(b\cdot c) = a\cdot b\cdot c \qquad b\cdot(a\cdot c) = b\cdot a\cdot c$$

Sind die Variablen allerdings vektorwertig (was SMath hier bekannt ist), dann werden trotzdem bei symbolischer Auswertung die gleichen Vereinfachungen vorgenommen, obwohl sie nicht mehr zulässig sind:

$$a := \begin{pmatrix}1\\1\\1\end{pmatrix} \quad b := \begin{pmatrix}3\\2\\1\end{pmatrix} \quad c := \begin{pmatrix}1\\2\\3\end{pmatrix}$$

$$a\cdot b = 6 \qquad a\cdot c = 6 \qquad b\cdot c = 10$$

Auswertung symbolisch

$$(a\cdot b)\cdot c = \begin{pmatrix}6\\12\\18\end{pmatrix} \qquad a\cdot(b\cdot c) = \begin{pmatrix}6\\12\\18\end{pmatrix} \qquad b\cdot(a\cdot c) = \begin{pmatrix}6\\12\\18\end{pmatrix}$$

Dieses Beispiel wird auch noch einmal im Abschnit 4.9.3 im Kontext der Matrizenoperationen besprochen.

Beispiel 3

Falsche Auswertung von benutzerdefinierten Funktionen, die Vektoroperationen enthalten. Die folgende Funktion kann als Argumente entweder Skalare oder 2×2-Matrizen annehmen:

$$\mathrm{f}(x) = \left(x^{-1} - 1\right)\begin{bmatrix}1\\2\end{bmatrix}$$

Bei Matrizen wird die 1 elementweise abgezogen. Korrekte Ergebnisse sind beispielsweise:

$$\mathrm{f}(1) = \left(\frac{1}{1} - 1\right)\begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}, \quad \mathrm{f}\left(\begin{bmatrix}1&4\\2&5\end{bmatrix}\right) = \left(\begin{bmatrix}1&4\\2&5\end{bmatrix}^{-1} - \begin{bmatrix}1&1\\1&1\end{bmatrix}\right)\begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}-2\\-3\end{bmatrix}.$$

Das Ergebnis in SMath hängt von der Optimierungseinstellung sowohl bei der Funktionsdefinition als auch beim Funktionsaufruf (Auswertung) ab. Falsche Ergebnisse sind rot gekennzeichnet:

sym $\quad \mathrm{f}(x) := \left(\frac{1}{x} - 1\right)\cdot\begin{pmatrix}1\\2\end{pmatrix}$

num $\quad \mathrm{f}(1) = \begin{pmatrix}0\\0\end{pmatrix}$

symb $\quad \mathrm{f}(1) = 0$

num $\quad \mathrm{f}\left(\begin{pmatrix}1&4\\2&5\end{pmatrix}\right) = \begin{pmatrix}0\\-1\end{pmatrix}$

symb $\quad \mathrm{f}\left(\begin{pmatrix}1&4\\2&5\end{pmatrix}\right) = \begin{pmatrix}0\\-1\end{pmatrix}$

num $\quad \mathrm{f}(x) := \left(\frac{1}{x} - 1\right)\cdot\begin{pmatrix}1\\2\end{pmatrix}$

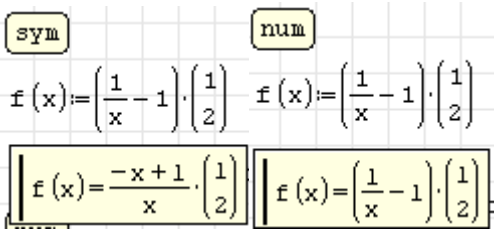
num $\quad \mathrm{f}(1) = \begin{pmatrix}0\\0\end{pmatrix}$

symb $\quad \mathrm{f}(1) = 0$

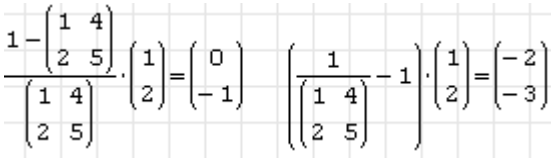
num $\quad \mathrm{f}\left(\begin{pmatrix}1&4\\2&5\end{pmatrix}\right) = \begin{pmatrix}-2\\-3\end{pmatrix}$

symb $\quad \mathrm{f}\left(\begin{pmatrix}1&4\\2&5\end{pmatrix}\right) = \begin{pmatrix}-2\\-3\end{pmatrix}$

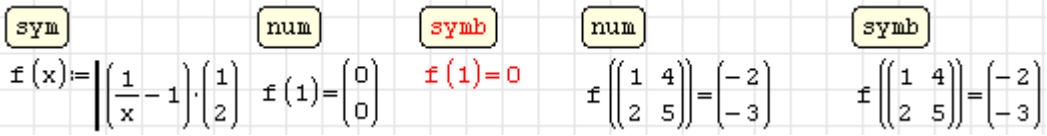
Auch hier liegt die Ursache in symbolischen Umformungen der Definition, die nur für skalare Argumente zulässig sind. Dies erkennt man in der dynamischen Auswahl (Mauszeiger auf der Definition verweilen lassen).



Der symbolisch umgeformte Ausdruck wird dann völlig korrekt weiterverarbeitet, nur eben war die Umformung für Matrixargumente nicht gültig:



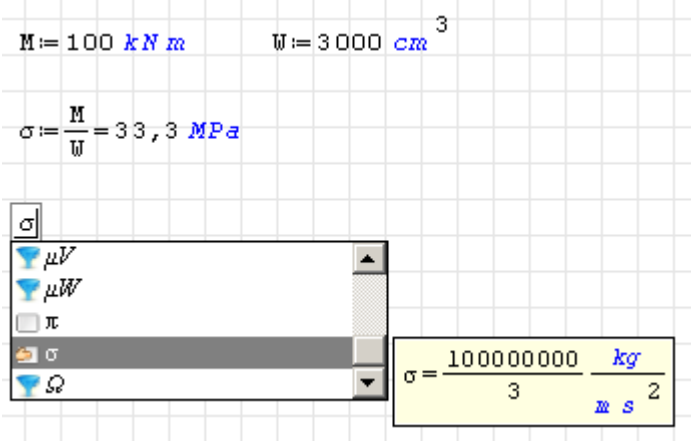
Die `line()`-Funktion (eigentlich zur Bildung von Anweisungsblöcken gedacht) unterdrückt die symbolische Vereinfachung bei der Definition.



4.4.3. Parametrische Rechnungen

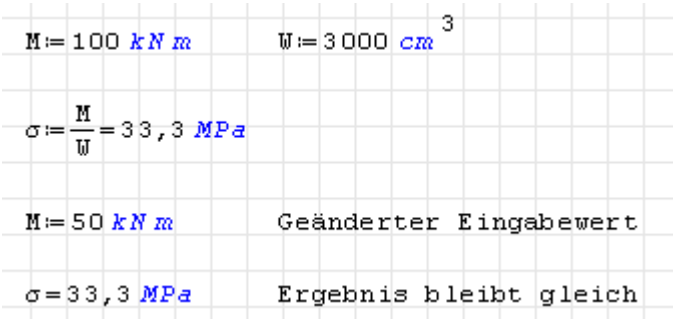
Wird eine Größe durch einen mathematischen Ausdruck definiert, wird dieser Ausdruck in vereinfachter Form gespeichert. Beispielsweise werden bereits bekannte Variablenwerte oder symbolische Definitionen eingesetzt.

Wenn Sie wie üblich, im Rechenblatt zunächst die gegebenen Größen definieren und dann die abhängigen Größen, werden in den Formeln für die abhängigen Größen nur die Zahlenwerte, nicht aber die Variablennamen gespeichert:



Wenn Sie die Ergebnisse für andere Werte benötigen, dann ändern Sie die Eingabewerte und aktualisieren Ihr Rechenblatt.

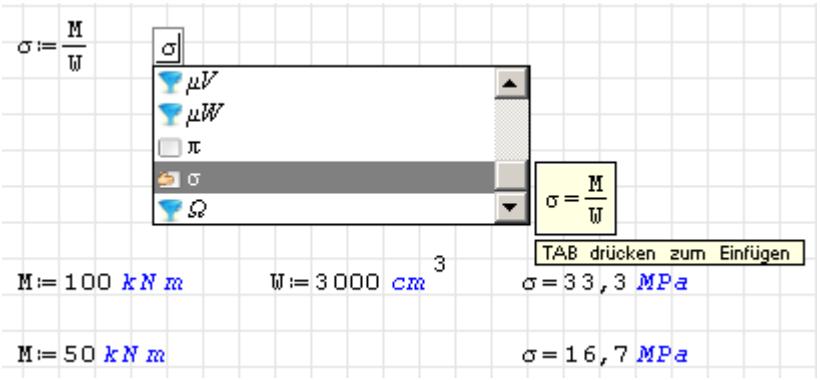
Werden die Eingabewerte weiter unten geändert, hat das keine Auswirkungen.



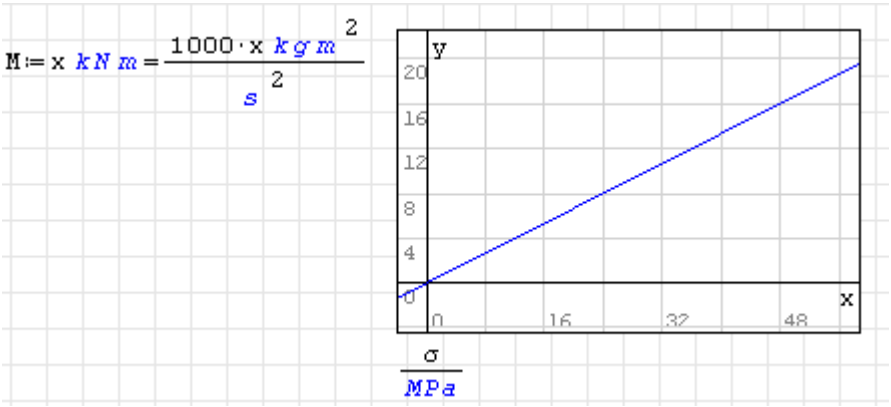
Mitunter wollen Sie weiter unten im Rechenblatt die abhängigen Größen für verschiedene Eingabedaten nebeinander stellen oder Einflüsse von Eingabedaten in Diagrammen darstellen.

Dann müssen Sie dafür sorgen, dass die Definitionen der abhängigen Größen in nicht vereinfachter Form gespeichert werden.

Dafür dürfen die Variablen für die Eingabedaten noch keinen Wert haben, wenn die abhängigen Größen definiert werden. Dann ist keine Vereinfachung möglich und der komplette Ausdruck wird gespeichert.

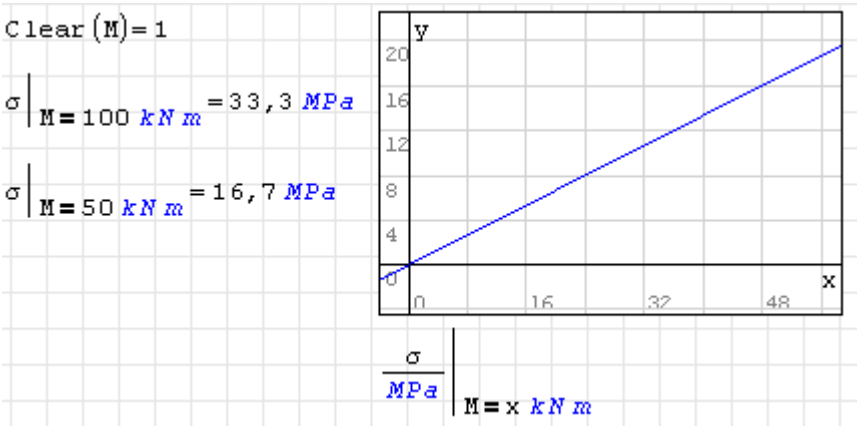


Möchte man ein Diagramm erstellen, benötigt man einen von der dimensionslosen Variable x abhängigen Ausdruck. Dazu kann einen Parameter in der Definition der abhängigen Größe durch x ersetzen.

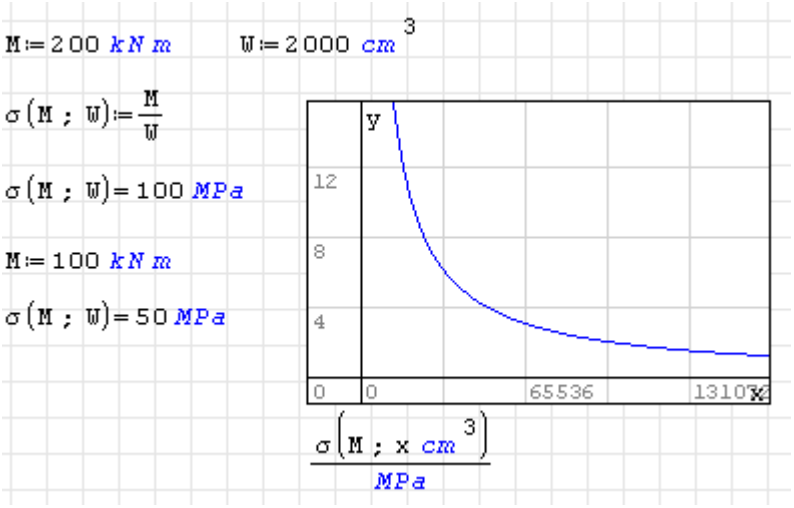


Diese Variante ist allerdings nicht sehr transparent, da dem Ausdruck im Plot nicht anzusehen ist, welche Größe eigentlich auf der x-Achse variiert wird.

Die Substitutionsfunktion `at()` kann hier helfen, sie funktioniert aber nur für Variabe, die noch keinen Wert haben. Also gegebenenfalls mit `Clear()` löschen.



Alternativ kann man den gleichen Effekt auch erreichen, indem man die abhängigen Größen als explizite Funktionen der zu variierenden Parameter definiert. Dabei ist es dann auch egal, ob die Parameter schon definiert sind.



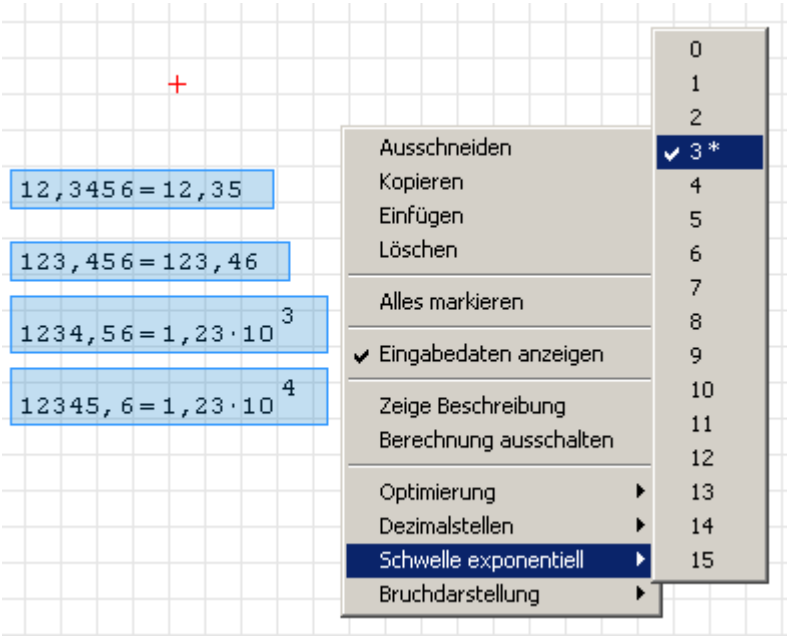
4.5. Format der Ergebnisanzeige

Die Ergebnisanzeige kann mit folgenden Einstellungen modifiziert werden.

- Zahl der Nachkommastellen (mit oder ohne nachfolgende Nullen am Ende)
- Exponentialschwelle (ab welcher Zahl von Stellen vor dem Komma auf Exponentialdarstellung umgeschaltet wird)
- Abtrennung eines vorgegebenen Faktors (Maßeinheit, Bezugsgröße)

4.5.1. Darstellung von Dezimalzahlen

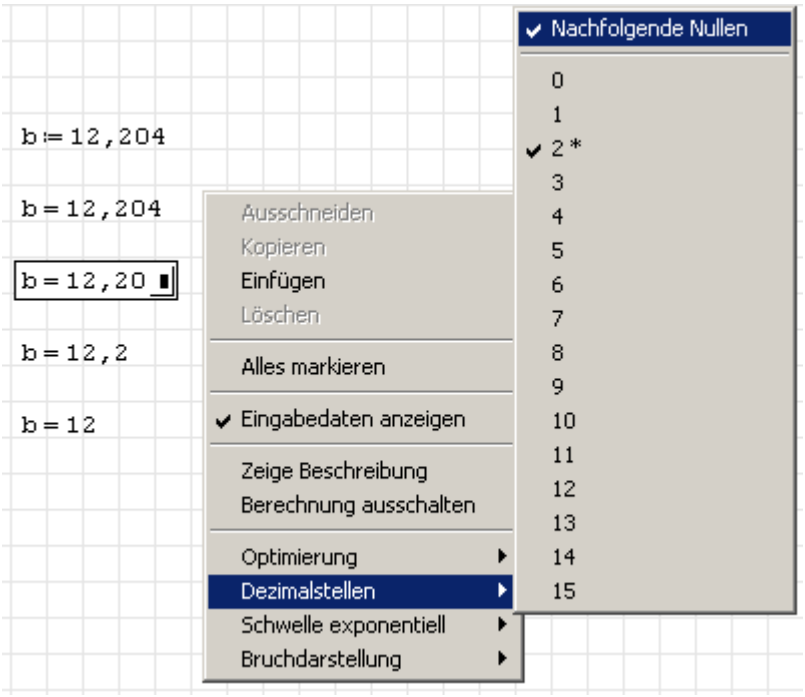
Ebenfalls im Kontextmenü kann die Exponentialschwelle eingestellt werden, also ab welcher Zehnerpotenz Zahlen in Exponentialform dargestellt werden. Im Kontextmenü sind die aktuelle Einstellung für den Formelbereich mit einem $\surd$ und die Voreinstellung mit einem * gekennzeichnet. Die Voreinstellung kann unter **Hauptmenü> Extras> Einstellungen> Berechnung** eingestellt werden.



Die Zahl der Nachkommastellen wird mit der Option „Dezimalstellen” im Kontextmenü eingestellt.

Dezimalstellen	a:=123,456	b:=12,204
3	a=123,456	b=12,204
2	a=123,46	b=12,2
1	a=123,5	b=12,2
0	a=123	b=12

Optional kann eingestellt werden, dass Nullen am Ende des Dezimalbruchs („nachfolgende Nullen”) angezeigt werden sollen:



Eine eingebaute Funktion zur Rundung auf eine bestimmte Gesamtzahl von Stellen steht leider nicht zur Verfügung. Man kann natürlich eine entsprechende Funktion schreiben:

$$\text{format}(x\_ ; n\_) := \left| \begin{array}{l} oom := \text{round}(\log10(x\_) - 0{,}5 ; 0) \\ \text{round}\left(\frac{x\_}{10^{oom}} ; n\_ - 1\right) \cdot 10^{oom} \end{array} \right.$$

$$\text{format}(123456 ; 3) = 123000$$

4.5.2. Abtrennen eines Faktors (Zehnerpotenz, Konstante, Maßeinheit)

Von jedem Ergebnis kann man einen beliebig vorgebbaren Faktor abspalten. Sobald man ein angezeigtes Ergebnis anklickt, erscheint ein Platzhalter ■, in den man den separat anzuzeigenden Faktor eingibt. Dieser Mechanismus ist für Maßeinheiten gedacht, er funktioniert aber auch mit Zahlen und Variablen.

$$123456 = 1{,}23 \cdot 10^{5}\ ■$$

10^3

$$123456 = ■ \cdot 10^{3}$$

↵

$$123456 = 123{,}46 \cdot 10^{3}$$

Im folgenden Beispiel wird der Wert von `arcsin(1)` einmal unmodifiziert, dann als Vielfaches von π und dann in der Maßeinheit ° angezeigt:

arcsi ⇆ 1= ↵

arcsi ⇆ 1= → → p Strg-G ↵

arcsi ⇆ 1= → → ° ⇆ ↵

$$\arcsin(1) = 1{,}571$$

$$\arcsin(1) = 0{,}5\,\pi$$

$$\arcsin(1) = 90\,°$$

4.5.3. Prozentrechnung

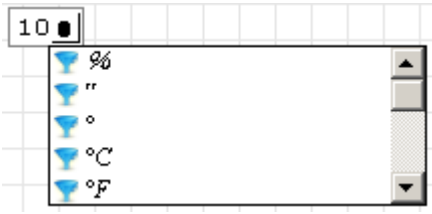
Für Prozentrechnung gibt es eine Maßeinheit %[1]. Diese kann nur über die dynamische Hilfe oder über den Dialog „Einheit einfügen" eingegeben werden, da das eigentliche Prozentzeichen in Formelbereichen nicht eingebbar ist.

Um etwa z.B. 10% einzugeben, tippen Sie zunächst

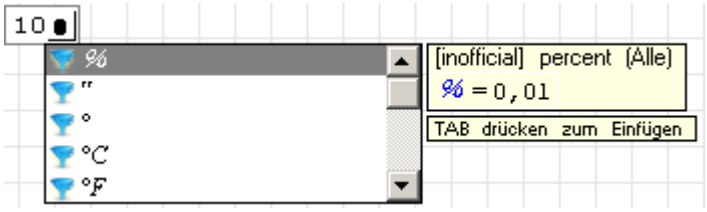
```
10'
```

Es erscheint die dynamische Hilfe mit einer Liste aller Maßeinheiten, an derem Anfang das Prozentzeichen steht.

[1] Sie müssen dazu die inoffizielle portable Programmversion benutzen, dort sind zusätzliche Maßeinheiten definiert.

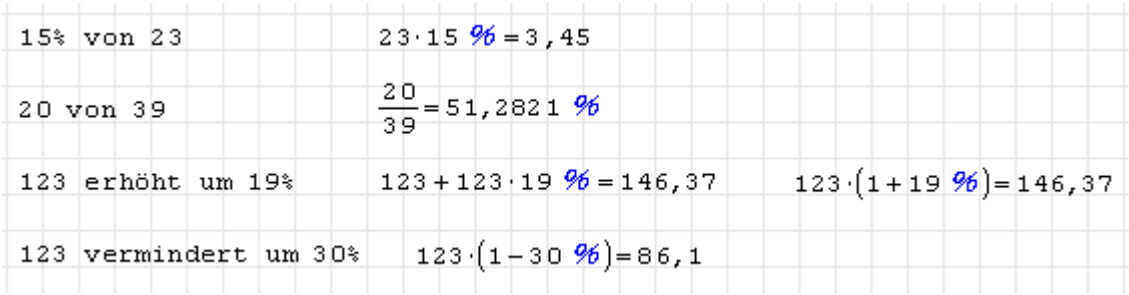


Wählen Sie das Prozentzeichen mit der Pfeiltaste ↓ aus



und bestätigen Sie mit ⇆:

Das Prozentzeichen steht für einen Faktor von 0,01 und kann damit weitgehend intuitiv verwendet werden.



Das Prozentzeichen kann alternativ auch über den Dialog „Einheit einfügen" in der Kategorie „Alle" eingefügt werden. Der Dialog wird mit **Hauptmenü> Einfügen> Einheit** oder Strg -W aufgerufen.

4.6. SMath Studio mit Maxima

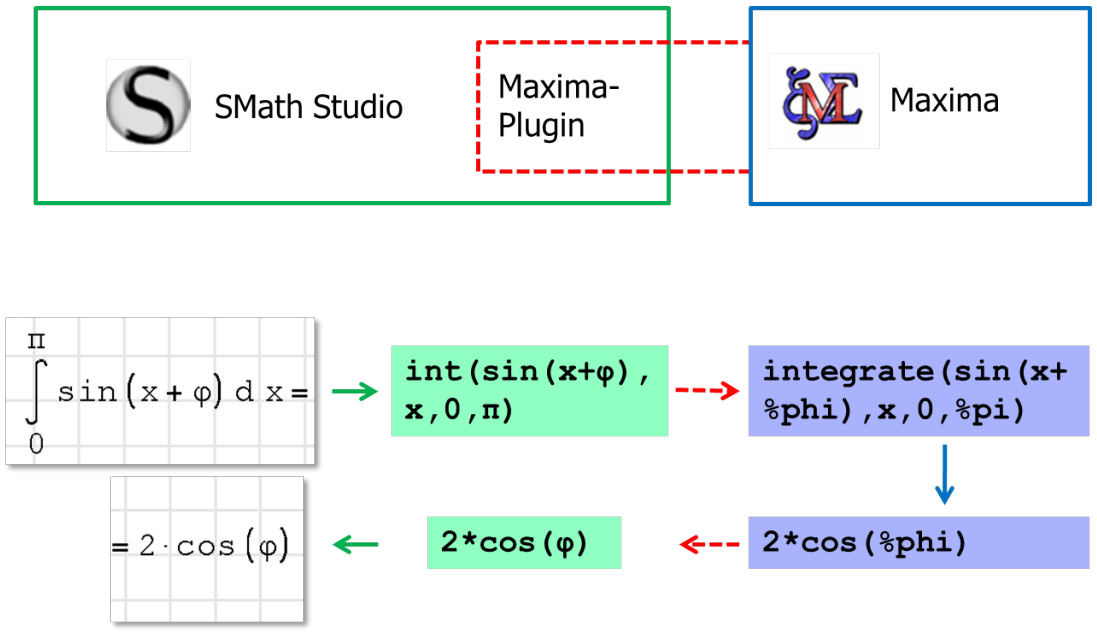


Abbildung 4.1.: Grundsätzliche Arbeitsweise der SMath-Maxima-Schnittstelle. Die Kommunikation erfolgt über einen TCP/IP-Socket.

Maxima ist ein quelloffenes und kostenloses Computeralgebra-Programm, also eine Software für das symbolische Rechnen mit Variablen. An der Fachhochschule Brandenburg wurde von Kay Graubmann und Martin Kraska ein Plugin entwickelt, welches den Zugriff auf Maxima von SMath Studio aus erlaubt.

Das Plugin und Maxima selbst sind Bestandteil der FHB-Distribution von SMath Studio, alles ist einsatzbereit vorkonfiguriert.

- Es gibt eine Funktion `Maxima()`, mit der Ausdrücke an Maxima übergeben, dort bearbeitet und an SMath zurückgegeben werden. Mit den gegenwärtig realisierten Übersetzungsfähigkeiten kann man bereits einen erheblichen Anteil des Funktionsumfangs von Maxima ansprechen.
- Das Plugin kann die internen Funktionen für Integration `int()`, Grenzwertbildung `lim()`, Ableitung `diff()`, Determinante `det()`und Summierung `sum()` übernehmen und zu Maxima umleiten. Das hat den Vorteil, dass die übersichtlichen Operatorformen dieser Funktionen weiter benutzt werden können.
- Darüber hinaus können einige Maxima-Funktionen direkt angesprochen werden, z.B. für die Lösung von algebraischen Gleichungen, z.B. die Funktion `Solve()` und von Differentialgleichungen, z.B. die Funktion `ODE.2()`.
- Diagramme können in Maxima erzeugt und mit dem Image-Bereich aus dem Plugin *ImageRegion* angezeigt werden. Da Maxima intern mit Gnuplot für die Diagrammerstellung arbeitet, stehen viele Möglichkeiten von Gmuplot nun auch innerhalb von SMath zur Verfügung.
- Maßeinheiten, Textindizes und griechische Buchstaben in Bezeichnern werden voll unterstützt.

- Als Datentypen werden Zahlen (reell und komplex), Zeichenketten, Listen und Matrizen unterstützt.

Einschränkungen:

- Die Schnittstelle funktioniert nicht unter Linux.
- Die numerischen Verfahren in Maxima kommen nicht mit Maßeinheiten zurecht.
- Logische Ausdrücke werden zwar übersetzt, aber in Maxima anders interpretiert. In Maxima gibt es den Datentyp `boolean`, der die Werte `true` oder `false` annehmen kann. In SMath entspricht dies den Zahlenwerten Null für `false` und ungleich Null für `true`.
- Es gibt Beschränkungen für die Länge von Ausdrücken, bedingt durch die Übersetzung mit Hilfe von regulären Ausdrücken in der Schnittstelle.

Internet-Ressourcen zu Maxima

- Maxima Homepage auf Sourceforge
- Maxima Overview (englisch) für den Schnelleinstieg
- Deutschsprachiges Maxima-Handbuch findet man hier..
- Edwin T. Woollett Internetseite mit Maxima-Beispielen aus der Physik

4.6.1. Die Maxima-Sitzung

Jedes laufende SMath Studio kann bei Bedarf über eine TCP-IP-Verbindung eine Maxima-Sitzung aufbauen.

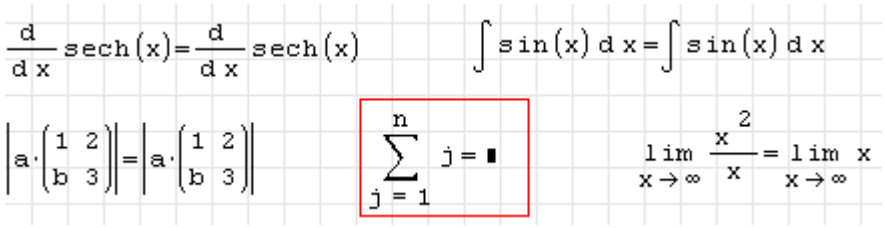
- Jede SMath-Instanz (Programmfenster) hat eine eigene Maxima-Sitzung, mit der das jeweils offene Rechenblatt kommuniziert.
- Einmal gestartet, läuft die Sitzung durch.
- Die Sitzung kann mit der Funktion `MaximaControl()` zurückgesetzt werden. Ein Neustart wird mit dem Argument ,,`restart`'' erzeugt, mit ,,`cleanup`'' werden definierte Variable und geladene Pakete wieder vergessen.

```
MaximaControl("restart")="Restart complete."

MaximaControl("cleanup")="Restart complete."
```

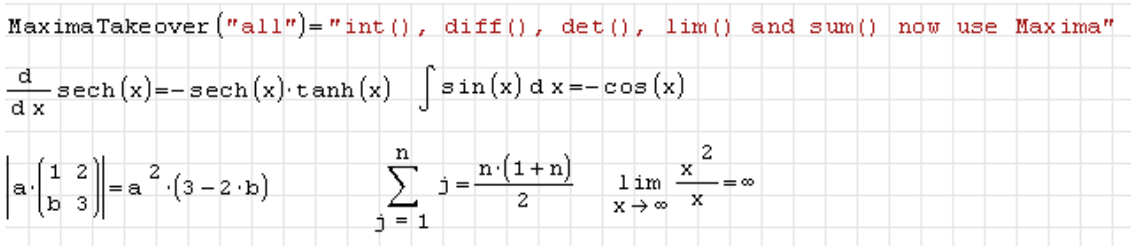
- Um unerwünschte Nebenwirkungen beim Umschalten zwischen Rechenblättern zu vermeiden, sollte man die Maxima-Sitzung am Anfang eines Rechenblatts zurücksetzen oder für jedes Rechenblatt ein neues SMath-Programmfenster öffnen.
- Es ist nicht auszuschließen, dass Maxima oder SMath abstürzen. Daher speichern Sie bitte ausreichend oft. Normalerweise schafft die Schnittstelle es, abgestürzte Sitzungen zu beenden. In Ausnahmefällen müssen Sie *maxima.exe* im Windows-Task-Manager beenden.

Da sich die Schnittstelle noch in der Entwicklung befindet, ist die Übernahme der Funktionen für Integration, Grenzwerte, Ableitungen, Determinanten und Summen standardmäßig ausgeschaltet. Sie können einzeln oder gemeinsam ab- oder angeschaltet werden. Dazu dient die Funktion `MaximaTakeover()`.

Folgende Ausdrücke kann SMath standardmäßig nicht symbolisch berechnen oder vereinfachen:



All diese Funktionen können aber durch Maxima übernommen werden, dann funktioniert es:



Diese Umschaltung kann an jeder Stelle eines Rechenblatts vorgenommen werden. Mit dem Argument `"none"` wird die Übernahme wieder komplett abgeschaltet, mit den Argumenten `"int()"`, `"diff()"`, `"lim()"`, `"det()"` und `"sum()"` können die Funktionen einzeln eingeschaltet werden.

Die symbolischen Ergebnisse können mit der Optimierungseinstellung „Symbolisch" oder „abgeschaltet" angezeigt werden. Bei „symbolisch" versucht SMath selbst noch, das Ergebnis weiter zu vereinfachen. Probieren Sie im Einzelfall aus, welche Einstellung Ihnen besser gefällt.

Die Lösung von Gleichungssystemen mit `Solve()` wird im Abschnitt 4.13.1 erläutert.

4.6.2. Die Funktion Maxima()

Diese Funktion nimmt einen Ausdruck, übersetzt ihn in die Maxima-Syntax, führt ihn in Maxima aus und übersetzt das Ergebnis zurück nach Maxima. Nicht alles, was in Maxima geht, kann man auch von SMath Studio aus machen, es ist aber schon sehr viel.

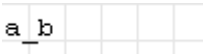
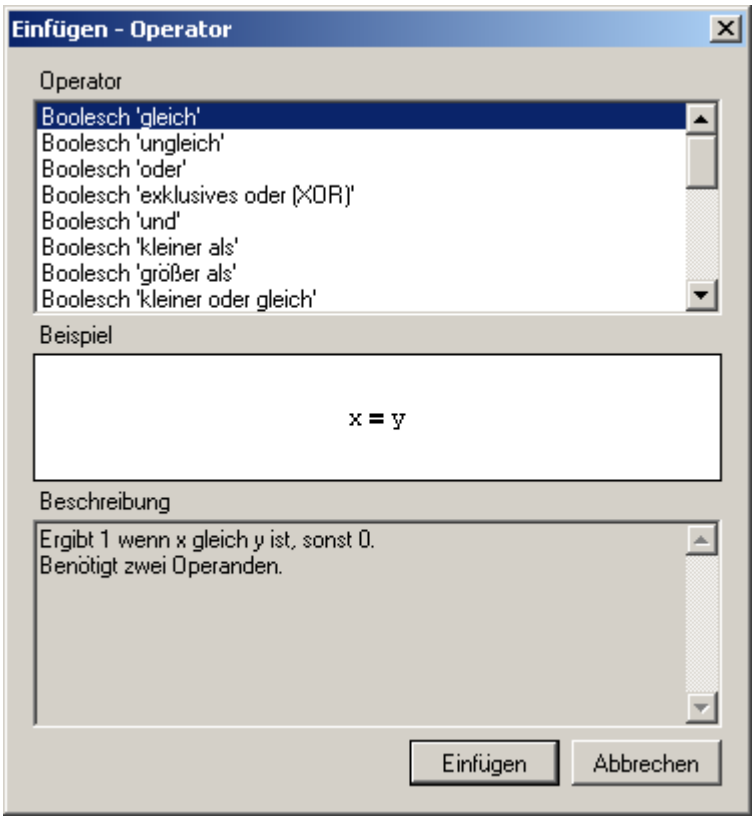
Objekt	SMath Studio	Maxima
Dezimaltrennzeichen	Komma oder Punkt	Punkt
Argumenttrennzeichen	Semikolon oder Komma	Komma
Textindex		`a_%_b`
Unterstrich in Namen		`a_b`
Griechische Buchstaben		`%alpha`
Spezielle Konstanten	e, i	`%e, %i`
Listen		`[]`
Matrizen		`matrix()`
Adressindex		`v[j]`
Zeichenketten		`"abc"`
Maßeinheiten		`%unitcm`

Tabelle 4.1.: SMath-Objekte und ihre Entsprechung in Maxima

4.7. Funktionen und Operatoren

In SMath werden als Operatoren diejenigen Funktionen bezeichnet, die auch intern durch Sonderzeichen ohne Klammern um die Argumente dargestellt werden. Bei der Eingabe von Operatoren werden die benötigten Operanden durch Platzhalter (■) dargestellt, die mit Ausdrücken ausgefüllt werden müssen.

Unter **Hauptmenü> Einfügen> Operatoren** findet man den Dialog „Einfügen-Operator”:



Zur Lesbarkeit von SMath-Dokumenten trägt die Operator-Formatierung einiger Rechenoperationen bei, die intern als Funktionen geschrieben werden, also als `Funktionname(Argument1; Argument2...)`.

Im Anhang A sind alle verfügbaren Operatoren mit ihrer Bedeutung und der Tastatureingabe aufgelistet.

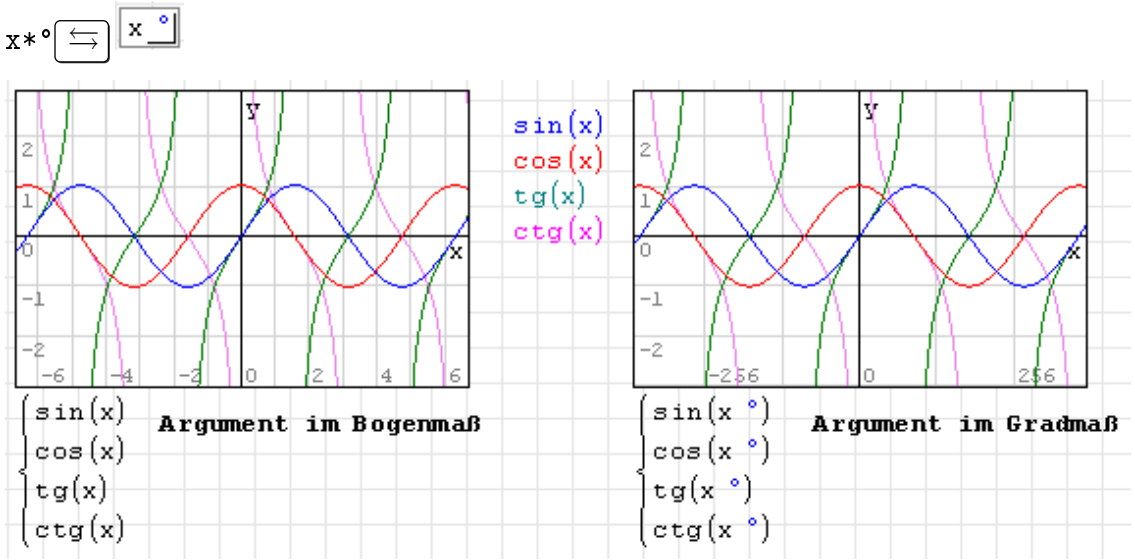
Nicht immer ist den Operatoren anzusehen, wie weit sich ihre Wirkung erstreckt. Beim Editieren kann man dies anhand der Einfügemarke (Länge des Unterstrichs) erkennen. Für Eindeutigkeit im gedruckten Dokument muss man eventuell mit Klammern sorgen.

Funktionen werden anhand ihres Namens angesprochen. Die dynamische Hilfe kennt alle definierten Funktionen und die Zahl ihrer Argumente.

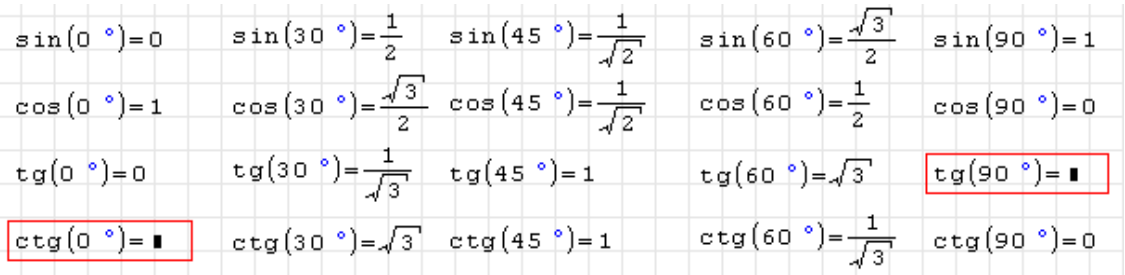
Die Definition eigener Funktionen wird im Kapitel „Programmierung” im Abschnitt 7.1 erläutert.

4.7.1. Winkelfunktionen

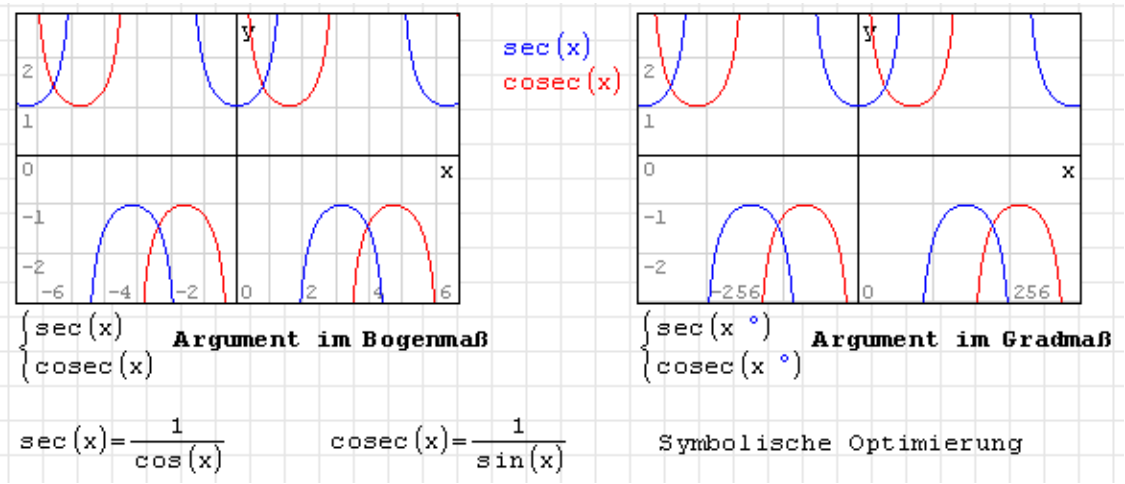
SMath kennt die elementaren trigonometrischen Funktionen. Ihre Argumente können im Bogenmaß (linkes Bild) und im Gradmaß angegeben werden (rechtes Bild). Werte im Gradmaß werden durch Multiplikation mit der Maßeinheit ° erzeugt:



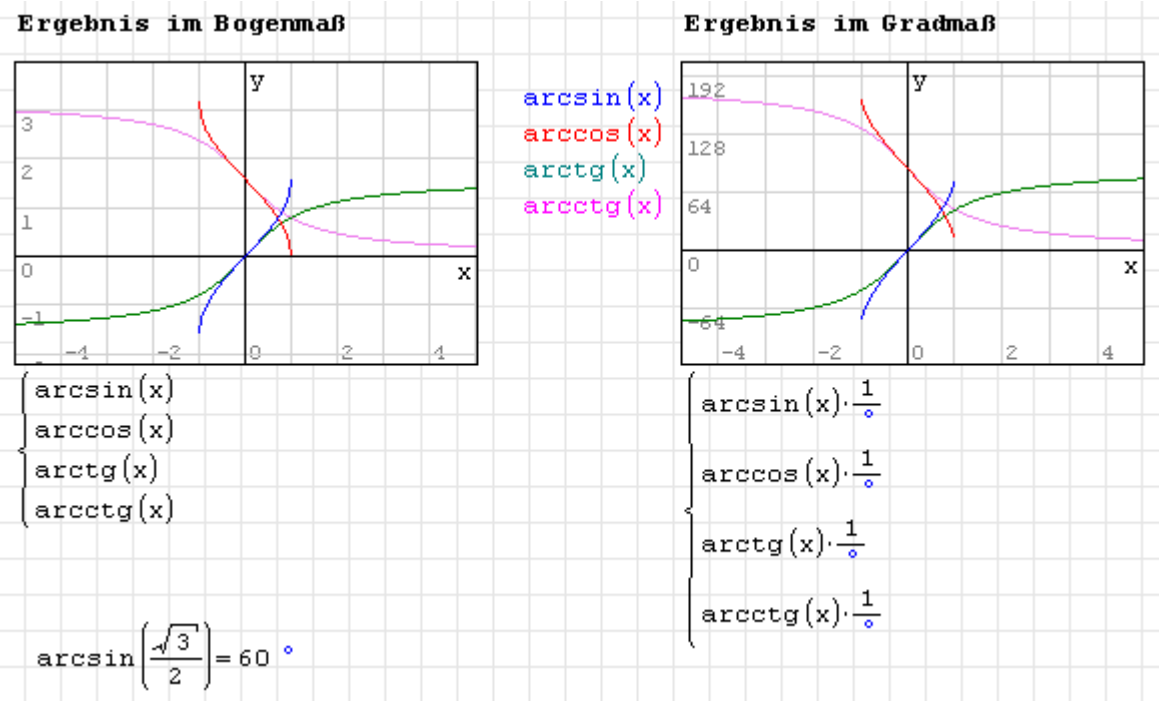
Einige Werte sind symbolisch bekannt (Anzeige mit „Optimierung symbolisch")

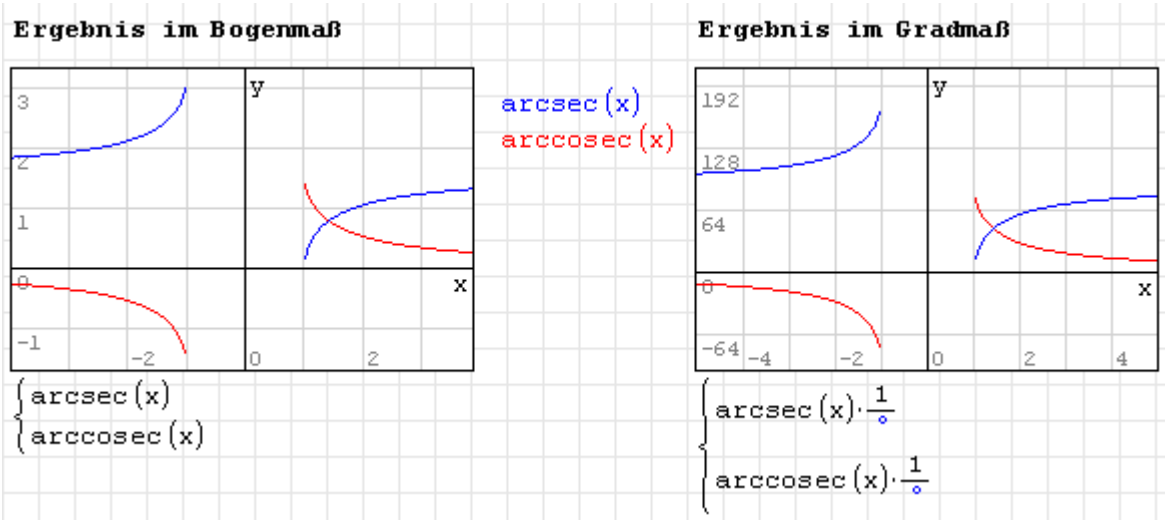


Die Sekans- und Kosekans-Funktion werden nicht direkt bereitgestellt, sondern intern auf den Sinus und Kosinus zurückgeführt.



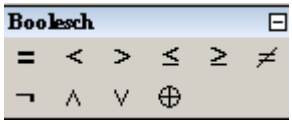
Umkehrfunktionen der Winkelfunktionen:





4.7.2. Logische und Vergleichsoperationen

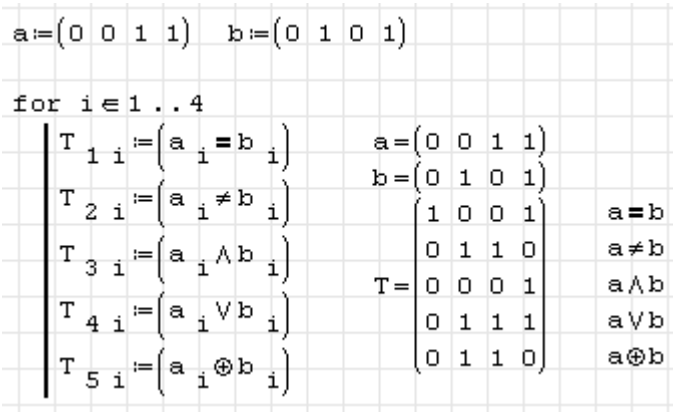
SMath verfügt über die üblichen logischen und Vergleichsoperationen. Diese liefern den Zahlenwert 1 wenn sie wahr sind und 0 wenn sie falsch sind. Die Operationen sind über das Panel „Boolesch" der Seitenleiste erreichbar, zudem gibt es einige Tastaturkürzel.



Funktion		Taste	Priorität gegenüber :=
▮=▮	Logisches Gleich	~	niedriger
▮<▮	Kleiner als	<	höher
▮>▮	Größer als	>	höher
▮≤▮	Kleiner oder gleich	Strg - 9	höher
▮≥▮	Größer oder gleich	Strg - 0	höher
▮≠▮	Ungleich	Strg - 3	niedriger
¬▮	Logische Negation		
▮∧▮	Logisches Und	&	niedriger
▮∨▮	Logisches Oder	|	niedriger
▮⊕▮	Exklusives Oder (XOR)		niedriger

Tabelle 4.3.: Binäre logische Operatoren

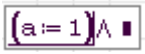
Das Verhalten der Operatoren kann man anhand einer Wahrheitstabelle prüfen:

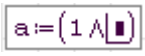


Operator	Funktion
▮≤▮≤▮	lele(#;#;#)
▮≤▮<▮	lelt(#;#;#)
▮<▮≤▮	ltle(#;#;#)
▮<▮<▮	ltlt(#;#;#)
▮≥▮≥▮	gege(#;#;#)
▮≥▮>▮	gegt(#;#;#)
▮>▮≥▮	gtge(#;#;#)
▮>▮>▮	gtgt(#;#;#)

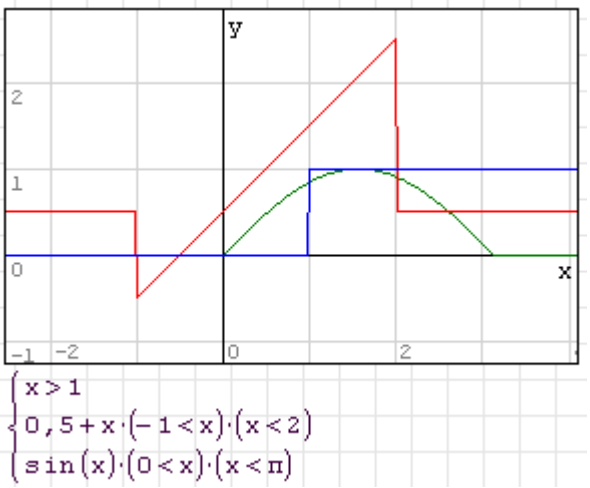
Tabelle 4.5.: Ternäre logische Operatoren aus dem Plugin *Functions Extensions*

Aufgrund der niedrigen Priorität einiger Operatoren muss die rechte Seite von Zuweisungen in solchen Fällen in Klammern gesetzt werden.

a:1& →  (erst Zuweisung, dann Und-Verknüpfung)

a:(1& →  (erst Und-Verknüpfung, dann Zuweisung)

Logische Ausdrücke können auch benutzt werden, um stückweise stetige Funktionen zu definieren:

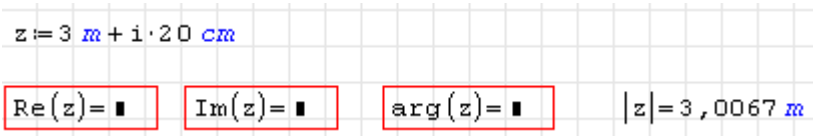


Achtung: In Maxima gibt es einen separaten boolschen Datentyp mit den Werten `true` oder `false`. Das kann bei der Übersetzung, beispielsweise beim Plotten von logischen Ausdrücken, Probleme geben.

4.8. Komplexe Zahlen

Hinweise:

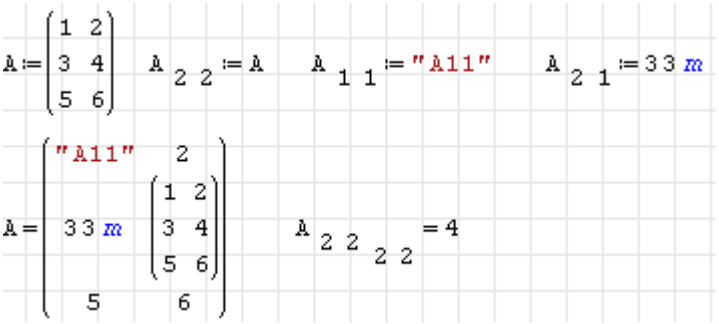
- Vermeiden Sie die Umdefinition der imaginären Einheit i (z.b. durch Verwendung als Schleifenzähler). Sie sind auf der sicheren Seite, wenn Sie statt i immer $\sqrt{-1}$ schreiben.
- Einige wichtige Funktionen wie `Re()`, `Im()`, `arg()`kommen nicht mit Maßeinheiten zurecht.



- Beachten Sie, dass manche Funktionen für komplexe Zahlen nur ordentlich arbeiten, wenn komplexe Lösungsmengen zugelassen sind. Bei Problemen erkunden Sie bitte stets die Auswirkung der Einstellung unter **Hauptmenü> Extras> Einstellungen> Berechnung**: Ergebnis (Mengen).

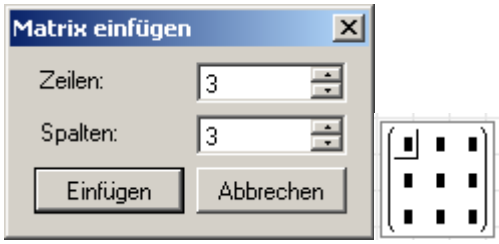
4.9. Matrizen und Vektoren

Matrizen in SMath können beliebige Ausdrücke als Elemente haben, auch andere Matrizen, Zeichenketten oder Listen.



Matrizenelemente können durch Adress-Indizes angesprochen werden, die mit der eckigen Klammer `[` erzeugt werden.

Der Dialog „Matrix einfügen" erlaubt die Angabe der Zeilen- und Spaltenzahl und erzeugt mit Einfügen ein entsprechend großes Feld von Platzhaltern



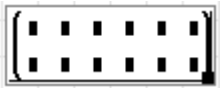
Der Dialog wird aufgerufen mit

- Strg - M
- Menü > Einfügen > Matrix
- Seitenpanel Matrizen > 

Ein 3 × 3 Platzhalter wird erzeugt durch Eintippen von `mat(`

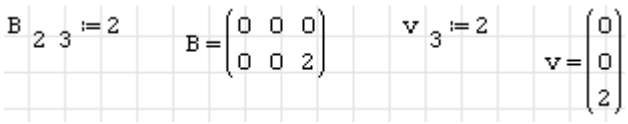
Ein 2 × 2 Platzhalter wird erzeugt durch Eintippen von `mat` ⇆

Größe verändern: linke oder rechte Klammer anklicken, dann erscheint eine Marke rechts unten, an der die Größe mit der Maus gezogen werden kann.



Die Funktion `matrix(m,n)` erzeugt eine $m \times n$-Nullmatrix.

Mit der Zuweisung eines Matrixelements wird eine Matrix der erforderlichen Größe angelegt, wenn sie noch nicht existiert, oder vergrößert, wenn sie nicht groß genug war.



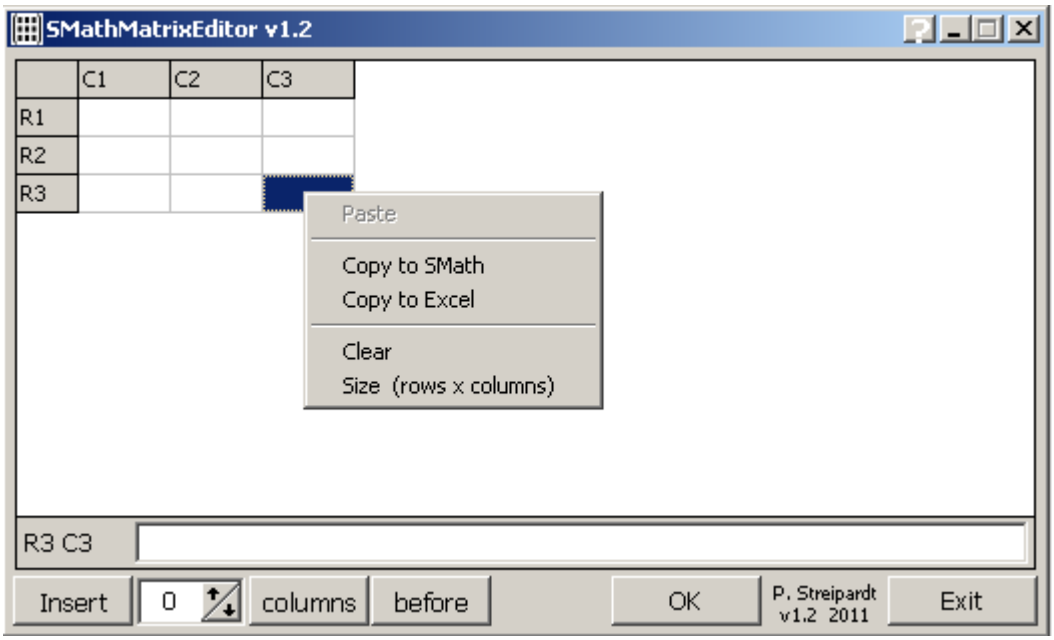
Adress-Indizes werden mit der Funktion `el()` oder mit dem Tastaturkürzel `[` erzeugt. Dabei entsteht zunächst nur ein Index, ein zweiter kann durch Eingabe eines Argumenttrennzeichens (Semikolon) hinzugefügt werden.

Matrizenelemente können durch Adress-Indizes angesprochen werden, die mit der eckigen Klammer `[` erzeugt werden.

4.9.1. Der SMathMatrixEditor

Momentan ist es innerhalb von SMath Studio nicht möglich, Zeilen oder Spalten zu Matrizen hinzuzufügen, lediglich bei Vektoren funktionieren die gleichen Methoden wie bei Listen und Anweisungsblöcken.

Um diesen Mangel auszugleichen, hat Peter Streipardt ein separates Programm geschrieben, den *SMathMatrixEditor*. Er ist Bestandteil der inoffiziellen SMath-Distribution.



Schritte für das Editieren einer Matrix:

- Kopieren der Matrix in SMath
- Einfügen im Matrixeditor
- Markieren einer Spalte (column) oder Zeile (row) durch Klick auf den jeweiligen Titel
- Auswahl der Aktion
  - Einfügen/Löschen
  - Anzahl
  - Zeilen oder Spalten
  - vor oder nach der aktuellen Zeile oder Spalten
- Ausführen mit OK
- Kopieren der Matrix (Kontextmenü> Copy to SMath)

4.9.2. Funktionen zur Erzeugung von Matrizen und Vektoren

Einheitsmatrix gegebener Dimension erzeugen

$$\text{identity}(2)=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad \text{identity}(3)=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

Diagonalmatrix aus einem Vektor

$$\text{diag}\left(\begin{pmatrix}1\\ 3\\ 5\end{pmatrix}\right)=\begin{pmatrix}1 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 5\end{pmatrix}$$

Nullmatrix gegebener Dimension erzeugen:

$$\text{matrix}(2\ ;\ 4)=\begin{pmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}$$

Die Funktion `range()` erzeugt **Vektoren mit auf- oder absteigenden Zahlenwerten**. Anzugeben sind der erste Wert und der Maximalwert. Optional kann der zweite Wert angegeben werden, der dann die Schrittweite definiert. Sie ist auf +1 oder -1 voreingestellt, je nach dem, ob der Maximalwert größer oder kleiner als der Endwert ist.

$$2\,..\,-2=\begin{pmatrix}2\\1\\0\\-1\\-2\end{pmatrix}\qquad 1\,..\,5=\begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix}\qquad 0{,}5\,..\,3{,}5=\begin{pmatrix}0{,}5\\1{,}5\\2{,}5\\3{,}5\end{pmatrix}$$

$$x_1:=0\qquad x_2:=\frac{\pi}{2}\qquad x_n:=2\cdot\pi$$

$$x_1;x_2\,..\,x_n=\begin{pmatrix}0\\0{,}5\\1\\1{,}5\\2\end{pmatrix}\pi$$

4.9.3. Rechnen mit Matrizen

Multiplikation mit einer Zahl

$$\begin{pmatrix}1&2\\3&4\end{pmatrix}\cdot 3=\begin{pmatrix}3&6\\9&12\end{pmatrix}\qquad 5\cdot\begin{pmatrix}1&2\\3&4\end{pmatrix}=\begin{pmatrix}5&10\\15&20\end{pmatrix}$$

Addition, Subtraktion

$$\begin{pmatrix}1&2\\3&4\end{pmatrix}+\begin{pmatrix}3&6\\9&12\end{pmatrix}=\begin{pmatrix}4&8\\12&16\end{pmatrix}\qquad \begin{pmatrix}1&2\\3&4\end{pmatrix}-\begin{pmatrix}3&6\\9&12\end{pmatrix}=\begin{pmatrix}-2&-4\\-6&-8\end{pmatrix}$$

Transposition (Eingabe als `transpose()` oder  aus dem Matrixpanel der Seitenleiste)

$$\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}^T=\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}$$

Matrixmultiplikation

$$\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\cdot\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}=\begin{pmatrix}5&11&17\\11&25&39\\17&39&61\end{pmatrix}\qquad \begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\cdot\begin{pmatrix}1&2\\3&4\end{pmatrix}=\begin{pmatrix}7&10\\15&22\\23&34\end{pmatrix}$$

Determinante (Eingabe als `det()` oder  aus dem Matrixpanel der Seitenleiste)

$$\left|\begin{pmatrix}1&2\\3&4\end{pmatrix}\right|=-2$$

Rang einer Matrix

$$\mathrm{rank}\left(\begin{pmatrix}1&2\\3&4\end{pmatrix}\right)=2\qquad \mathrm{rank}\left(\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\cdot\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}\right)=2$$

$$C := \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad |C| = a \cdot d - c \cdot b \qquad \mathrm{rank}(C) = 2$$

$$D := \begin{pmatrix} a & b \\ 2 \cdot a & 2 \cdot b \end{pmatrix} \qquad |D| = 0 \qquad \mathrm{rank}(D) = 1$$

SMath Studio bekommt schon bei moderat großen Matrizen Probleme mit der numerischen oder symbolischen Berechnung der Determinante.

$$A := \begin{pmatrix} 8 & 0 & 4 & 8 & 8 & 2 & 8 & 3 & 3 & 2 \\ 5 & 3 & 0 & 7 & 4 & 7 & 5 & 4 & 6 & 7 \\ 5 & 8 & 4 & 1 & 8 & 3 & 1 & 8 & 6 & 9 \\ 6 & 3 & 5 & 0 & 0 & 3 & 8 & 9 & 8 & 0 \\ 5 & 8 & 3 & 5 & 8 & 3 & 4 & 6 & 6 & 5 \\ 9 & 6 & 7 & 2 & 8 & 9 & 6 & 3 & 9 & 5 \\ 1 & 9 & 6 & 4 & 5 & 1 & 8 & 0 & 6 & 1 \\ 9 & 5 & 5 & 5 & 1 & 7 & 5 & 8 & 2 & 4 \\ 8 & 4 & 2 & 1 & 9 & 9 & 5 & 2 & 1 & 1 \\ 6 & 5 & 9 & 3 & 9 & 5 & 1 & 8 & 0 & 7 \end{pmatrix}$$

Numerische Optimierung

$\mathrm{rank}(A) =$ ▮

$|A| =$ ▮

symbolische Optimierung

$\mathrm{rank}(A) = 10$

$|A| = 1621309359234030000$

Auch skalierte (mit einem Faktor multiplizierte) Determinanten werden nicht symbolisch berechnet:

$$\left| D \cdot \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix} \right| = \left| D \cdot \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix} \right|$$

Beide Probleme werden behoben, wenn die Determinantenfunktion `det()` durch Maxima übernommen wird:

MaximaTakeover("det()") = "det() now use Maxima"

$|A| = 784894425$

$$\left| D \cdot \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix} \right| = D^3 \cdot (-b \cdot (d \cdot j - f \cdot g) + a \cdot (e \cdot j - f \cdot h) + c \cdot (d \cdot h - e \cdot g))$$

Inverse Matrix (Eingabe als `^-1`)

$$\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \right)^{-1} = \begin{pmatrix} -2 & 1 \\ 1{,}5 & -0{,}5 \end{pmatrix} \qquad \left(\begin{pmatrix} a & b \\ c & d \end{pmatrix} \right)^{-1} = \begin{pmatrix} -\frac{d}{b \cdot c - a \cdot d} & \frac{b}{b \cdot c - a \cdot d} \\ \frac{c}{b \cdot c - a \cdot d} & -\frac{a}{b \cdot c - a \cdot d} \end{pmatrix}$$

Der **Minor** ist der Wert der Unterdeterminante, die sich durch Streichung der i-ten Zeile und der j-ten Spalte ergibt. Die Eingabe erfolgt als `minor()` oder mit  aus dem Matrixpanel der Seitenleiste. Die Funktion `vminor()` liefert die entsprechende Matrix, leider ist das an der Operatordarstellung nicht zu unterscheiden.

minor() $M_{1\,2}\left(\begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{pmatrix}\right) = 8$

vminor() $M_{1\,2}\left(\begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{pmatrix}\right) = \begin{pmatrix} 11 & 39 \\ 17 & 61 \end{pmatrix} \qquad \left|\begin{pmatrix} 11 & 39 \\ 17 & 61 \end{pmatrix}\right| = 8$

Das **algebraische Komplement** ist gleich dem Minor, multipliziert mit $(-1)^{i+j}$. Eingabe mit `alg()` oder  aus dem Matrixpanel der Seitenleiste. Vorsicht, die Funktion wird genauso dargestellt wie Adressindizes an einer Matrix A

$A_{1\,2}\left(\begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{pmatrix}\right) = -8$

Spur einer Matrix (Summe der Diagonalelemente)

$\mathrm{tr}\left(\begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{pmatrix}\right) = 91$

Symbolisches Rechen mit Matrizen und Vektoren

Seien Sie extrem vorsichtig mit symbolischer Auswertung (Optimierung) bei Matrizen und Vektoroperationen. SMath vereinfacht arithmetische Ausdrücke nach den Regeln für skalare Werte (Distributivität und Kommutativität der Multiplikation). Diese gelten im Allgemeinen nicht für Vektoren und Matrizen. **Schalten Sie also im Zweifelsfall auf numerische Auswertung.**

Ein Beispiel ist die Multiplikation dreier Vektoren, bei dem mit zwei Vektoren ein Skalarprodukt gebildet wird und das Ergebnis mit dem dritten Vektor multipliziert wird. Dabei kommt es entscheidend auf die Klammersetzung an. Die Vektoren $\underline{a}$, $\underline{b}$ und $\underline{c}$ sind hier noch nicht definiert (SMath weiß überhaupt noch nicht, dass die Variablen mal Vektoren werden sollen). Daher muss symbolisch ausgewertet werden.

$(a \cdot b) \cdot c = a \cdot b \cdot c \qquad a \cdot (b \cdot c) = a \cdot b \cdot c \qquad b \cdot (a \cdot c) = b \cdot a \cdot c$

Wie man sieht, verschwinden die Klammern. Der Ausdruck wird bei der Berechnung von links nach rechts abgearbeitet. Das Produkt $\underline{a} \cdot \underline{b}$ oder $\underline{b} \cdot \underline{a}$ wird zuerst berechnet. Betrachten wir, was passiert, wenn wir den Variablen Vektoren zuweisen:

$a := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad b := \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \quad c := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

$a \cdot b = 6 \qquad a \cdot c = 6 \qquad b \cdot c = 10$

Auswertung symbolisch

$(a \cdot b) \cdot c = \begin{pmatrix} 6 \\ 12 \\ 18 \end{pmatrix} \qquad a \cdot (b \cdot c) = \begin{pmatrix} 6 \\ 12 \\ 18 \end{pmatrix} \qquad b \cdot (a \cdot c) = \begin{pmatrix} 6 \\ 12 \\ 18 \end{pmatrix}$

Wenn SMath merkt, dass beide Faktoren Vektoren sind, dann wird das Skalarprodukt berechnet. Dies ergibt eine Zahl, die dann im zweiten Schritt mit dem Vektor $\underline{c}$ multipliziert wird. Dies ist aber definitiv nicht gemeint. Die rot markierten Ausdrücke sind falsch berechnet, weil SMath die symbolischen Vereinfachungen der Ausdrücke vor dem Einsetzen der konkreten Werte vornimmt. Stellt man auf numerische Auswertung um, so erhält man die richtigen Ergebnisse:

Optimierung numerisch

$$(a\cdot b)\cdot c=\begin{pmatrix}6\\12\\18\end{pmatrix}\qquad a\cdot(b\cdot c)=\begin{pmatrix}10\\10\\10\end{pmatrix}\qquad b\cdot(a\cdot c)=\begin{pmatrix}18\\12\\6\end{pmatrix}$$

4.9.4. Manipulation von Matrizen

Abmessungen von Matrizen ermitteln (**Zeilenzahl, Spaltenzahl, Elementanzahl**):

$$A:=\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\quad \mathrm{cols}(A)=2\quad \mathrm{rows}(A)=3\quad \mathrm{length}(A)=6$$

Zugriff auf Zeilen, Spalten und Elemente:

Adressindizes werden mit der Funktion `el()` oder mit dem Tastaturkürzel `[` erzeugt. Dabei entsteht zunächst nur ein Index, ein zweiter kann durch Eingabe eines Argumenttrennzeichens (Semikolon) hinzugefügt werden.

$$\mathrm{col}(A;1)=\begin{pmatrix}1\\3\\5\end{pmatrix}\quad \mathrm{row}(A;1)=\begin{pmatrix}1&2\end{pmatrix}\quad A_{2\,2}=4$$

Matrixelemente können einzeln definiert werden:

$$A:=\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\quad A_{1\,1}:=10\quad A=\begin{pmatrix}10&2\\3&4\\5&6\end{pmatrix}$$

Überschreiten die Indexwerte die Größe der Matrix, wird diese dynamisch erweitert:

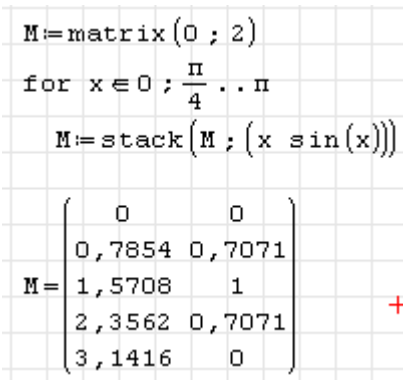
$$A=\begin{pmatrix}10&2\\3&4\\5&6\end{pmatrix}\quad A_{3\,4}:=1\quad A=\begin{pmatrix}10&2&0&0\\3&4&0&0\\5&6&0&1\end{pmatrix}$$

Lineare Adressierung: Matrixelemente können auch über einen einzelnen Index angesprochen werden, der von 1 bis zur Länge der Matrix läuft. Allerdings funktioniert dies nicht auf der linken Seite von Zuweisungen.

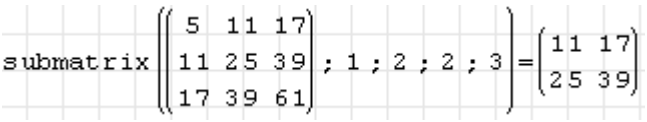
Matrizen können bei passenden Abmessungen **übereinander oder nebeneinander verbunden** werden:

$$\mathrm{augment}\left(\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix};\begin{pmatrix}1\\3\\5\end{pmatrix}\right)=\begin{pmatrix}1&2&1\\3&4&3\\5&6&5\end{pmatrix}\qquad \mathrm{stack}\left(\begin{pmatrix}1&2\\3&4\end{pmatrix};\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\right)=\begin{pmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{pmatrix}$$

Eine wichtige Anwendung ist der zeilenweise oder spaltenweise Aufbau von Matrizen in Programmschleifen. Zeilenweise aufzubauende Matrizen können dafür mit null Zeilen und der passenden Spaltenzahl initialisiert werden:

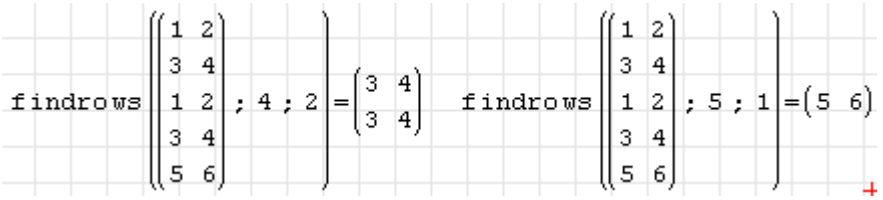


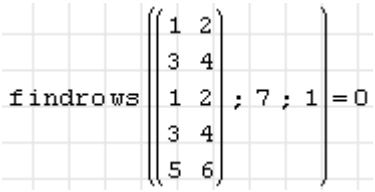
Zugriff auf **Teilmatrizen**:



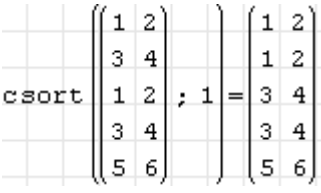
Suchen und Sortieren

Die Funktion `findrows(Matrix; Ausdruck; i)` greift aus einer Matrix alle Zeilen heraus, deren Wert in Spalte `i` gleich dem `Ausdruck` ist. Wird kein passender Eintrag gefunden, liefert die Funktion den Wert Null.

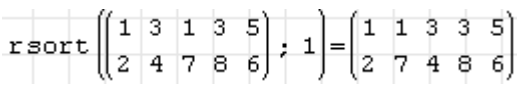




Mit `csort(Matrix; i)` werden die Zeilen einer Matrix anhand der i-ten Spalte sortiert:



Entsprechend sortiert `rsort(Matrix; i)` die Spalten einer Matrix anhand der i-ten Zeile:



Die Funktion `sort()` sortiert die Elemente einer Spaltenmatrix (eines Vektors) der Größe nach.

$$\mathrm{sort}\left(\begin{pmatrix}1\\3\\1\\3\\5\end{pmatrix}\right)=\begin{pmatrix}1\\1\\3\\3\\5\end{pmatrix}$$

`reverse()` kehrt die Reihenfolge der Zeilen in einer Matrix um:

$$\mathrm{reverse}\left(\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\right)=\begin{pmatrix}5&6\\3&4\\1&2\end{pmatrix}$$

Man kann dies benutzen, um absteigend zu sortieren:

$$\mathrm{reverse}\left(\mathrm{csort}\left(\begin{pmatrix}1&2\\3&4\\1&7\\3&8\\5&6\end{pmatrix};2\right)\right)=\begin{pmatrix}3&8\\1&7\\5&6\\3&4\\1&2\end{pmatrix}$$

Maximum und Minimum einer Matrix bestimmt man mit folgenden Funktionen:

$$\max\left(\begin{pmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{pmatrix}\right)=6 \qquad \min\left(\begin{pmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{pmatrix}\right)=1$$

Das Plugin *Custom Functions* (siehe Abschnitt 9.4.3) enthält erweiterte Versionen der Maximum- und Minimum-Funktionen, die auch mit mehreren Argumenten unterschiedlicher Formen (Listen, Matrizen, Zahlen) zurechtkommen.

$$\mathrm{Min}\left(8;\begin{pmatrix}\pi\\\frac{7}{5}\\(1\ 25\ 44)\\-10\\3\end{pmatrix};\frac{14}{6};\begin{pmatrix}\sqrt{2}&12{,}7\\\pi&e^{4}\\\begin{cases}2\\5\end{cases}&\\-4&-\frac{6}{7}\\7&\end{pmatrix}\right)=-10$$

$$\mathrm{Max}\left(8;\begin{pmatrix}\pi\\\frac{7}{5}\\(1\ 25\ 44)\\-10\\3\end{pmatrix};\frac{14}{6};\begin{pmatrix}\sqrt{2}&12{,}7\\\pi&e^{4}\\\begin{cases}2\\5\end{cases}&\\-4&-\frac{6}{7}\\7&\end{pmatrix}\right)=54{,}6$$

4.9.5. Funktionen auf Vektor- oder Matrixelemente anwenden

Matrizen kann man als Datentabellen auffassen, ähnlich wie in Excel. Will man Spalten mit Funktionen zu neuen Spalten verknüpfen, dann sind in der Regel Schleifen über alle Datenzeilen erforderlich.

Datenvektoren mit einem Wertebereich erzeugen:

Dazu dient die Funktion `range(2)`:

`x:rang` ⇆ `1` → `5` ↵

$$x := 1..5 \qquad x = \begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \qquad 2..-2 = \begin{pmatrix}2\\1\\0\\-1\\-2\end{pmatrix} \qquad 0{,}5..3{,}5 = \begin{pmatrix}0{,}5\\1{,}5\\2{,}5\\3{,}5\end{pmatrix}$$

Alternativ kann eine Schrittweite vorgegeben werden, dies leistet die Funktion `range(3)`

`s:rang` ↓ ⇆ `0` → `p` Strg-G → `p` Strg-G `/4` ↵

$$s := 0; \frac{\pi}{4}..2\cdot\pi \qquad s = \begin{pmatrix}0\\0{,}79\\1{,}57\\2{,}36\\3{,}14\\3{,}93\\4{,}71\\5{,}5\\6{,}28\end{pmatrix}$$

Berechnungen mit for-Schleifen

Typischerweise benötigt man eine Schleife, deren Zähler von 1 bis zur Länge des Vektors läuft. Diese erzeugt man so:

`for` ⇆ `i` → `rang` ⇆ `1` → `len` ⇆ `s` ↵

```
for i∈1..length(s)
  ■
```

Im Schleifenkörper weist man dann dem i-ten Element des Ergebnisvektors den gewünschten Wert zu. Dabei wird typischerweise das i-te Element eines oder mehrerer anderer Vektoren verwendet.

`y[i` Leert. `:sin` ⇆ `s[i` ↵

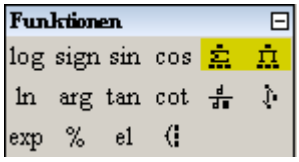
```
for i∈1..length(s)
  y_i := sin(s_i)
```

$$y = \begin{pmatrix}0\\0{,}7071\\1\\0{,}7071\\0\\-0{,}7071\\-1\\-0{,}7071\\-2{,}4492\cdot10^{-16}\end{pmatrix}$$

4.9.6. Summen und Produkte

Summen- und Produktzeichen können auf drei Wegen in das Rechenblatt eingefügt werden:

- über das Funktionen-Panel der Seitenleiste



- über das Menü **Einfügen> Funktionen> Alle> sum(4)** oder **product**
- über die Tastatur

  sum ↓ ⇆
  pr ⇆

Die Summe aller Zahlen von 1 bis 100 berechnet sich so:

sum ↓ ⇆ i → i → 1 → 100=

$$\sum_{i=1}^{100} i = 5050$$

Größen, die man in einer Summe mit ihrem Index ansprechen will, müssen als Vektor definiert werden. Achten Sie darauf, Adress-Indizes (statt Textindizes) zu verwenden. Man erzeugt sie mit der Funktion `el()` aus dem Funktionenpanel der Seitenleiste oder durch das Tastaturzeichen `[` (eckige Klammer auf).

Im folgenden Beispiel wird der Gesamtschwerpunkt dreier Flächen A_i mit den Teilschwerpunkten x_i berechnet. Die Größen werden definiert

A[1:20cm ⇆ ^2

$$A_1 := 20\ cm^2 \qquad A_2 := 30\ cm^2 \qquad A_3 := 20\ cm^2$$

$$x_1 := 2\ cm \qquad x_2 := 4\ cm \qquad x_3 := -2\ cm$$

$$A = \begin{pmatrix} 20 \\ 30 \\ 20 \end{pmatrix} cm^2 \qquad x = \begin{pmatrix} 20 \\ 40 \\ -20 \end{pmatrix} mm \qquad n := \mathrm{length}(A)$$

x.S:sum ↓ ⇆ A[j Leert. x[j → → → → j → 1 → n Leert.
/sum ⇆ A=

$$x_S := \frac{\sum_{j=1}^{n} A_j \cdot x_j}{\sum A} = 17{,}1\ mm$$

Im Nenner steht eine verkürzte Form des Summenzeichens, diese summiert über alle Elemente eines Vektors und kann über den Dialog „Funktion einfügen" oder die Tastatur eingegeben werden, dabei ist `sum(1)` auszuwählen.

4.9.7. Matrizen aus Dateien einlesen oder in Dateien schreiben

Standardmäßig steht nur die Funktion `importData()` zur Verfügung, die es erlaubt, Daten aus Textdateien zu lesen. Dabei können das Dezimaltrennzeichen und das Spaltentrennzeichen sowie Start- und End-Zeilen und -Spalten vorgegeben werden. Es können allerdings nur Zahlen oder interne SMath-Ausdrücke gelesen werden.

Die Funktion `importCell()` aus dem Plugin *Excel Collaboration* liest eine einzelne Zelle aus einer *xls* (Excel)-Datei, dies funktioniert aber nicht für Fließkommazahlen, nur für ganze Zahlen oder Zeichenketten.

Das Plugin *Data Exchange* bietet die Funktionen `ImportData.ODS()` und `ImportData.XLSX()`, die ganze Matrizen auf einmal einlesen können. *ods* ist das Dateiformat von OpenOffice Calc, *xlsx* ist das Format von MS Excel.

Das Plugin *Mathcad File Access* bietet Funktionen für das Einlesen von Pixeldaten aus Audio- und Grafikdateien sowie für das Lesen und Schreiben von Audio- und Binärdateien.

Dateinamen können mit relativem oder absoluten Pfaden versehen sein. Die Regeln für das aktuelle Verzeichnis beschreibt Abschnitt 3.17.

Funktion	Seite	Plugin	Format	Num.	Zeichenk.	Ausdr.
`importData()`	298	(Standard)	Ascii	x		x
`importCell()`	297	*Excel Collaboration*	xls	(Int.)	x	x
`importData.ODS()`	298	*Data Exchange*	ods	x	x	x
`importData.XLSX()`	298	*Data Exchange*	xlsx	x	x	x
`READ_IMAGE()`	326	*Mathcad File Access*	Bild	x		
`READ_RED()`	326	*Mathcad File Access*	Bild	x		
`READ_GREEN()`	326	*Mathcad File Access*	Bild	x		
`READ_BLUE()`	325	*Mathcad File Access*	Bild	x		
`READ_RGB()`	328	*Mathcad File Access*	Bild	x		
`READ_BMP()`	327	*Mathcad File Access*	bmp	x		
`READBIN()`	327	*Mathcad File Access*	binär	x		
`READWAV()`	328	*Mathcad File Access*	wav	x		

Tabelle 4.6.: Funktionen für den Datenimport. (Int.): nur ganze Zahlen

Ausgegeben werden können Daten mit den Funktionen

Funktion	Seite	Plugin	Format	Num.	Zeichenk.	Ausdr.
`exportCell()`	275	*Excel Collaboration*	(xls)	(Int.)	x	x
`exportData.CSV()`	275	*Data Exchange*	csv	x	x	x
`exportData.ODS()`	276	*Data Exchange*	ods	x	x	x
`exportData.XLSX()`	278	*Data Exchange*	xlsx	x	x	x
`WRITEBIN()`	343	*Mathcad File Access*	binär	x		
`WRITEWAV()`	343	*Mathcad File Access*	wav	x		

Tabelle 4.7.: Funktionen für den Datenexport. (Int.): nur ganze Zahlen, (xls): Format mit Excel nicht lesbar

4.9.8. Vektor- und Tensorrechnung in kartesischen Koordinaten

Vektoren werden durch Spaltenmatrizen ihrer kartesischen Koordinaten dargestellt. Entsprechend werden Tensoren als quadratische Koordinatenmatrizen dargestellt.

Basis, Vektorkomponenten und Vektorkoordinaten

Basisvektoren im kartesischen Koordinatensystem:

$$e_x := \begin{pmatrix}1\\0\\0\end{pmatrix} \quad e_y := \begin{pmatrix}0\\1\\0\end{pmatrix} \quad e_z := \begin{pmatrix}0\\0\\1\end{pmatrix}$$

Man kann Basisvektoren auch als Funktion definieren. Der i-te Basisvektor ist die i-te Spalte in einer Einheitsmatrix passender Dimension (hier 3):

$$e(k) := \mathrm{col}(\mathrm{identity}(3); k) \qquad e(1) = \begin{pmatrix}1\\0\\0\end{pmatrix} \quad e(2) = \begin{pmatrix}0\\1\\0\end{pmatrix} \quad e(3) = \begin{pmatrix}0\\0\\1\end{pmatrix}$$

Ein physikalischer Vektor kann dargestellt werden als Summe aus den Produkten einheitenbehafteter Koordinaten und den zugehörigen Basisvektoren. F_x, F_y, F_z seien beispielsweise die Koordianten eines Kraftvektors:

$$F_x := 1\ N \quad F_y := 3\ N \quad F_z := 2\ N$$

Dann sind die Produkte dieser Koordinaten mit den zugehörigen Basisvektoren die sogenannten Komponenten des Kraftvektors in Richtung der Koordinatenachsen. Merke: Koordinaten sind (gegebenenfalls einheitenbehaftete) Zahlen, Komponenten sind Vektoren, haben also eine Richtung.

$$F_x \cdot e_x = \begin{pmatrix}1\\0\\0\end{pmatrix} N \quad F_y \cdot e_y = \begin{pmatrix}0\\3\\0\end{pmatrix} N \quad F_z \cdot e_z = \begin{pmatrix}0\\0\\2\end{pmatrix} N$$

Die Summe der Komponenten ergibt den kompletten Kraftvektor:

$$F := F_x \cdot e_x + F_y \cdot e_y + F_z \cdot e_z \qquad F = \begin{pmatrix}1\\3\\2\end{pmatrix} N$$

Man kann auch direkt die Koordinaten in eine Spaltenmatrix schreiben:

$$\begin{pmatrix}F_x\\F_y\\F_z\end{pmatrix} = \begin{pmatrix}1\\3\\2\end{pmatrix} N$$

Mit SMath kann man die Koordinaten eines Vektors mit der Indexfunktion ansprechen:

$$F_1 = 1\ N \quad F_2 = 3\ N \quad F_3 = 2\ N$$

Skalarprodukt

Das Skalarprodukt zweier Vektoren **a** und **b** ist eine Zahl, die gleich dem Produkt aus dem Betrag |**a**|des einen, dem Betrag |**b**| des anderen und dem Kosinus des eingeschlossenen Winkels ist. In kartesischen Koordinaten ist dies gleich der Summe über die Produkte der Koordinaten beider Vektoren:

$$a := \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \quad b := \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \quad a \cdot b = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z \quad \text{(symbolische Auswertung)}$$

Die Basisvektoren kartesischer Koordinatensysteme haben die Länge 1 und stehen senkrecht aufeinander. Daher hat das Skalarprodukt eines Basisvektors mit sich selbst den Wert 1 und mit einem anderen Basisvektor den Wert 0. Solch eine Basis nennt man orthonormiert (rechtwinklig und Länge 1).

```
for i∈1..3
  for j∈1..3
    M_ij := e(i)·e(j)
```

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Betrag

Der Betrag eines Vektors ist gleich der Wurzel aus dem Skalarprodukt mit sich selbst.

$$F = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} N \quad F \cdot F = 14\,N^2 \quad \sqrt{F \cdot F} = 3{,}7417\,N \quad \text{norme}(F) = 3{,}7417\,N$$

$$\sqrt{a \cdot a} = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad \text{(symbolische Auswertung)}$$

Kreuzprodukt

Das Kreuzprodukt zweier Vektoren **a** und **b** ist gleich einem Vektor, der auf beiden Vektoren senkrecht steht und dessen Betrag gleich der von **a** und **b** aufgespannten Trapezfläche ist. Diese Fläche wiederum ist gleich dem Produkt aus den Beträgen von **a** und **b** und dem Sinus des eingeschlossenen Winkels. Das Kreuzprodukt kann sehr einfach aus den kartesischen Koordinaten der beiden Vektoren berechnet werden. In SMath steht dafür aber eine separate Funktion bereit, erreichbar über den Knopf  im Panel „Matrizen" der Seitenleiste oder Strg - 8 .

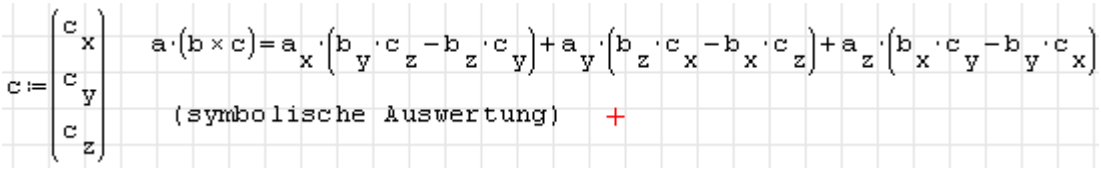
$$a \times b = \begin{pmatrix} a_y \cdot b_z - a_z \cdot b_y \\ a_z \cdot b_x - a_x \cdot b_z \\ a_x \cdot b_y - a_y \cdot b_x \end{pmatrix}$$

Die Kreuzprodukte zweier Basisvektoren liefern jeweils den senkrecht auf beiden stehenden, also den dritten Basisvektor:

$$e_x \times e_y = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad e_y \times e_z = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad e_z \times e_x = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Spatprodukt

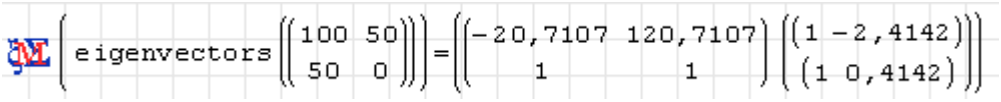
Das Spatprodukt dreier Vektoren ist das Volumen des von diesen drei Vektoren aufgespannten Parallelepipeds (Spat).

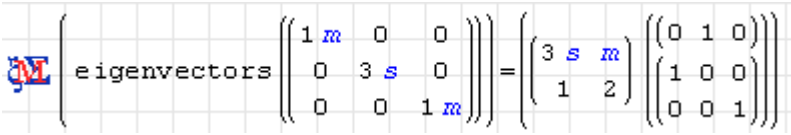


4.9.9. Eigenwerte

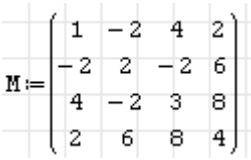
Die Maxima-Funktion `eigenvectors()` liefert Eigenwerte und Eigenvektoren quadratischer $n \times n$-Matrizen. Das Ergebnis besteht aus zwei Matrizen, die erste enthält die Eigenwerte λ_i und deren Vielfachheit m_i, die zweite die zugehörigen Eigenvektoren.

$$\begin{bmatrix} \lambda_1 & \dots & \lambda_k \\ m_1 & \dots & m_k \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \underline{\Phi}_1 \\ \vdots \\ \underline{\Phi}_{m_1} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \vdots \\ \underline{\Phi}_{m_k} \end{bmatrix} \end{bmatrix}$$

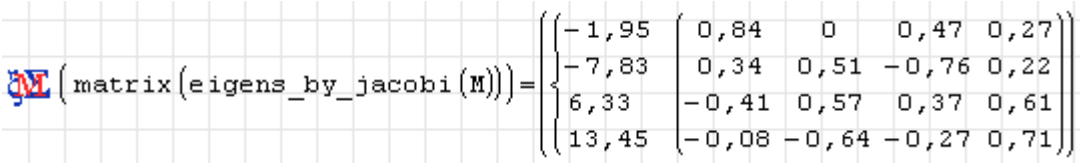




Die Funktion `eigens_by_jacobi()` berechnet die Eigenwerte und -vektoren reeller symmetrischer Matrizen numerisch.



Der Rückgabewert besteht aus einer Liste der Eigenwerte und einer Matrix der zugehörigen Eigenvektoren. Aus kosmetischen Gründen wird hier beides in eine Matrix gepackt:



Für die weitere Verwendung kann es hilfreich sein, die Eigenwertliste in einen Vektor umzuwandeln.

$$\lambda := \text{sys2mat}\left(\text{eigens\_by\_jacobi}(M)_1\right)$$

$$\Phi := \text{eigens\_by\_jacobi}(M)_2$$

$$\Phi = \begin{pmatrix} 0{,}8416 & 0{,}0037 & 0{,}4651 & 0{,}2746 \\ 0{,}3446 & 0{,}5103 & -0{,}7569 & 0{,}2188 \\ -0{,}4076 & 0{,}5693 & 0{,}374 & 0{,}6082 \\ -0{,}0823 & -0{,}6446 & -0{,}2662 & 0{,}7119 \end{pmatrix} \quad \lambda = \begin{pmatrix} -1{,}952 \\ -7{,}8257 \\ 6{,}3272 \\ 13{,}4505 \end{pmatrix}$$

Aus Eigenwerten und Eigenvektoren kann die Spektraldarstellung gebildet werden:

$$\left(\Phi \cdot \text{diag}(\lambda)\right) \cdot \Phi^T = \begin{pmatrix} 1 & -2 & 4 & 2 \\ -2 & 2 & -2 & 6 \\ 4 & -2 & 3 & 8 \\ 2 & 6 & 8 & 4 \end{pmatrix} \quad \left(\Phi^T \cdot M\right) \cdot \Phi = \begin{pmatrix} -1{,}95 & 0 & 0 & 0 \\ 0 & -7{,}83 & 0 & 0 \\ 0 & 0 & 6{,}33 & 0 \\ 0 & 0 & 0 & 13{,}45 \end{pmatrix}$$

4.9.10. Singulärwertzerlegung

Jede $m \times n$-Matrix $\underline{\underline{M}}$ kann in eine unitäre[2] $m \times m$-Matrix $\underline{\underline{U}}$, eine $m \times n$-Diagonalmatrix $\underline{\underline{\Sigma}}$ und eine unitäre $n \times n$-Matrix $\underline{\underline{V}}^T$ zerlegt werden. Die Diagonalelemente von $\underline{\underline{\Sigma}}$ sind die Singulärwerte σ_i. Die Spalten $\underline{u}_i$ von $\underline{\underline{U}}$ heißen Links-Singulärvektoren, die Spalten $\underline{v}_i$ von $\underline{\underline{V}}$ (oder Zeilen von $\underline{\underline{V}}^T$) heißen Rechts-Singulärvektoren von $\underline{\underline{M}}$.

In SMath stehen mehrere Plugins mit Funktionen zur Singulärwertzerlegung zur Verfügung. In Maxima wird die LAPACK-Funktion `dgesvd()` benutzt, die auf reelle Matrizen beschränkt ist.

Zunächst muss das Paket *lapack* geladen werden:

$$\text{msg} := \text{load}(\text{lapack})$$

Die Funktion `dgesvd()` hat drei Argumente: die zu zerlegende Matrix und zwei Schalter zur Anforderung der Matrizen $\underline{\underline{U}}$ und $\underline{\underline{V}}^T$. Der Aufruf

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 3{,}5 & 0{,}5 & 8 \\ -1 & 2 & -3 \\ 4 & 9 & 7 \end{pmatrix}$$

$$\text{SUVT} := \text{dgesvd}(M\ ;\ \text{true}\ ;\ \text{true})$$

liefert eine Liste mit den Bestandteilen

- Liste mit den Singulärwerten σ_i
- Matrix $\underline{\underline{U}}$
- Matrix $\underline{\underline{V}}^T$.

Extraktion der Matrizen:

[2] Eine unitäre Matrix hat die Determinante Eins.

$$U := SUVT_2 = \begin{pmatrix} -0,2567 & 0,0082 & 0,959 & -0,1195 \\ -0,5265 & 0,6721 & -0,2062 & -0,4781 \\ 0,108 & -0,5323 & -0,0708 & -0,8367 \\ -0,8033 & -0,5147 & -0,1809 & 0,239 \end{pmatrix}$$

$$V := SUVT_3^T = \begin{pmatrix} -0,3745 & 0,1306 & -0,918 \\ -0,5382 & -0,8368 & 0,1005 \\ -0,755 & 0,5317 & 0,3837 \end{pmatrix}$$

$$\Sigma := 0 \cdot M \quad \text{for } j \in 1..\text{Min}(\text{rows}(M); \text{cols}(M)) \quad \Sigma_{j,j} := SUVT_{1,j} \qquad \Sigma = \begin{pmatrix} 14,4744 & 0 & 0 \\ 0 & 6,3864 & 0 \\ 0 & 0 & 0,4525 \\ 0 & 0 & 0 \end{pmatrix}$$

Die Matrizen $\underline{\underline{U}}$ und $\underline{\underline{V}}$ haben wie zu erwarten die Determinante Eins:

$$|V| = 1 \qquad |U| = 1$$

und es gilt $\underline{\underline{M}} = \underline{\underline{U}}\,\underline{\underline{\Sigma}}\,\underline{\underline{V}}^T$:

$$U \cdot \Sigma \cdot V^T = \begin{pmatrix} 1 & 2 & 3 \\ 3,5 & 0,5 & 8 \\ -1 & 2 & -3 \\ 4 & 9 & 7 \end{pmatrix}$$

4.10. Listen

Listen sind eindimensionale Felder, die in SMath als Spalte mit einer geschweiften Klammer links angezeigt werden.

Listen werden in unterschiedlichen Zusammenhängen verwendet:

- Bei vielen Maxima-Funktionen treten Argumente oder Ergebnisse als Listen auf.
- Gleichzeitige Verwendung mehrerer Werte für eine Variable in Ausdrücken, Mengenobjekt.
- Darstellung von Gleichungssystemen und Unbekanntenvektoren in den Lösungsfunktionen `Solve()` und `FindRoot()`

Im Englischen werden diese Objekte auch *multiple values objects* oder *systems* genannt.

4.10.1. Listenobjekte erzeugen

Listen erzeugt man mit dem Knopf  im Seitenpanel „Funktionen” oder durch Eingabe von

`sys` ⇆

Elemente können eingefügt werden, indem ein Semikolon vor oder hinter einem Ausdruck eingegeben wird. Leere Platzhalter werden durch ⇐ (Rücktaste) gelöscht.

Die Länge von Listen kann wie bei Matrizen durch Anklicken der Klammer und Ziehen an der rechten unteren Ecke vergrößert oder verkleinert werden.

Manche Funktionen oder Rechenoperationen liefern auch Listen als Ergebnisse, so zum Beispiel der Plus/Minus-Operator (auf dem Arithmetik-Seitenpanel zu finden) und der Minus/Plus-Operator (durch Eingabe des %-Zeichens):

$$a \mp b = \begin{pmatrix} a-b \\ a+b \end{pmatrix} \qquad a \pm b = \begin{pmatrix} a+b \\ a-b \end{pmatrix}$$

Das Beispiel mit undefinierten Variablen und symbolischer Auswertung zeigt deutlich den Unterschied zwischen beiden Operatoren.

Die Listeneinträge können für Anzeige oder Definition mit Adressindizes angesprochen werden. Falls erforderlich, wird die Liste dabei verlängert. Im folgenden Beispiel wird

- die Liste *s* mit den Elementen 1 und 2,5 definiert,
- das Element Nummer 2 angezeigt,
- das Element Nummer 4 auf den Wert 3 gesetzt
- die ganze Liste angezeigt.

$$s := \begin{pmatrix} 1 \\ 2{,}5 \end{pmatrix} \qquad s_2 = 2{,}5 \qquad s_4 := 3 \qquad s = \begin{pmatrix} 1 \\ 2{,}5 \\ 0 \\ 3 \end{pmatrix}$$

Eine weitere Anzeigeoption ist die Frage, ob mehrfach auftretende Werte zusammengefasst werden. Dabei kann sich die Listenlänge ändern. Die Einstellung erfolgt in Hauptmenü> Extras> Einstellungen> Berechnung> Art des Systems. Mit „Alle Werte" werden alle erzeugten Listenelemente angezeigt:

$$1 \pm 1 \pm 1 = \begin{pmatrix} 3 \\ 1 \\ 1 \\ -1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix}$$

Mit „Wiederholte Werte nur einmal" werden Wertwiederholungen unterdrückt, so dass die Liste keine gleichen Werte mehrfach enthält:

$$1 \pm 1 \pm 1 = \begin{pmatrix} 3 \\ 1 \\ -1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Standardmäßig werden im vorliegenden Handbuch „alle Werte" angezeigt.

4.10.2. Rechnen mit Listen

Rechenoperationen können gleichzeitig auf alle Elemente einer Liste angewendet werden.

$$s := \begin{cases} 1 \\ 2,5 \end{cases} \quad s+2 = \begin{cases} 3 \\ 4,5 \end{cases} \quad s^2 = \begin{cases} 1 \\ 6,25 \end{cases} \quad \cos(s) = \begin{cases} 0,5403 \\ -0,8011 \end{cases}$$

Treten in einem Ausdruck zwei oder mehrere Listen auf, so wird der Ausdruck für alle möglichen Elementkombinationen ausgewertet:

$$s+s = \begin{cases} 2 \\ 3,5 \\ 3,5 \\ 5 \end{cases} \quad s \cdot s = \begin{cases} 1 \\ 2,5 \\ 2,5 \\ 6,25 \end{cases} \quad s^s = \begin{cases} 1 \\ 2,5 \\ 1 \\ 9,8821 \end{cases} \quad s^s + s = \begin{cases} 2 \\ 3,5 \\ 2 \\ 10,8821 \\ 3,5 \\ 5 \\ 3,5 \\ 12,3821 \end{cases}$$

Das kann man beispielsweise benutzen, um Toleranzanalysen durchzuführen. In welchen Grenzen schwankt das Volumen eines Quaders, wenn die Kantenlängen mit ihren Toleranzen bekannt sind? Zunächst werden die Abmessungen mit ihren Toleranzen definiert:

$$l := 10\ cm + \begin{cases} 0\ mm \\ 0,5\ mm \end{cases} \quad b := 5\ cm \pm 0,2\ mm \quad h := 3\ cm \cdot \begin{cases} 0,9 \\ 1,1 \end{cases} \quad \text{(entspricht } \pm 10\%\text{)}$$

$$l = \begin{cases} 100 \\ 100,5 \end{cases} mm \quad b = \begin{cases} 50,2 \\ 49,8 \end{cases} mm \quad h = \begin{cases} 27 \\ 33 \end{cases} mm$$

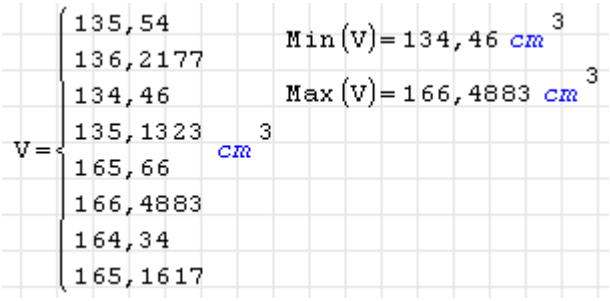
Das Volumen ist das Produkt der drei Kantenlängen. Berechnet wird das Volumen jeder möglichen Kombination von Einzelwerten.

$$V := l \cdot b \cdot h \qquad V = \begin{cases} 135,54 \\ 136,2177 \\ 134,46 \\ 135,1323 \\ 165,66 \\ 166,4883 \\ 164,34 \\ 165,1617 \end{cases} cm^3$$

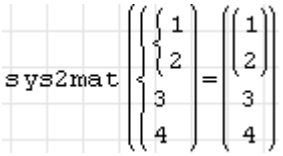
Nun sind noch die Extremwerte zu ermitteln. Dafür hat SMath keine eingebaute Funktion, denn `min()` und `max()` funktionieren nur mit Vektoren und Matrizen. Dabei hilft das Plugin *Custom Functions* (siehe Abschnitt 9.4.3).

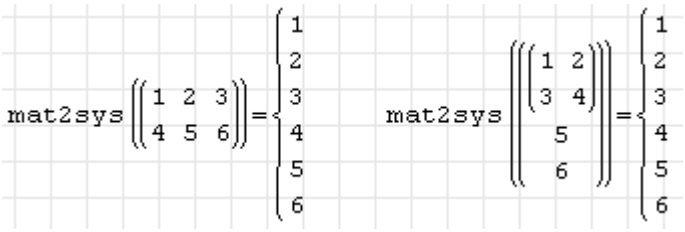
4.10.3. Listenfunktionen im Plugin Custom Functions

Das Plugin *Custom Functions* stellt unter anderem die Funktionen `Min()` und `Max()` bereit, die mit beliebig strukturierten Kombinationen aus Matrizen und Listen zurechtkommen.



Daneben bietet das Plugin auch Umwandlungsfunktionen für Matrizen in Listen und umgekehrt.

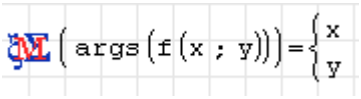




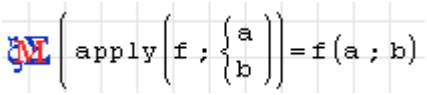
4.10.4. Maxima-Funktionen für Listen

Maxima ist in Lisp (list processing language) geschrieben, also spielen Listen im Umgang mit Maxima auch eine bedeutende Rolle. Einige Funktionen, die die Arbeit mit Listen vereinfachen, seien hier genannt.

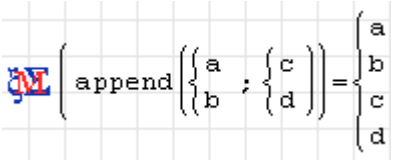
`args()` liefert die Parameter einer Funktion als Liste. Das kann nützlich sein, wenn Maxima Funktionen mit Argumenten als Ergebnis liefert und man diese Argumente extrahieren möchte:



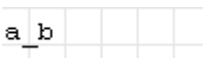
Andersrum kann man mit `apply()` die Elemente einer Liste zu Argumenten einer Funktion machen:



Mit `append()` können Listen aneinandergehängt werden:



Um eine Maxtrix in eine Liste von Listen (mit den Matrixzeilen) umzuwandeln, benutzen Sie ebenfalls die Funktion `args()`:



Das Ergebnis ist Maxima-intern eine Liste von Listen, diese wird zur Platzersparnis aber als Zeilenmatrix angezeigt.

4.11. Ableitungen

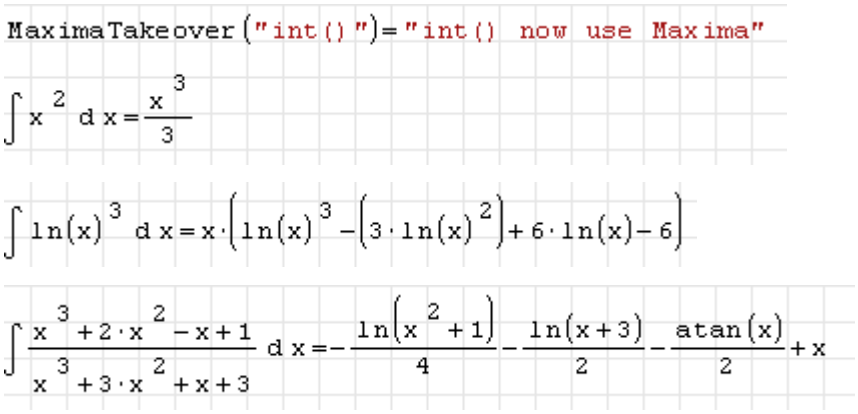
Maxima kann die Ableitungen von Ausdrücken symbolisch berechnen. Dafür ist die Funktion `diff()` zuständig, die es je nach Argumentzahl in drei Operatorformen gibt.

4.12. Integrale

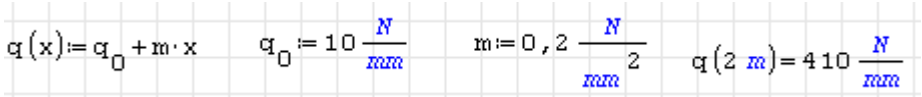
SMath Studio hat von Haus aus nur eine einfache numerische Integrationsfunktion, die nur eingeschränkt mit Maßeinheiten umgehen kann. Maxima kann die Integrationsfunktion übernehmen, dann können bestimmte und unbestimmte Integrale symbolisch gelöst werden. In späteren Versionen der FHB-Distribution kann die Übernahme durch Maxima zur Voreinstellung werden.

	Maxima symbolisch	SMath Studio numerisch
Symbolisches Ergebnis	ja	nein
Maßeinheiten in Integrand	ja	ja
Maßeinheiten in den Grenzen (in der Integrationsvariable)	ja	nein
Stückweise stetige Funktionen	nur mit `sign()`	ja, aber ohne `if`-Statement

4.12.1. Symbolische Berechnung mit Maxima



Die Integrationsfunktion kann mit Maßeinheiten umgehen. Integrand, Integrationsvariable und Integrationsgrenzen können Maßeinheiten haben.



$$x_1 := 1\,m \qquad x_2 := 3\,m$$

$$\int_{x_1}^{x_2} q(x)\,dx = 820\,kN$$

Maxima geht beim symbolischen Integrieren nicht automatisch davon aus, dass die Untergrenze kleiner als die Obergrenze ist und fragt zurück. Da der Anwender normalerweise keinen Grund hat, die Untergrenze größer als die Obergrenze zu setzen, wird diese Abfrage von der Maxima-SMath-Schnittstelle abgefangen. Dem Ergebnis wird dann die Meldung über die getroffene Annahme hinzugefügt:

$$\varepsilon(x) := \varepsilon_0 \cdot \left(\frac{x}{x_0}\right) \qquad y(x) := y_0 \cdot \left(\frac{x}{x_0}\right)^2$$

$$\int_0^{x_0} \varepsilon(x) \cdot \sqrt{1 + \left(\frac{d}{dx} y(x)\right)^2}\,dx = \begin{cases} \text{"x\_\%\_0 is assumed to be positive."} \\ \dfrac{\varepsilon_0 \cdot \left(\left(4 \cdot y_0^{\,2} + x_0^{\,2}\right)^{\frac{3}{2}} - x_0^{\,3}\right)}{12 \cdot y_0^{\,2}} \end{cases}$$

Meldung und eigentliches Ergebnis werden in einer Liste zurückgegeben. Möchte man die Frage vermeiden, muss man vorher Maxima eine Annahme mitteilen. Das macht man mit der Maxima-Funktion `assume()`.

$$\text{assume}(x_0 > 0) = \{x_0 > 0$$

$$\int_0^{x_0} \varepsilon(x) \cdot \sqrt{1 + \left(\frac{d}{dx} y(x)\right)^2}\,dx = \frac{\varepsilon_0 \cdot \left(\left(4 \cdot y_0^{\,2} + x_0^{\,2}\right)^{\frac{3}{2}} - x_0^{\,3}\right)}{12 \cdot y_0^{\,2}}$$

Macht man mehrmals die gleiche Annahme, kommt die Meldung `redundant` als Ergebnis der `assume()`-Funktion.

Möchte man eine Annahme ändern, muss man sie vorher mit `forget()` löschen, sonst kommt die Meldung `inconsistent`.

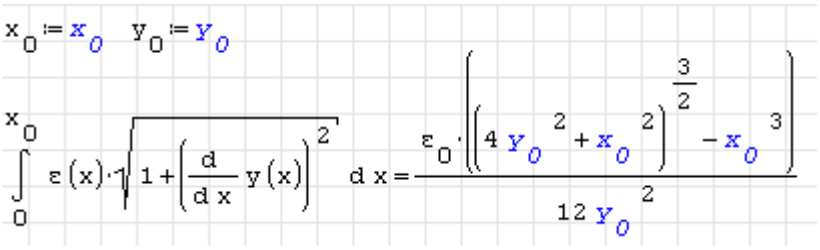
$$\text{assume}(x_0 < 0) = \{\text{inconsistent}$$

$$\text{forget}(x_0 > 0) = \{x_0 > 0$$

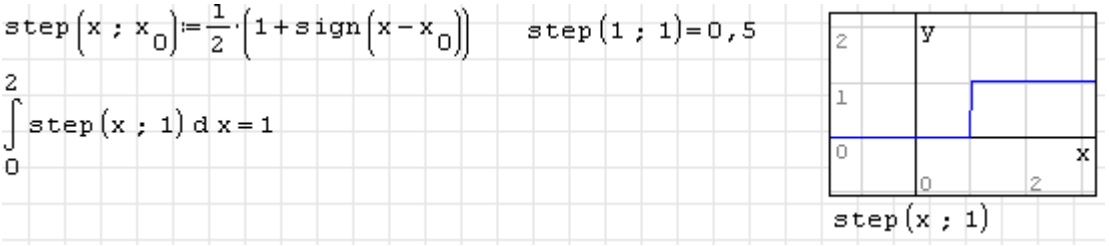
$$\text{assume}(x_0 < 0) = \{x_0 < 0$$

$$\int_0^{x_0} \varepsilon(x) \cdot \sqrt{1 + \left(\frac{d}{dx} y(x)\right)^2}\,dx = -\frac{\varepsilon_0 \cdot \left(\left(4 \cdot y_0^{\,2} + x_0^{\,2}\right)^{\frac{3}{2}} + x_0^{\,3}\right)}{12 \cdot y_0^{\,2}}$$

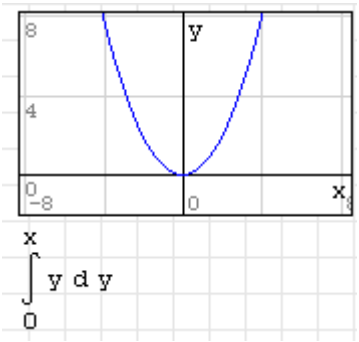
Bei Maßeinheiten weiß die Integrationsfunktion in Maxima, dass sie positiv sind. Man kann also auch eine Variable als Maßeinheit kennzeichnen (vorangestelltes Hochkomma), um Vorzeichenfragen zu vermeiden:



Stückweise stetige Funktionen müssen mit `sign()` definiert werden, da die Übersetzung boolscher Ausdrücke noch nicht implementiert ist.



Im Unterschied zur numerischen Berechnung mit SMath bereiten Integrale mit variablen Grenzen keine Probleme:



4.12.2. Numerische Berechnung mit SMath

SMath kann ohne Maxima nur bestimmte Integrale (mit Integrationsgrenzen) und diese auch nur numerisch berechnen.

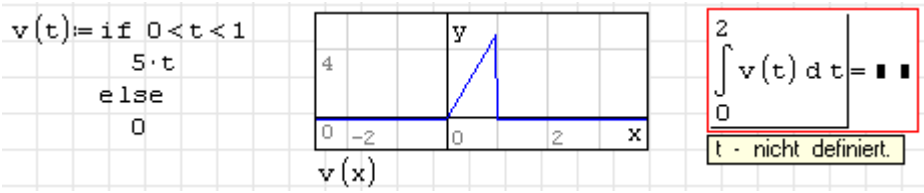
- Maßeinheiten an den Integrationsgrenzen werden ignoriert. Die Integrationsvariable wird daher als dimensionslos betrachtet.
- Funktionen dürfen keine `if()`-Anweisung enthalten. Sollen stückweise stetige Funktionen integriert werden, kann man sie mit logischen Ausdrücken oder der `cases()`-Funktion definieren.
- Die Integrationsgenauigkeit wird global unter **Hauptmenü> Extras> Einstellungen> Berechnung> Integrale: Genauigkeit** eingestellt. Der Wert gibt die Zahl der Intervalle zwischen den Integrationsgrenzen an, die bei der numerischen Berechnung verwendet wird. Im Zweifelsfall muss man den Wert erhöhen und prüfen, in welchem Maße das Ergebnis von der Einstellung abhängt. Bei hinreichend glatten Funktionen ist die Voreinstellung (100) gut geeignet, bei Funktionen mit Unstetigkeiten können schnell höhere Werte erforderlich sein.

- Im Interesse der Genauigkeit ist es vorteilhaft, das Integrationsgebiet an Unstetigkeitsstellen aufzuteilen.

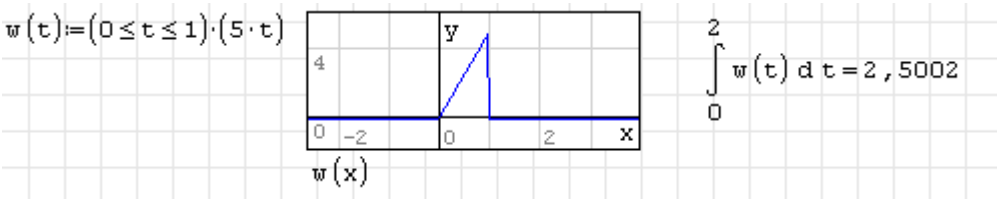
Die numerische Berechnung sollten Sie nur dann verwenden, wenn Maxima kein symbolisches Ergebnis findet.

Integrale stückweise stetiger Funktionen

Die Integrationsfunktion `int()` versagt, wenn Funktionen unter Verwendung von `if()` definiert werden. Es kommt dann die unsinnige Fehlermeldung, die Integrationsvariable sei nicht definiert.



Der Ausweg besteht in der direktien Multiplikation mit entsprechenden logischen Ausdrücken (Integrationsgenauigkeit hier auf 1000 eingestellt):



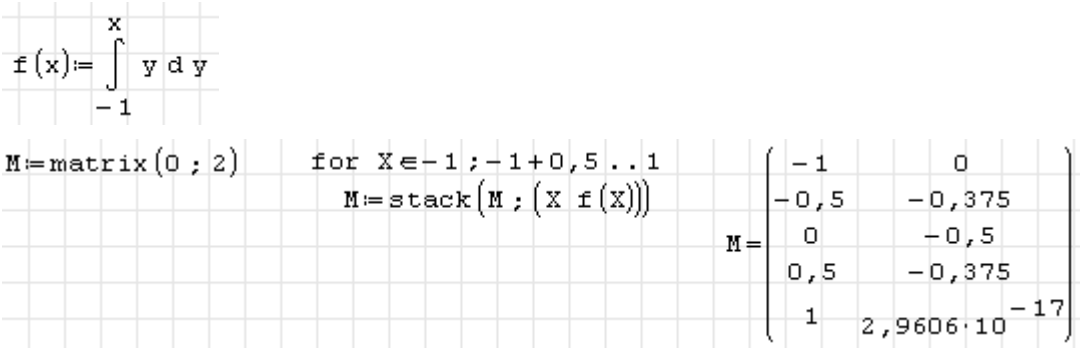
Eine Konvergenzanalyse zeigt folgende Abhängigkeit des Ergebnisses von der eingestellten Genauigkeit (**Hauptmenü> Extras> Einstellungen> Berechnung> Integrale: Genauigkeit**):

Genauigkeitseinstellung	Wert
exakt	2,5000
100	2,5167
1000	2,5017
10000	2,5002

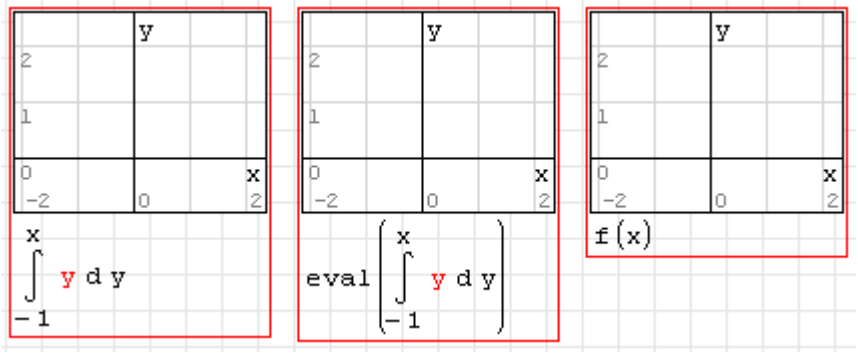
Aufgrund der Unstetigkeit bei $t = 1$ kommt es bei der Standardeinstellung (Genauigkeit 100) zu einem Fehler bereits in der dritten Mantissenstelle.

Integrale mit variablen Grenzen

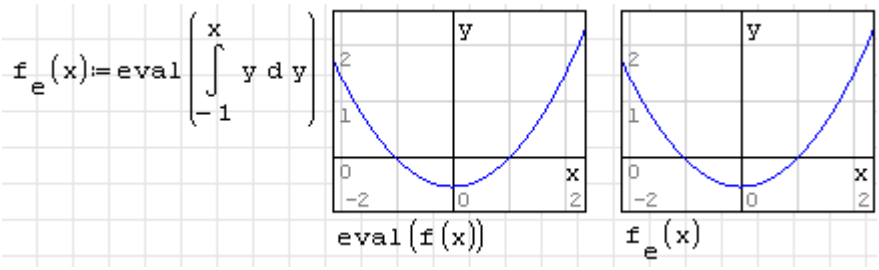
Man kann Funktionen definieren, die Integrationen mit variablen Grenzen enthalten. Achten Sie darauf, unterschiedliche Variablen für Grenzen (hier x) und Integration (hier y) zu verwenden!



Ausdrücke mit variablen Grenzen lassen sich allerdings nicht ohne weiteres plotten.

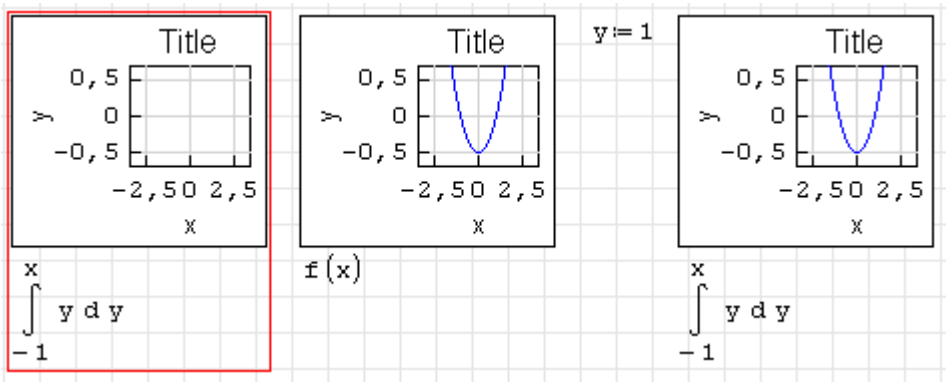


Beim Standard-Diagrammbereich gelingt der Plot erst, wenn man den Ausdruck als Funktion definiert und bei Definition oder Aufruf `eval()` anwendet.



Auch beim X-Plot Diagrammbereich (siehe Abschnitt F.3) gelingt die Darstellung nur unter bestimmten Bedingungen:

- es wird eine Funktion definiert, wobei `eval()` nicht erforderlich ist, oder
- die Integrationsvariable, die im Ausdruck verwendet wird, darf nicht undefiniert sein. Der Plotbereich identifiziert sonst zwei undefinierte Variablen im Ausdruck und geht von einer impliziten Funktionn aus.



Bei symbolischer Integration mit Maxima ist dies alles kein Thema.

Integrale über Größen mit Maßeinheiten

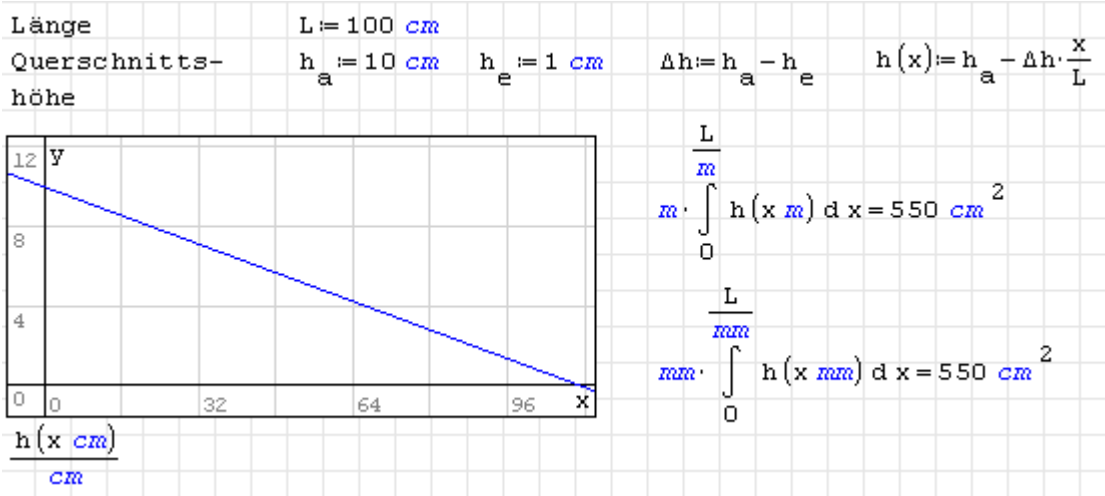
Will man beispielsweise das Integral über eine Länge berechnen, so muss man eine dimensionslose Koordinate $\xi = x/\ell$ einführen:

$$\int_0^{\ell} f(x)\mathrm{d}x = \ell \int_0^{1} f(\xi\ell)\mathrm{d}\xi$$

Als Bezugslänge ℓ kann man auch direkt eine Längeneinheit *LE* verwenden. Dann schreibt sich das Integral:

$$\int_0^{\ell} f(x)\mathrm{d}x = LE \int_0^{l/LE} f(\xi \cdot LE)\mathrm{d}\xi$$

Im folgenden Beispiel wird als Längeneinheit das Meter verwendet, das Ergebnis ist aber von dieser Wahl unabhängig, lediglich die Dimension muss passen.



4.13. Gleichungslösung

Für die Lösung von linearen und nichtlinearen Gleichungssystemen ist stets die Funktion `Solve()` aus dem Maxima-Plugin (siehe Kapitel **??**) zu empfehlen.

- Lineare und nichtlineare Gleichungssysteme werden symbolisch gelöst.
- Startwerte oder Suchbereiche werden nicht benötigt.
- Das Ergebnis wird in Form von Listen mit Gleichungen `Variable=Wert` zurückgegeben.
- Die Funktion kann mit Maßeinheiten umgehen.

In manchen Fällen werden die symbolischen Ausdrücke zu kompliziert und `Solve()` findet keine Lösung. Dann kann man auf die Funktion `FindRoot()` aus dem Plugin *Nonlinear Solvers* benutzen. Das ist ein numerischer Löser, der Startwerte für die Variablen benötigt. Er kann ebenfalls mit Maßeinheiten umgehen. Darauf wird in Abschnitt 4.13.2 eingegangen.

Hat man nur eine Gleichung für eine Variable und kennt den Bereich, in dem die Lösung liegen muss, bietet sich auch das Halbteilungsverfahren an (Funktion `Bisection()`, siehe Abschnitt 4.13.3).

SMath Studio selbst bringt die Funktionen `solve()` und `roots()` mit, die allerdings weniger leistungsfähig sind und für den Standardanwender nicht empfohlen werden. Ihre Beschreibung wurde daher in den Dachboden ausgelagert (Abschnitt 349).

4.13.1. Symbolische Lösungen mit `Solve()`

Das Gleichungssystem wird als Liste oder Vektor von Gleichungen vorgegeben, dabei wird das boolsche Gleichheitszeichen verwendet. Eine weitere Liste oder ein weiterer Vektor geben die Variablen an, nach denen aufzulösen ist.

Die Lösung kommt als Liste von Gleichungen der Form `Name=Wert`. Wenn es mehrere Lösungen gibt, werden diese als Matrix von Listen (bei mehreren Variablen) oder als Liste (bei einer Variablen) zurückgegeben.

Für die Zuweisung der Lösung auf die Variablen gibt es die Funktion `Assign()`. Sie interpretiert die Gleichungen `Name=Wert` als Zuweisungen und führt diese aus. Bei mehreren Lösungen muss man sich dabei für eine Lösung entscheiden, die man über einen Adressindex an der Ergebnismatrix auswählen kann.

Eine Gleichung, eine Unbekannte, eine Lösung

Einzelne Gleichungen oder Variablen können auch skalar (also nicht als Liste oder Matrix) angegeben werden....

$$\text{Solve}(a\cdot x+b=y,\ a)=a=\frac{y-b}{x}$$

Zwei Gleichungen, zwei Unbekannte, eine Lösung

$$L:=\left(\text{Solve}\left(\begin{pmatrix}3\cdot x+4\cdot y=7\\ 2\cdot x+a\cdot y=13\end{pmatrix},\begin{pmatrix}x\\ y\end{pmatrix}\right)\right)\qquad L=\begin{pmatrix}\left(x=\frac{-52+7\cdot a}{-8+3\cdot a}\right)\\ \left(y=\frac{25}{-8+3\cdot a}\right)\end{pmatrix}$$

$$\text{Assign}(L)=\begin{pmatrix}\frac{-52+7\cdot a}{-8+3\cdot a}\\ \frac{25}{-8+3\cdot a}\end{pmatrix}\qquad x=\frac{-52+7\cdot a}{-8+3\cdot a}\qquad y=\frac{25}{-8+3\cdot a}$$

Eine Gleichung, eine Unbekannte, drei Lösungen

$$L:=\left(\text{Solve}\left(x^3-3\cdot x^2+5\cdot x=15,\ x\right)\right)\qquad L=\begin{pmatrix}\left(x=\left(-i\cdot\sqrt{5}\right)\right)\\ \left(x=i\cdot\sqrt{5}\right)\\ (x=3)\end{pmatrix}$$

Mehrere Lösungen für eine einzelne Variable werden als Listen angezeigt. Man kann per Adressindex eine davon auswählen.

$\text{Assign}\left(L_1\right)=-i\cdot\sqrt{5}$	$x=-2.2361\cdot i$	$\text{Clear}(x)=1$
$\text{Assign}\left(L_2\right)=i\cdot\sqrt{5}$	$x=2.2361\cdot i$	$\text{Clear}(x)=1$
$\text{Assign}\left(L_3\right)=3$	$x=3$	$\text{Clear}(x)=1$

Bei der Lösung und der Auswahl der Lösung dürfen die verwendeten Variablen noch keinen Wert haben. Gegebenenfalls muss ein vorhandener Wert mit `Clear()` gelöscht werden.

Zwei Gleichungen, zwei Unbekannte, zwei Lösungen

Hier werden die Gleichungen einzeln zugewiesen. `Gl` ist ein Vektor, der alle Gleichungen des Systems enthält. Das kann sinnvoll sein, wenn Sie die Bedeutung oder Herleitung der Gleichungen einzeln erläutern wollen. Bei der Zuweisung muss die Gleichung in Klammern stehen, da der Zuweisungsoperator eine höhere Priorität als das Boolsche Gleichheitszeichen hat.

$$Gl_1 := \left(x^2 + 3\cdot x\cdot y + y^2 = 0\right) \qquad Gl_2 := (3\cdot x + y = 1)$$

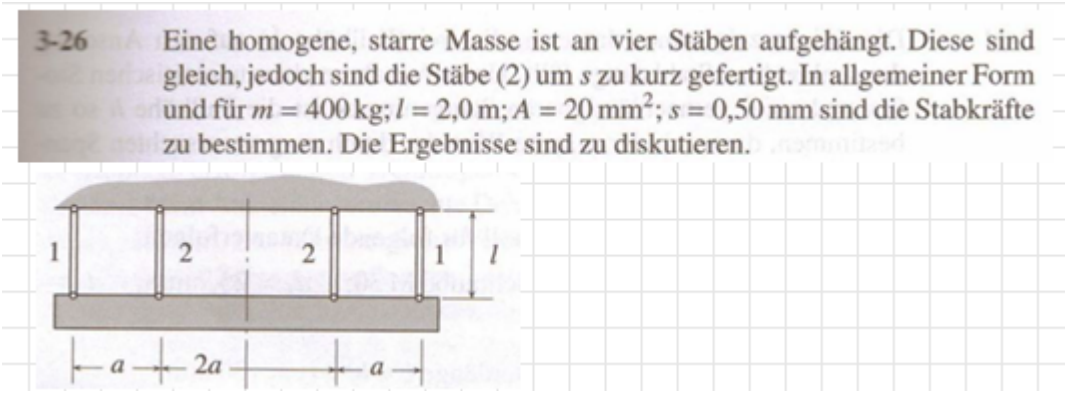
$$L := \mathrm{Solve}\left(Gl, \begin{pmatrix} x \\ y \end{pmatrix}\right) \qquad L = \left\{\left(x = \left(-\frac{-3+\sqrt{5}}{2}\right)\right) \left(y = \frac{-7+3\cdot\sqrt{5}}{2}\right)\right\} \left\{\left(x = \frac{3+\sqrt{5}}{2}\right) \left(y = \left(-\frac{7+3\cdot\sqrt{5}}{2}\right)\right)\right\}$$

$$\mathrm{Assign}\left(L_1\right) = \begin{cases} -\frac{-3+\sqrt{5}}{2} \\ \frac{-7+3\cdot\sqrt{5}}{2} \end{cases} \qquad x = -\frac{-3+\sqrt{5}}{2} \qquad y = \frac{-7+3\cdot\sqrt{5}}{2} \qquad \mathrm{Clear}(x, y) = 1$$

$$\mathrm{Assign}\left(L_2\right) = \begin{cases} \frac{3+\sqrt{5}}{2} \\ -\frac{7+3\cdot\sqrt{5}}{2} \end{cases} \qquad x = \frac{3+\sqrt{5}}{2} \qquad y = -\frac{7+3\cdot\sqrt{5}}{2} \qquad \mathrm{Clear}(x, y) = 1$$

4.13.1.1. Anwendungsbeispiel: Lineares System

Die Aufgabenstellung wurde aus Assmann/Selke 2009, Aufgabe 3-26, gescannt und als Bildbereich eingefügt.



Die gegebenen Größen werden mit den Maßeinheiten aus der Aufgabenstellung übernommen.

Gegeben $m := 400\ kg$ $l := 2\ m$ $A := 20\ mm^2$ $s := 0,5\ mm$

$E := 210000\ MPa$ (Annahme, hat keine Auswirkung auf das Ergebnis)

Gesucht S_1, S_2

Aus dem elastischen Gesetz für die Stäbe, der Geometriebedingung (alle Stäbe müssen im verformten Zustand gleich lang sein) und der Gleichgewichtsbedingung ergeben sich vier

Gleichungen, die das mathematische Modell des Problems bilden. Die Gleichungen werden in die Variable `Gl` an die durch den jeweiligen Adressindex bezeichnete Stelle geschrieben. Wenn diese Variable vorher nicht existiert, wird sie durch die Zuweisung von Elementen automatisch zu einem Vektor.

Modellbildung

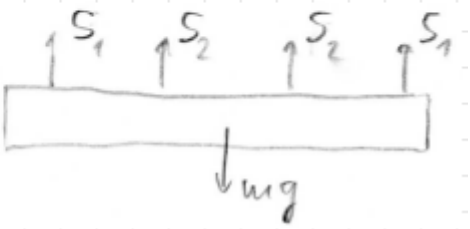
Elastisches Gesetz für die Stäbe

$Gl_1 := \left(\Delta l_1 = \frac{1}{E \cdot A} \cdot S_1\right) \qquad Gl_2 := \left(\Delta l_2 = \frac{1}{E \cdot A} \cdot S_2\right)$

Geometrie $\qquad Gl_3 := \left(\Delta l_2 = \Delta l_1 + s\right)$

Gleichgewicht an der Masse in vertikaler Richtung

$Gl_4 := \left(m\,g = 2 \cdot S_1 + 2 \cdot S_2\right)$



Mit der Funktion `Unknowns()` kann man prüfen, welche Variablen im Gleichungssystem unbekannt sind, also noch keinen Wert haben. Die Funktion `Solve()` löst das System und liefert das Ergebnis als Liste von logischen Gleichungen `Variable=Wert` zurück. Das ist hier nicht zu sehen, da das Ergebnis direkt in `Assign()` als Variablendefinition weiter verwendet wird.

Berechnung

$$\text{Unknowns}(Gl) = \begin{pmatrix} S_1 \\ S_2 \\ \Delta l_1 \\ \Delta l_2 \end{pmatrix} \qquad \text{Assign}\left(\text{Solve}\left(Gl;\begin{pmatrix} S_1 \\ S_2 \\ \Delta l_1 \\ \Delta l_2 \end{pmatrix}\right)\right) = \begin{pmatrix} 455{,}665\,\frac{kg\,m}{s^2} \\ 1505{,}665\,\frac{kg\,m}{s^2} \\ 0{,}0002\,m \\ 0{,}0007\,m \end{pmatrix}$$

$S_2 = 1506\,N$ $\qquad$ $S_1 = 456\,N$

4.13.1.2. Anwendungsbeispiel: Nichtlineares System

Die Aufgabenstellung ist wiederum Assmann/Selke 2009, Aufgabe 7-7, entnommen.

> Eine Druckspindel, $l = 800\,\text{mm}$ lang, mit $F = 20\,\text{kN}$ belastet, soll als Hohlwelle mit $d_i = 25\,\text{mm}$ ausgeführt werden. Für $S_K > 3{,}5$ ist der Außendurchmesser auf volle mm gerundet zu bestimmen.

Gegebene Größen

$l := 800\ mm$	Spindellänge
$F := 20\ kN$	Druckkraft
$d_i := 25\ mm$	Innendurchmesser
$S_K := 3,5$	Knicksicherheit
$E := 210000\ MPa$	E-Modul Stahl (Annahme)
$l_K := l$	beidseitig gelenkig gelagert (Annahme)

Gleichungen

Knicklast

$$Gl_1 := \left(F \cdot S_K = \frac{\pi^2 \cdot E \cdot I}{l_K^2}\right)$$

Flächenträgheitsmoment der Hohlwelle

$$Gl_2 := \left(I = \frac{\pi}{64} \cdot \left(d_a^4 - d_i^4\right)\right)$$

$$\text{Unknowns}(Gl) = \begin{pmatrix} d_a \\ I \end{pmatrix}$$

Gleichung 2 ist nichtlinear, es können also mehrere Lösungen auftreten. Die korrekte Lösung muss dann aus dem Zusammenhang der Aufgabenstellung ermittelt werden. Im vorliegenden Fall gibt es vier Lösungen (vier mögliche Werte für die vierte Wurzel).

Lösung

$$L := \left(\text{Solve}\left(Gl ; \begin{pmatrix} d_a \\ I \end{pmatrix}\right)\right)$$

$$L_1 = \left\{\begin{pmatrix} d_a = \frac{i\,m \cdot \sqrt[4]{65536 + 1875 \cdot \pi^3}}{200 \cdot \pi^{\frac{3}{4}} \cdot \sqrt[4]{3}} \\ I = \frac{m^4}{4687500 \cdot \pi^2} \end{pmatrix}\right.$$

$$L_2 = \left\{\begin{pmatrix} d_a = \left(-\frac{m \cdot \sqrt[4]{65536 + 1875 \cdot \pi^3}}{200 \cdot \pi^{\frac{3}{4}} \cdot \sqrt[4]{3}}\right) \\ I = \frac{m^4}{4687500 \cdot \pi^2} \end{pmatrix}\right.$$

$$L_3 = \left\{\begin{pmatrix} d_a = \left(-\frac{i\,m \cdot \sqrt[4]{65536 + 1875 \cdot \pi^3}}{200 \cdot \pi^{\frac{3}{4}} \cdot \sqrt[4]{3}}\right) \\ I = \frac{m^4}{4687500 \cdot \pi^2} \end{pmatrix}\right.$$

$$L_4 = \left\{\begin{pmatrix} d_a = \frac{m \cdot \sqrt[4]{65536 + 1875 \cdot \pi^3}}{200 \cdot \pi^{\frac{3}{4}} \cdot \sqrt[4]{3}} \\ I = \frac{m^4}{4687500 \cdot \pi^2} \end{pmatrix}\right.$$

Da der Durchmesser d_a positiv und reell sein muss, kommt nur Lösung 4 in Frage. Diese wird mit `Assign()` zugewiesen und weiterverarbeitet. Der gewählte Durchmesser wird als der nächstgrößere ganzzahlige mm-Wert mit Hilfe der Funktion `Ceil()` bestimmt, die bei Benutzung des Plugins *FunctionsExtensions* in der Operatorform $\lceil x \rceil$ dargestellt wird.

$$\text{Assign}\left(L_4\right)=\begin{cases}0{,}0302\ m\\ 2{,}1615\cdot10^{-8}\ m^4\end{cases}\qquad d_a=30{,}1923\ mm$$

$$d_{a\_gew}:=\left\lceil\frac{d_a}{mm}\right\rceil mm\qquad \boxed{d_{a\_gew}=31\ mm}$$

4.13.2. Numerische Lösung mit Startwerten: `FindRoot()`

Das Plugin *Non Linear Solvers* (Abschnitt 9.4.10) stellt eine Reihe von Funktionen zur Gleichungslösung bereit. Die wichtigste davon ist `FindRoot()`, die intern verschiedene spezielle Funktionen ausprobiert, um eine Lösung zu finden. Das Besondere an den Funktionen des Plugins ist, dass sie alle mit Maßeinheiten umgehen können, obwohl sie rein numerische Verfahren enthalten.

Die Funktion `FindRoot()` nimmt zwei oder drei Argumente:

1. Eine skalare Gleichung oder ein Gleichungssystem, gegeben als Vektor oder Matrix,
2. eine skalare Variable mit Anfangswert, gegeben als `Variable=Anfangswert` (logisches Gleichheitszeichen) oder ein Vektor oder eine Liste solcher Gleichungen.
3. Optional: eine Liste mit Toleranzen für die Erfüllung der Gleichungen. Dabei ist zu beachten, dass die Toleranzen die für die jeweilige Gleichung richtigen Maßeinheiten haben.

Da abhängig von den Anfangswerten immer nur eine Lösung ermittelt wird, kann diese auch gleich auf die Variablen zugewiesen werden, so dass der Zusatzschritt mit `Assign()` nicht erforderlich ist.

Die Lösung des Beispiels in Abschnitt 4.13.1.2 kann mit `FindRoot()` so aussehen:

Gleichungen

Knicklast

$$Gl_1:=\left(F\cdot S_K=\frac{\pi^2\cdot E\cdot I}{l_K^2}\right)$$

Flächenträgheitsmoment der Hohlwelle

$$Gl_2:=\left(I=\frac{\pi}{64}\cdot\left(d_a^4-d_i^4\right)\right)$$

$$\text{Unknowns}\left(Gl\right)=\begin{pmatrix}d_a\\ I\end{pmatrix}$$

$$\text{Clear}\left(d_a\ ;\ I\right)=1$$

$$\text{FindRoot}\left(Gl\ ;\begin{cases}d_a=d_i\\ I=d_i^4\end{cases};\begin{cases}1\ N\\ 1\ mm^4\end{cases}\right)=\begin{cases}0{,}0302\ m\\ 2{,}1615\cdot10^{-8}\ m^4\end{cases}$$

$$d_a=30{,}1923\ mm\qquad d_{a\_gew}:=\left\lceil\frac{d_a}{mm}\right\rceil mm\qquad \boxed{d_{a\_gew}=31\ mm}$$

Gleichung 1 hat die Dimension einer Kraft, daher wird als Toleranz hier ein Newton vorgegeben. Gleichung 2 hat die Dimension eines Flächenträgheitsmoments, daher ist auch eine entsprechende Toleranz erforderlich.

4.13.3. Numerische Lösung mit vorgegebenem Intervall: `Bisection()`

Wenn man sicher ist, dass es in einem bestimmten Intervall genau eine Lösung gibt, kann man auch die Funktion `Bisection()` aus dem Plugin *Non Linear Solvers* verwenden. Dabei wird das Halbteilungsverfahren verwendet. Dieses teilt das Intervall in zwei Teile und ermittelt das Vorzeichen, wenn die rechte Gleichungsseite von der linken abgezogen wird. Wechselt das Vorzeichen, muss die Lösung in dem betreffenden Intervall liegen. Auf diese Weise kann man die Lösung immer weiter eingrenzen, bis eine vorgegebene Toleranz erreicht ist.

Die Funktion `Bisection()` kann nur skalare Gleichungen nach einer Variablen auflösen, Gleichungssysteme sind hier also nicht lösbar. Die Argumente sind:

1. Die zu lösende Gleichung,
2. Die Variable, nach der aufzulösen ist und deren Untergrenze in der Form `Variable=Untergrenze`
3. Die Obergrenze der Variable
4. Optional: Toleranz für die Erfüllung der Gleichung.

Das soll wieder am Beispiel aus Abschnitt 4.13.1.2 demonstriert werden. Man muss jetzt allerdings die Gleichungen nacheinander auflösen, da `Bisection()` ja immer nur eine Gleichung lösen kann.

Wir bestimmen zunächst das erforderliche Flächenträgheitsmoment. Als Untergrenze setzen wir Null an, als Obergrenze einen Wert, von dem wir glauben, dass er größer als die tatsächliche Lösung ist.

$$\mathrm{Clear}(d_a ; I) = 1$$

$$\mathrm{Bisection}\left(F \cdot S_K = \frac{\pi^2 \cdot E \cdot I}{l_K^2} ; I = 0 ; 10\ cm^4\right) = 2{,}1615\ cm^4$$

Wir haben hier auf die Angabe einer Toleranz verzichtet, die Voreinstellung ist 10^{-4} bei Verwendung der Basiseinheit, hier also 10^{-4} N. Bei der Ermittlung des Durchmessers aus dem Trägheitsmoment funktioniert das nicht, der ermittelte Wert ist völlig falsch.

$$\mathrm{Bisection}\left(\left(I = \frac{\pi}{64} \cdot \left(d_a^4 - d_i^4\right)\right) ; d_a = d_i ; 10 \cdot d_i\right) = 137{,}5\ mm$$

Mit einer Toleranz von $10^{-4}\,I$ bekommen wir ein brauchbares Ergebnis:

$$\mathrm{Clear}(d_a) = 1$$

$$\mathrm{Bisection}\left(\left(I = \frac{\pi}{64} \cdot \left(d_a^4 - d_i^4\right)\right) ; d_a = d_i ; 10 \cdot d_i ; 10^{-4} \cdot I\right) = 30{,}1919\ mm$$

$$d_a = 30{,}1919\ mm \qquad d_{a\_gew} := \left\lceil \frac{d_a}{mm} \right\rceil mm \qquad \boxed{d_{a\_gew} = 31\ mm}$$

4.14. Gewöhnliche Differenzialgleichungen

SMath verfügt über keinerlei eingebaute Funktionen zur Lösung von gewöhnlichen Differenzialgleichungen. Allerdings gibt es einige Plugins, die numerische Verfahren bereitstellen. Zudem bieten die Plugins *MapleWrapper* und *Maxima* symbolische Lösungsfunktionen.

Funktion	Seite	Erläuterung
Plugin *AlgLib*		
`al_rkckadapt()`		Runge-Kutta-Cash-Karp-Verfahren
Plugin *DotNumerics*		
`dn_AdamsMoulton()`		Adams-Moulton-Verfahren für steife Systeme
`dn_ExplicitRK45()`		Explizites Runge-Kutta-Verfahren
`dn_GearsBDF()`		Gears BDF-Verfahren
`dn_ImplicitRK45()`		Implizites Runge-Kutta-Verfahren für steife Systeme
Plugin *ODE Solver*		
`Rkadapt()`	330	Runge-Kutta-Verfahren mit adaptiver Schrittweite
`rkfixed()`	331	Runge-Kutta-Verfahren mit fester Schrittweite
Plugin *Maxima*		
`ODE.2()`	315	Symbolischer Löser

Tabelle 4.8.: Funktionen zur Lösung gewöhnlicher Differenzialgleichungen

4.14.1. Löser aus dem Maxima-Plugin

Bemerkenswerte Erweiterungen für Differenzialgleichungen durch das Plugin *Maxima* sind:

- die Funktion `ODE.2()` zur symbolischen Lösung gewöhnlicher DGL bis zweiter Ordnung,
- die Maxima-Funktionen zur Laplace-Transformation,
- die Richtungsfeld-Diagrammfunktionen des Pakets *drawdf* (draw direction field).

Funktion ODE.2()

Für die Lösung von Differenzialgleichungen bis zu zweiter Ordnung steht die Funktion `ODE.2()` zur Verfügung. Anzugeben sind die Differenzialgleichung, die unbekannte Funktion und die unabhängige Variable.

$$\text{Maxima}\left(\text{assume}\left(\omega^2>0\right)\right)=\omega^2>0$$

$$L:=\text{ODE}_2\left(\left(\frac{d^2}{dt^2}w(t)+\omega^2\cdot w(t)=0\right);\ w(t);\ t\right)$$

$$L=w(t)=\left(k1\cdot\sin(\omega\cdot t)+k2\cdot\cos(\omega\cdot t)\right)$$

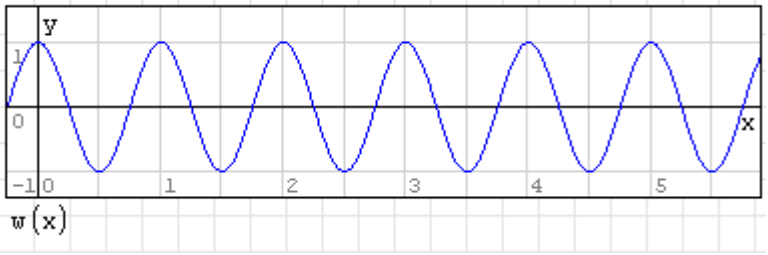
Zurückgegeben wird eine boolsche Gleichung, die mit `Assign()` zur Definition der unbekannten Funktion verwendet werden kann. Die Lösung enthält abhängig von der Ordnung der DGL eine oder zwei Integrationskonstanten, `c` (erste Ordnung) oder `k1` und `k2` bei zweiter Ordnung.

$$\text{Assign}(L) = (k1 \cdot \sin(\omega \cdot t) + k2 \cdot \cos(\omega \cdot t)) \qquad w(t) = k1 \cdot \sin(\omega \cdot t) + k2 \cdot \cos(\omega \cdot t)$$

Die Konstanten können aus den Anfangsbedingungen bestimmt werden, hier z.B. wird die Auslenkung zu Eins gesetzt und die Geschwindigkeit gleich Null.

$$\text{Assign}\left(\text{Solve}\left(\left\{\begin{array}{l} w(0) = 1 \\ \left.\frac{d}{dt}(w(t))\right|_{t=0} = 0 \end{array}\right., \begin{pmatrix} k1 \\ k2 \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) \qquad w(t) = (0 \cdot \sin(\omega \cdot t) + 1 \cdot \cos(\omega \cdot t))$$

$\omega := 2 \cdot \pi$



4.15. Statistik und Datenanalyse

Hier geht es um

- Deskriptive Statistik, also die Berechnung von statistischen Merkmalen gegebener Daten,
- Wahrscheinlichkeitsverteilungen und Spezialfunktionen für statistische Modelle
- Regression, also die Anpassung statistischer Modelle an gegebene Daten
- Interpolation ein- und zweidimensionaler Datenfelder

Funktionen zum Datenimport werden in Abschnitt 3.18 behandelt.

Daten liegen in der Regel als Matrizen oder Vektoren vor, der Umgang mit solchen Objekten wird in Abschnitt 4.9 besprochen.

4.15.1. Deskriptive Statistik

Funktion	Seite	Bedeutung
HarmonicMean()	289	Harmonischer Mittelwert
Max()	306	Maximum
Mean()	309	Arithmetischer Mittelwert
Median()	309	Median
Min()	309	Minimum
Mode()	310	Modalwert
Moment()	310	Moment
Skewness()	332	Schiefe
Slope()	332	Geradensteigung
StdDev()	335	Standardabweichung
Variance()	341	Varianz
WeightedMean()	342	Gewichteter Mittelwert

Tabelle 4.9.: Funktionen für die statistische Beschreibung von Daten

4.15.2. Spezielle Funktionen

Die folgenden speziellen Funktionen werden für die Berechnung statistischer Verteilungen benötigt und können für eigene statistische Modelle verwendet werden.

Name	Seite	Bedeutung
Beta()	254	Beta-Funktion, Eulersches Integral 1. Art
BetaRegularized()	255	Regularisierte unvollständige Beta-Funktion
Dirac()	270	Dirac-Funktion
erf()	273	Fehlerfunktion
erfc()	274	Komplementäre Fehlerfunktion
erfinv()	274	Inverse Fehlerfunktion
Gamma()	281	Gamma-Funktion
GammaRegularized.P()	282	Regularisierte Gamma-Funktion der oberen Grenze
GammaRegularized.Q()	282	Regularisierte Gamma-Funktion der unteren Grenze
Heaviside()	289	Sprungfunktion
Heaviside.D()	290	unsymmetrische Sprungfunktion

Abbildung 4.2.: Spezielle Funktionen aus dem Plugin *Statistical Tools*

4.15.3. Verteilungen

Verteilungen werden in der Regel mit jeweils drei Funktionen unterstützt, die mit den Bezeichnern `CDF`, `pdf` oder `ICDF` beginnen und den Verteilungsnamen als Index tragen:

- `CDF` (cumulative density function) Verteilungsfunktion oder kumulative Dichtefunktion, sie gibt die Wahrscheinlichkeit an, gemäß der Verteilung einen Wert kleiner gleich x zu finden.
- `pdf` (probability density function) Häufigkeitsfunktion oder Wahrscheinlichkeitsdichtefunktion.
- `ICDF` (inverse cumulative density function) Quantilfunktion oder inverse Verteilungsfunktion, sie gibt an, bei welchem x-Wert die Verteilungsfunktion einen bestimmten Wahrscheinlichkeitswert erreicht hat.

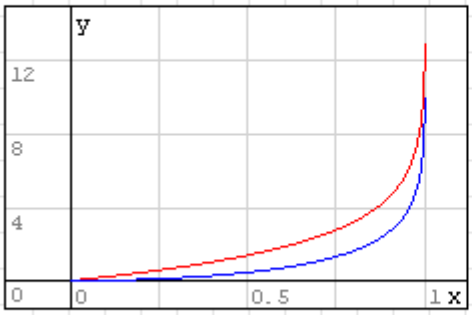
Verteilung	Bezeichner	CDF	ICDF	pdf
Binomialverteilung	`Binomial`	257	292	315
Cauchy-Verteilung	`Cauchy`	258	292	316
χ^2-Verteilung	`ChiSquare`	258		317
Exponentialverteilung	`Exponential`	259	292	317
Fisher-Verteilung	`F`	260		318
Geometrische Verteilung	`Geometric`	260	293	318
Verschobene geometrische Verteilung	`GeometricShifted`	261	293	319
Normalverteilung	`Normal`	261	294	319
Poisson-Verteilung	`Poisson`	262	294	320
Rayleigh-Verteilung	`Rayleigh`	263	295	320
Studentsche t-Verteilung	`t`	264		321
Gleichverteilung	`Uniform`	265	295	321
diskrete Gleichverteilung	`UniformDiscrete`	265	296	322
Weibullverteilung	`Weibull`	266	296	322

Tabelle 4.10.: Statistische Verteilungen mit den zugehörigen Verteilungs- (`CDF`), Quantil- (`ICDF`) und Häufigkeitsfunktionen (`pdf`). Die Verweise beziehen sich auf die Seitenzahlen der Funktion in der alphabetischen Funktionsliste in Kapitel D.

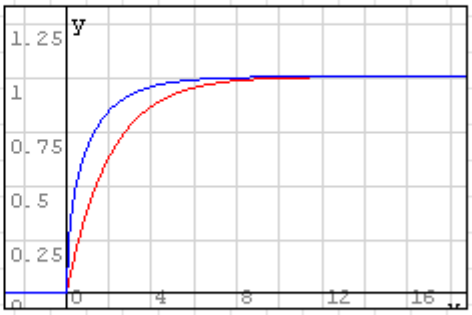
Die in der Tabelle fehlenden Quantilfunktionen können mit dem Maxima-Paket *distrib* ergänzt werden.

$$ICDF_{ChiSquare}(x) := eval\left(\text{float}\left(\text{quantile\_chi2}(x\ ;\ 1)\right)\right)$$

$$ICDF_{ChiSquare}(x\ ;\ n) := eval\left(\text{float}\left(\text{quantile\_chi2}(x\ ;\ n)\right)\right)$$



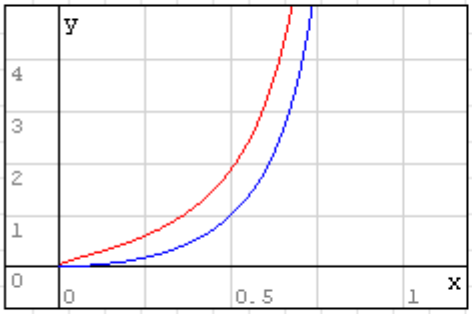
$$\begin{cases} ICDF_{ChiSquare}(x) \\ ICDF_{ChiSquare}(x\ ;\ 2) \end{cases}$$



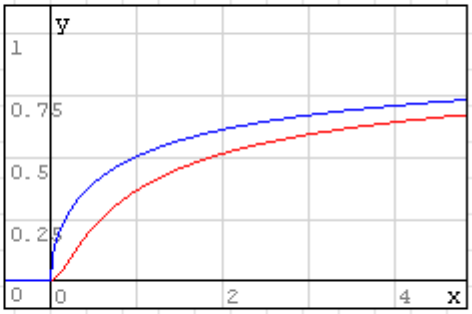
$$\begin{cases} CDF_{ChiSquare}(x) \\ CDF_{ChiSquare}(x\ ;\ 2) \end{cases}$$

$$ICDF_{F}(x) := eval\left(\text{float}\left(\text{quantile\_f}(x\ ;\ 1\ ;\ 1)\right)\right)$$

$$ICDF_{F}(x\ ;\ m\ ;\ n) := eval\left(\text{float}\left(\text{quantile\_f}(x\ ;\ m\ ;\ n)\right)\right)$$



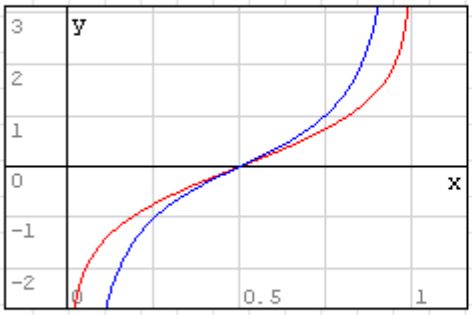
$$\begin{cases} ICDF_{F}(x) \\ ICDF_{F}(x\ ;\ 5\ ;\ 1) \end{cases}$$



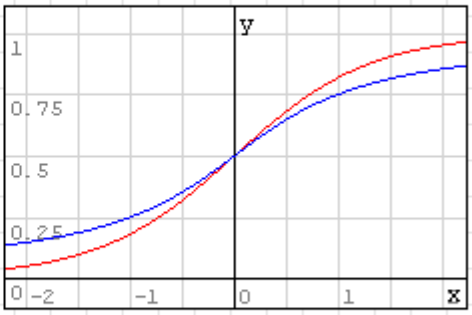
$$\begin{cases} CDF_{F}(x) \\ CDF_{F}(x\ ;\ 5\ ;\ 1) \end{cases}$$

$$ICDF_{t}(x) := eval\left(\text{float}\left(\text{quantile\_student\_t}(x\ ;\ 1)\right)\right)$$

$$ICDF_{t}(x\ ;\ n) := eval\left(\text{float}\left(\text{quantile\_student\_t}(x\ ;\ n)\right)\right)$$



$$\begin{cases} ICDF_{t}(x) \\ ICDF_{t}(x\ ;\ 5) \end{cases}$$



$$\begin{cases} CDF_{t}(x) \\ CDF_{t}(x\ ;\ 5) \end{cases}$$

4.15.4. Nichtlineare Regression

Multilineare Regressionen nach der Fehlerquadratmethode können mit den SMath-Bordmitteln durchgeführt werden.

Für nichtlineare Regressionsprobleme kann das Maxima-Paket *lsquares* benutzt werden. Die Funktion `Fit()` des Maxima-Plugins bildet hierfür eine bequeme Schnittstelle.

```
Fit(data; vars; eqn; pars; init)
```

`data` Matrix mit den gegebenen Daten. Jede Zeile ist ein Datensatz, jede Spalte eine Variable.

`vars` Liste der Variablennamen (für jede Spalte in `data` ein Eintrag).

`eqn` Gleichung, verknüpft die in `vars` aufgelisteten Größen und die in `pars` aufgelisteten Ausgleichsparameter.

`pars` Namen der freien Parameter in der Gleichung `eqn`, gegeben als Liste.

`init` Startwerte für die Suche nach den optimalen Parametern.

Rückgabewert ist eine Liste mit Gleichungen der Form `Parameter=Wert`. Dieser kann genau wie bei `Solve()` mit `Assign()` für die Zuweisung der Werte an die Parameter benutzt werden.

Beispiel 1:

$$Data := \begin{pmatrix} 1 & 1{,}5 & 2{,}25 & 3 & 2 \\ 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 \end{pmatrix}^{T}$$

$$Fit\left(Data;\begin{pmatrix} z \\ x \\ y \end{pmatrix};(z+D)^{2}=(C+A\cdot x+B\cdot y);\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix};\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}\right)=\begin{pmatrix} A=-3{,}628998943824597 \\ B=-1{,}652625813237137 \\ C=10{,}45496853509099 \\ D=-3{,}319624288317973 \end{pmatrix}$$

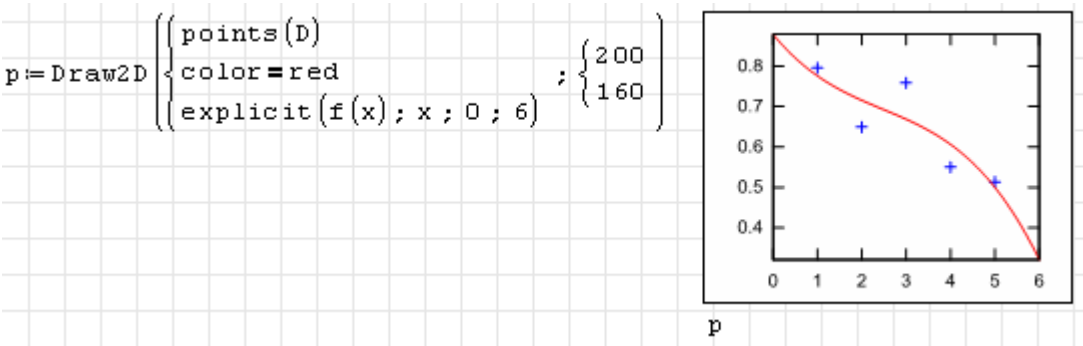
Beispiel 2:

$$D := augment(1..5;Random(5))$$

$$f(x) := a\cdot x^{3}+b\cdot x^{2}+c\cdot x+d$$

$$D=\begin{pmatrix} 1 & 0{,}796 \\ 2 & 0{,}6491 \\ 3 & 0{,}7588 \\ 4 & 0{,}5501 \\ 5 & 0{,}5136 \end{pmatrix}$$

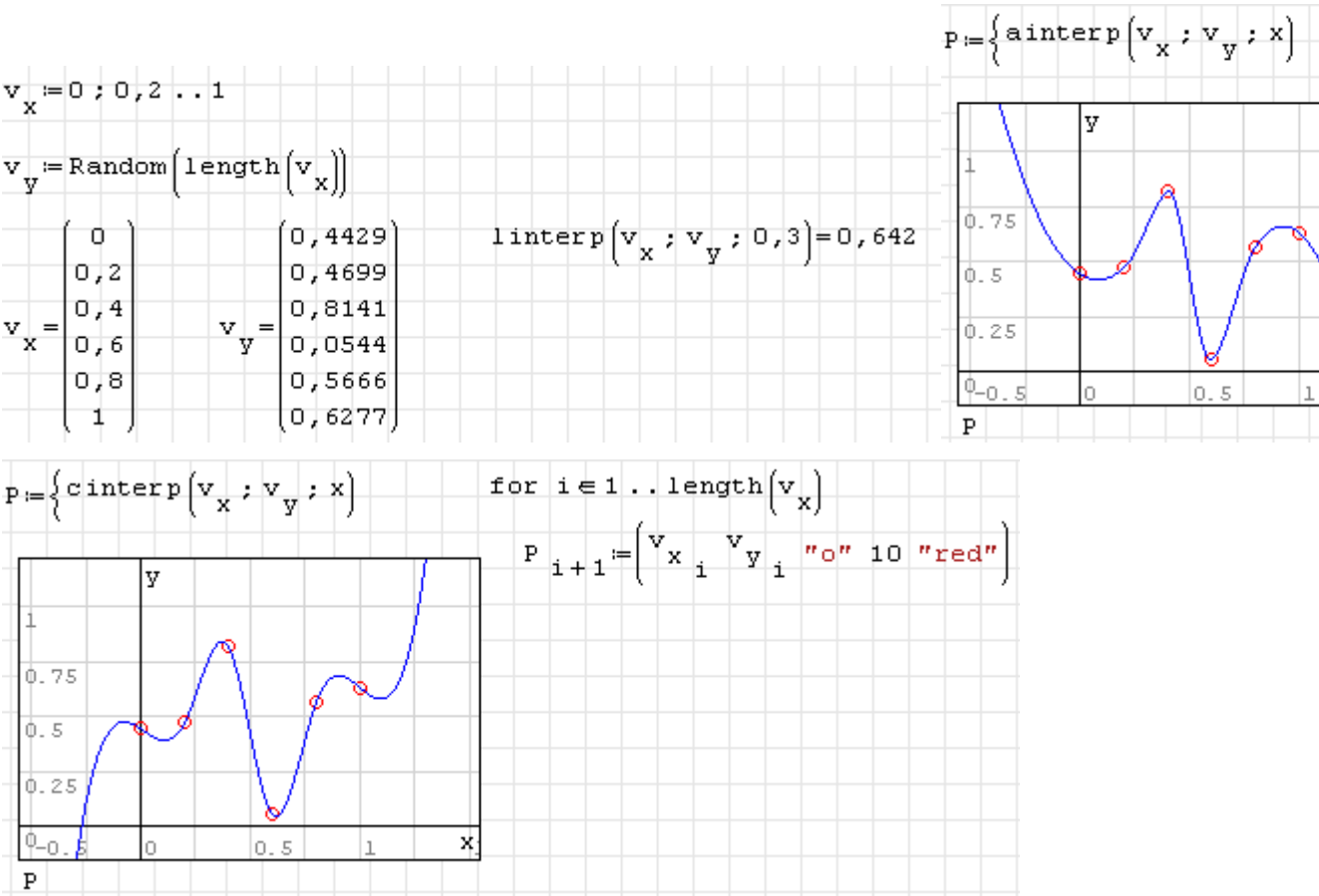
$$Assign\left(Fit\left(D;\begin{pmatrix} x \\ y \end{pmatrix};y=f(x);\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix};\begin{pmatrix} 1, \\ 3, \\ 1, \\ 1, \end{pmatrix}\right)\right)=\begin{pmatrix} -0{,}0048 \\ 0{,}0356 \\ -0{,}1337 \\ 0{,}879 \end{pmatrix}$$



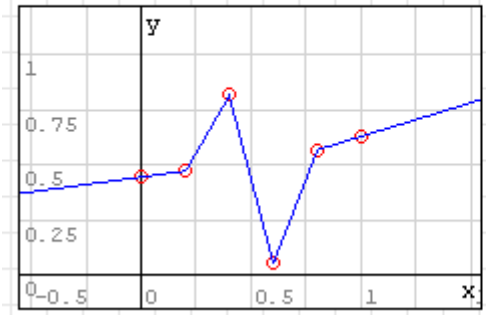
4.15.5. Interpolation

SMath bietet eingebaute Funktionen für die eindimensionale Interpolation:

- `ainterp(vx; vy; x)` kubische Spline-Interpolation der Vektoren `X-Daten` und `Y-Daten` am Punkt x nach Akima Erläuterung des Verfahrens (TU Wien). Das Verfahren berechnet vor der Interpolation an den Stützstellen die Neigung aus einer 5-Punkte-Umgebung.
- `cinterp(vx; vy; x)` kubische Spline-Interpolation
- `linterp(vx; vy; x)` lineare Interpolation



$$P := \begin{cases} \text{linterp}(v_x ; v_y ; x) \\ \text{for } i \in 1 .. \text{length}(v_x) \\ \quad P_{i+1} := \left(v_{x_i} \; v_{y_i} \; \text{"o"} \; 10 \; \text{"red"}\right) \end{cases}$$



Diese Funktionen können leider nicht abgeleitet werden und nicht mit Maßeinheiten umgehen.

Für die Interpolation von zweidimensionalen Tabellendaten steht die Funktion InterpBilinear() zur Verfügung. Die Daten sind als Matrix von z-Werten gegeben, deren Spalten bestimmten x-Werten und deren Zeilen bestimmten y-Werten zugeordnet sind. Die x- und y-Werte werden als Vektoren angegeben.

$$X := (20\ 50\ 100\ 150\ 200\ 250)$$

$$Y := \begin{pmatrix} 15 \\ 20 \\ 25 \\ 40 \\ 50 \end{pmatrix} \qquad M := \begin{pmatrix} 4 & 9 & 17 & 25 & 35 & 45 \\ 4 & 10 & 19 & 28 & 39 & 50 \\ 5 & 11 & 20 & 31 & 42 & 54 \\ 5 & 12 & 23 & 35 & 47 & 60 \\ 6 & 14 & 26 & 38 & 51 & 66 \end{pmatrix}$$

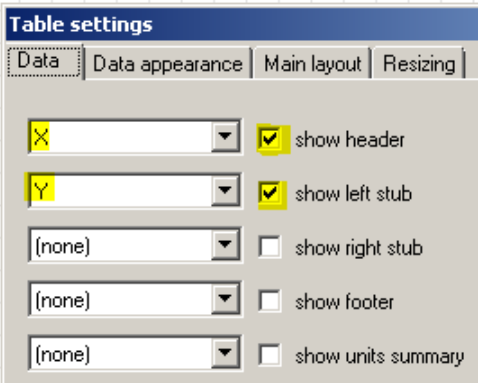
$$\text{InterpBilinear}(X ; Y ; M ; 30 ; 40) = 7,3333$$

Die Matrix und die Vektoren können übrigens übersichtlich mit dem Tabellenbereich dargestellt werden:

	20	50	100	150	200	250
15	4	9	17	25	35	45
20	4	10	19	28	39	50
25	5	11	20	31	42	54
40	5	12	23	35	47	60
50	6	14	26	38	51	66

Table 1

M



5. Grafik

Grundsätzlich werden die auf Gnuplot basierenden Grafikfunktionen des *Maxima*-Plugins empfohlen, welches die Funktionen `Draw2D()` und `Draw3D()` bereitstellt. Diese erzeugen Grafikdateien, die mit dem Image-Bereich angezeigt werden können.

Für einfache Diagramme mit wenig Anspruch an die Formatierung oder wenn Animationen erstellt werden sollen, eignet sich der Standard-2D-Diagrammbereich, den SMath Studio standardmäßig mitbringt.

Das Plugin *GPCPlugin* enthält Funktionen zum Umgang mit ebenen Polygonen.

Das *Modeller* Plugin stellt einen flexiblen interaktiven Grafikbereich bereit, der allerdings nur low-level Objekte bietet.

Weitere, allerdings nicht für den Standardanwender empfohlene Diagrammarten werden im Dachboden (Anhang 349) dokumentiert:

- Standard-3D-Diagrammbereich (fehlerbehaftet, nicht in der Weiterentwicklung)
- Plugin *X-Y Plot Region*: 2D-Diagramm
- Plugin *3D Plot Region*: 3D-Diagramm

Funktion	2D intern	3D intern	X-Y Plot	ZedGraph	3D Plot	Maxima
Objekte						
Funktionen	f(x)	f(x,y)	f(1)		f(2)	beliebig
Implizite Funktionen	-	-	**f(2)**		-	beliebig
Ausdrücke	x	x,y	beliebig		-	beliebig
Matrizen	$n \times 2$	$n \times 3$	$n \times 2$		-	$n \times 2$, $n \times 3$,
Text	**x**	-	-		-	x
Achsen						
Gitter	fest	fest	**variabel**	**variabel**	**variabel**	variabel
Grenzen	Maus	Maus	Menü	Menü	Menü	variabel
lin/log	-	-	-	x	-	x
polar	-	-	-		-	(x)
Formatierung						
Linien	fest	fest	**variabel**	**variabel**	?	variabel
Marker	separat	-	**variabel**	**variabel**	?	variabel
Sonderfunktionenx						
Animation	**x**	**x**	-	-	-	-
Entwicklung	stabil	inaktiv	inakativ	inaktiv	inaktiv	aktiv

5.1. Diagramme mit *Maxima*

Mit den Funktionen `Draw2D()` und `Draw3D()` können 2D- und 3D-Diagramme erstellt werden. Das geschieht unter Verwendung von Gnuplot, einem flexiblen und leistungsfähigen quelloffenen Grafikprogramm. Fähigkeiten und Eigenschaften der Schnittstelle sind:

- 2D- und 3D-Diagramme
- Grafikobjekte:
  - explizit, implizit und parametrisch gegebene Kurven und Flächen
  - Datenpunkte und Farbwerte aus Matrizen
  - Vektoren
  - Polygone, Ellipsen, Rechtecke
  - Beschriftungen
  - zusätzliche x- und y-Achsen
- Objekte und Optionen (Einstellungen) werden in Listen an die Grafikbefehle übergeben.
- Andere Maxima-Pakete definieren Spezialdiagramme, z.B. aus der Statistik oder Richtungsfelder für Differenzialgleichungen.

Neben dem Maxima-Handbuch findet man auch hier eine Übersicht über die Möglichkeiten des Maxima-Pakets *draw*.

Für die Verwendung in SMath werden Diagramme als temporäre Grafikdatei exportiert (svg) und dann mit einem Image-Bereich dargestellt.

Mit zusätzlichen Argumenten kann man Größe und Dateiname des Diagramms bestimmen:

```
Draw2D(Objekte)
Draw2D(Objekte; Name)
Draw2D(Objekte; Größe)
Draw2D(Objekte; Name; Größe)
```

Wird `name` angegeben, dann wird die Datei mit diesem Namen relativ zum aktuellen Verzeichnis erzeugt. Hat `name` die Endung *png*, wird eine Datei im Format PNG erzeugt, ansonsten als SVG. Die Größe kann angegeben werden durch eine Liste der Form

$$\begin{cases} \text{Breite} \\ \text{Höhe} \end{cases}$$

Für dreidimensionale Diagramme verwenden Sie die Funktion

```
Draw3D()
```

Sie hat die gleichen Argumente wie Draw2D, allerdings bei einer Standardgröße von 300x300 Punkten.

Die Bilder passen sich der Größe des Image-Bereichs an (bei Beibehaltung des Seitenverhältnisses), die Schrift skaliert dabei mit. In der Standardgröße entspricht die Schriftgröße 8 Punkten.

Einschränkungen sind momentan:

- Programmier-ähnliche Erstellung von Diagrammen (man muss die Objekte und Optionen kennen, keine Menü-Führung),
- Keine interaktive Skalierung und Drehung der Achsen.

- Die Grafikobjekte `polar()` und `spherical()` funktionieren gegenwärtig nicht. In Maxima selbst ist das nicht reproduzierbar.
- Einheitenbehaftete Größen werden in Basiseinheiten umgerechnet und dann dargestellt. Soll eine spezifische Maßeinheit verwendet werden, so sind abhängige Variablen oder Funktionen durch die entsprechende Einheit zu teilen und unabhängige Variablen mit diesen Einheiten zu multiplizieren.
- Nicht alle Funktionen und Ausdrücke werden korrekt von SMath nach Maxima übersetzt. Nicht unterstützt werden beispielsweise logische Ausdrücke.

5.1.1. Achsen und Gitter

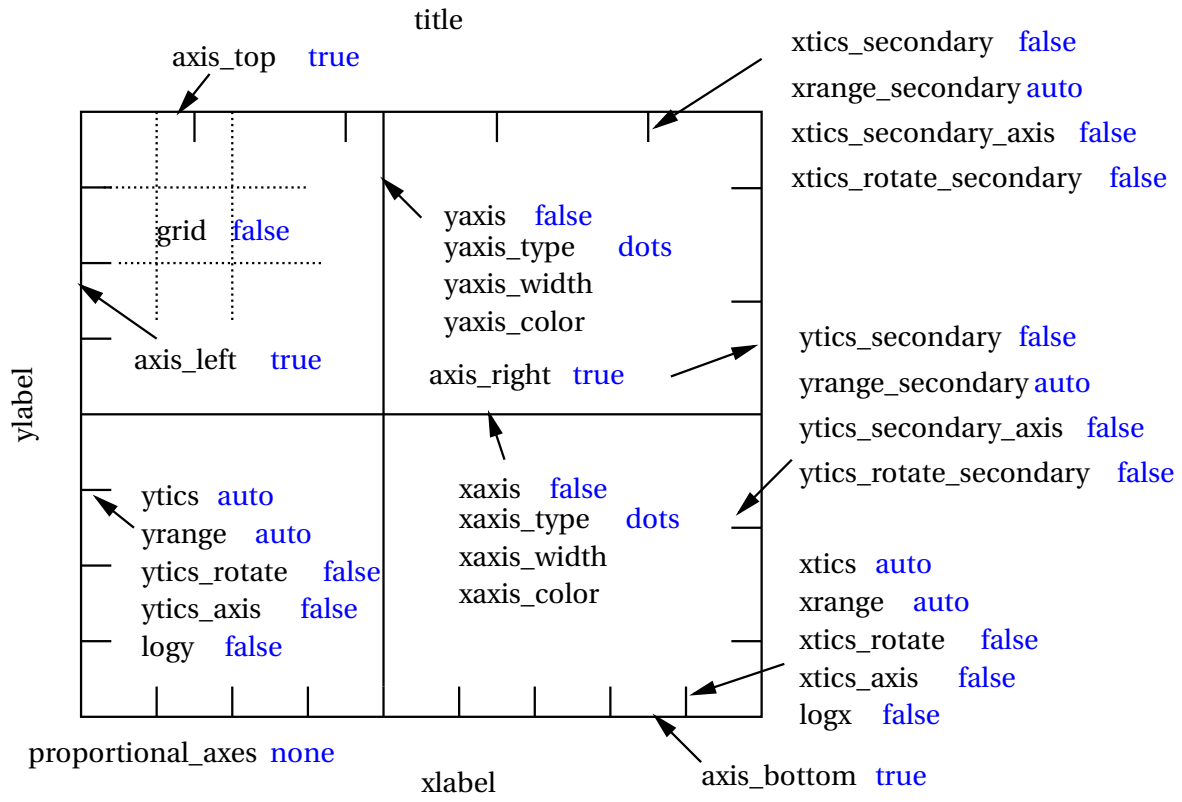


Abbildung 5.1.: Optionen zur Achsenformatierung in `Draw2D()`

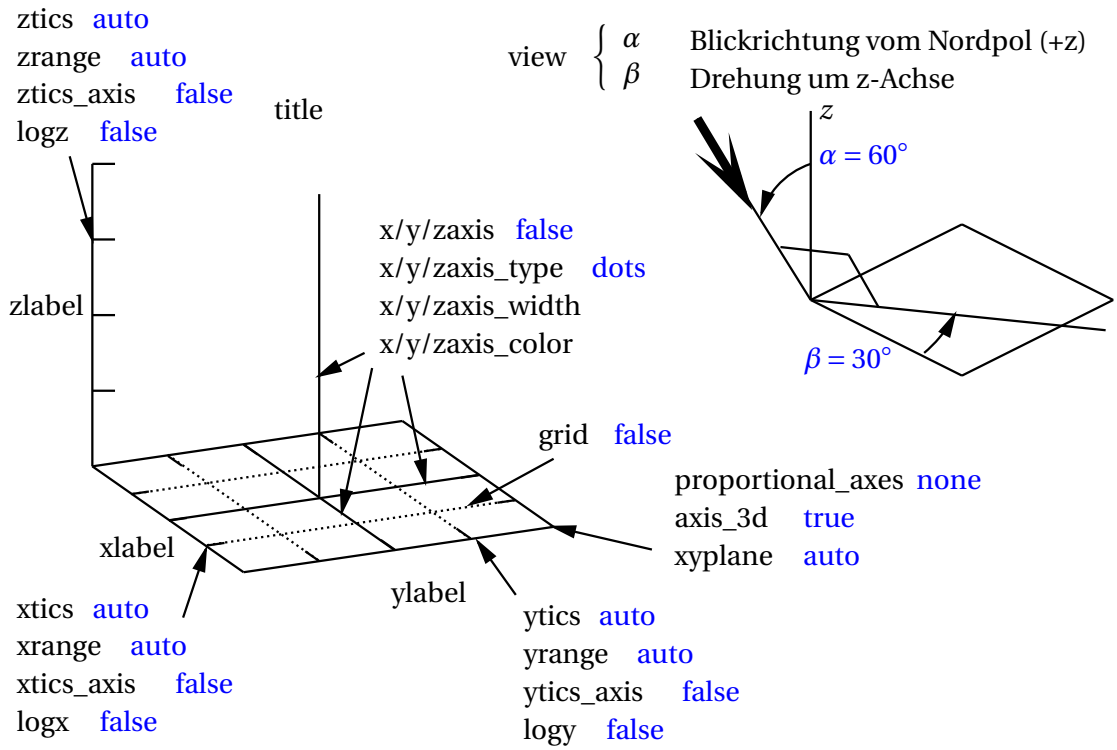
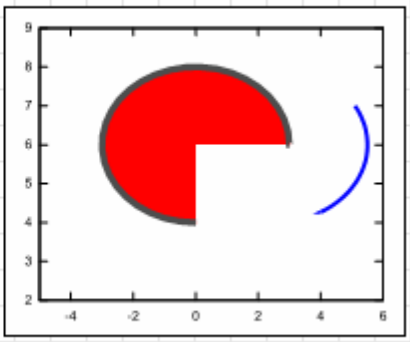


Abbildung 5.2.: Optionen zur Achsenformatierung in `Draw3D()`

5.1.2. Einfache Grafikobjekte

Ellipsen und Kreise können mit `ellipse()` gezeichnet werden, allerdings nicht mit gedrehten Hauptachsen.

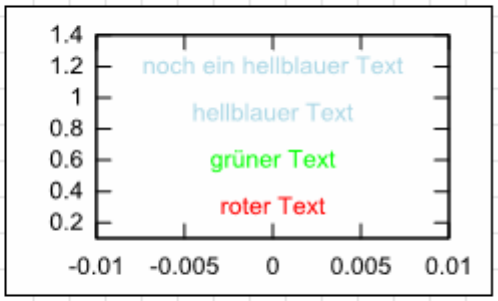
```
cm:=[fill_color=red
     color=gray30
     line_width=5
     ellipse(0;6;3;2;270;-270)
     transparent=true
     color=blue
     line_width=3
     ellipse(2,5;6;3;2;30;-90)
     xrange=[-5,6]
     yrange=[2,9]]
p:=Draw2D(cm)
```



Mit `label()` kann man Text an beliebigen Positionen einfügen (2D und 3D). Die Schriftgröße

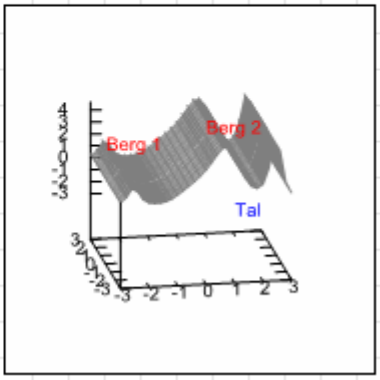
lässt sich jedoch nicht verändern.

```
cm:=[user_preamble={"set encoding iso_8859_2"
     yrange={0,1
             1,4
     color=red
     label({"roter Text"
            0
            0,3
     color=green
     label({"grüner Text"
            0
            0,6
     color=light_blue
     label({"hellblauer Text"  ;{"noch ein hellblauer Text"
            0                      0
            0,9                    1,2
p:=Draw2D(cm;{200
              120)
```



p

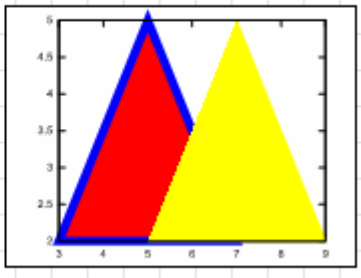
```
cm:=[color=gray50
     explicit(e^(sin(x)+cos(x^2)) ; x ; -3 ; 3 ; y ; -3 ; 3)
     color=red
     label({"Berg 1"  ;{"Berg 2"
            -2             1,5
            0              0
            3              4
     color=blue
     label({"Tal"
            2
            0
            -3
     view={70
           350
p:=Draw3D(cm;{200
              200)
```



p

Geschlossene Polygone zeichnet man mit `polygon()`:

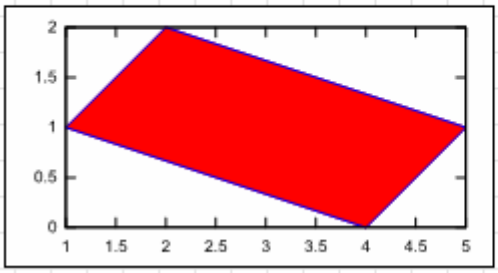
```
cm:=[line_width=8
     polygon(args({3 2
                   7 2
                   5 5)))
     border=false
     fill_color=yellow
     polygon(args({5 2
                   9 2
                   7 5)))
p:=Draw2D(cm)
```



p

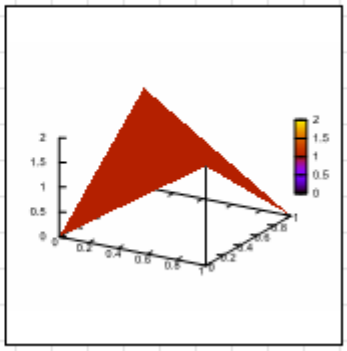
Eine Sonderform des Polygons ist `quadrilateral()`, dieses gibt es sowohl 2D als auch 3D:

```
p:Draw2D([quadrilateral([1,1];[2,2];[5,1];[4,0]);proportional_axes=xy];[300,160])
```



p

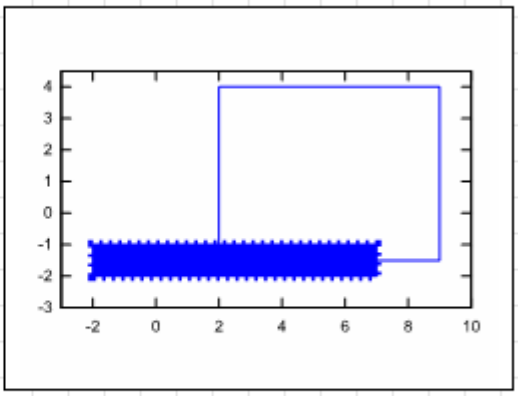
```
cm:[enhanced3d=true,quadrilateral([0,0,0];[1,0,2];[0,1,2];[1,1,0]),xyplane=0]
p:Draw3D(cm)
```



p

Rechtecke werden mit `rectangle()` gezeichnet:

```
cm:[line_width=6,line_type=dots,fill_color=blue,rectangle([-2,-2];[7,-1]),transparent=true,line_type=solid,line_width=1,rectangle([9,4];[2,-1,5]),xrange=[-3,10],yrange=[-3,4,5],proportional_axes=xy]
p:Draw2D(cm)
```

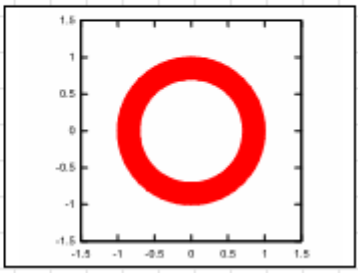


p

Mit `region(Bedingung; x; xmin; xmax; y; ymin; ymax)` wird ein Gebiet gezeichnet

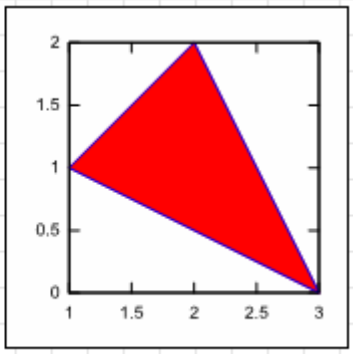
(nur Flächenfüllung, kein Rand), welches eine logische Bedingung erfüllt:

```
R:=((x^2+y^2<1) and (x^2+y^2>1/2))
cm:=[x_voxel=30,
     y_voxel=30,
     region(R,x,-3/2,3/2,y,-3/2,3/2),
     proportional_axes=xy]
p:=Draw2D(cm)
p
```



Dreiecke werden mit `triangle()` gezeichnet 2D und 3D:

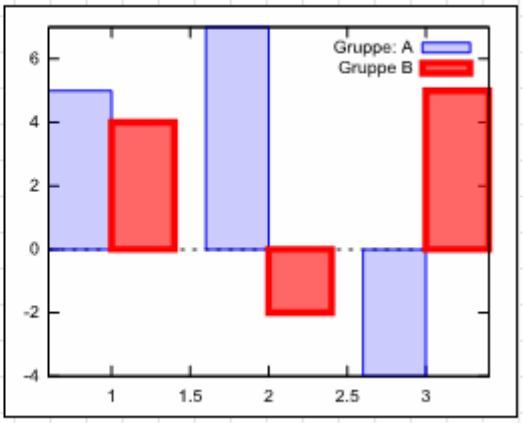
```
p:=Draw2D([[triangle([1,1],[2,2],[3,0]),200,200],
           proportional_axes=xy])
p
```



5.1.3. Darstellung von Wertetabellen

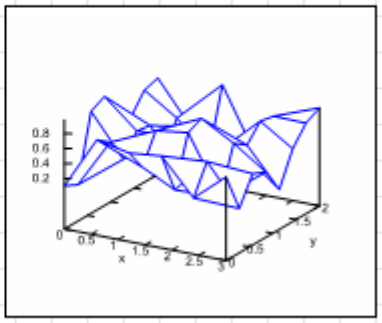
Die Funktion `bars()` kann für die Konstruktion von Balkendiagrammen benutzt werden, die Balkenabmessungen müssen allerdings komplett vorgegeben werden. Argumente sind Listen aus x-Position der Balkenmitte, Balkenhöhe und Balkenbreite.

```
cm:=[key="Gruppe: A",
     fill_color=blue,
     fill_density=0,2,
     bars([0,8,1,8,2,8],[5,7,-4],[0,4,0,4,0,4]),
     key="Gruppe B",
     fill_color=red,
     fill_density=0,6,
     line_width=4,
     bars([1,2,2,2,3,2],[4,-2,5],[0,4,0,4,0,4]),
     xaxis=true]
p:=Draw2D(cm)
p
```



Mit `elevation_grid()` kann man Matrizen als Höhenwerte auf einem äquidistanten xy-Gitter interpretieren.

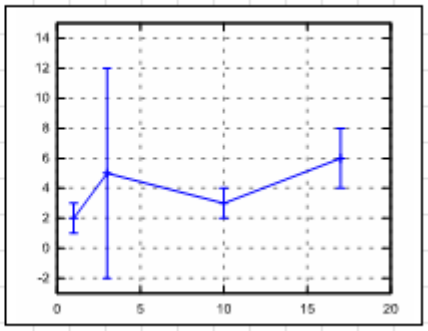
```
M:=Random(8 ; 6)
cm:=[elevation_grid(M ; 0 ; 0 ; 3 ; 2),
     xlabel="x",
     ylabel="y",
     ztics=0,2,
     surface_hide=true]
p:=Draw3D(cm)
```



p

Mit `errors()` können Fehlerbalken dargestellt werden.

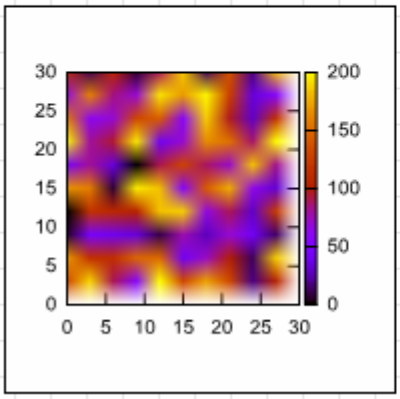
```
cmds:=[error_type=y,
       points_joined=true,
       errors(args([[1 2 1],[3 5 7],[10 3 1],[17 6 2]])),
       xrange=[0,20],
       yrange=[-3,15],
       grid=true]
p:=Draw2D(cmds)
```



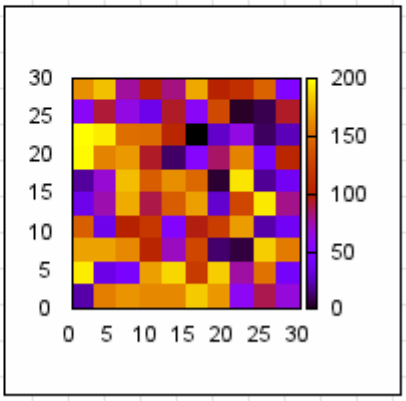
p

Mit `image(Matrix; xmin; ymin; xmax; ymax)` kann man Matrizen reeller Werte als Farbwerte in der xy-Ebene darstellen. Dabei sieht in diesem Fall die png-Ausgabe besser aus.

```
obj:=[palette=color,
      image(Random_N(10 ; 10 ; 0 ; 200) ; 0 ; 0 ; 30 ; 30),
      proportional_axes=xy]
ps:=Draw2D(obj ; "draw_im12" ; [200,200])
pp:=Draw2D(obj ; "draw_im12.png" ; [200,200])
```



ps



pp

`points(Liste_x; Liste_y)` 2D Punkte darstellen

`points(Liste_x; Liste_y; Liste_z)` 3D Punkte darstellen

`points(Matrix)` 2D oder 3D Punkte darstellen

`color=blue` Farbe

`key=""` Legendeneintrag

`point_size=1` Punktgröße

`point_type=plus` Punktart, gegeben als Zahl oder als Name.

`points_joined=false` Verbindungsart:

`false` keine Verbindung

`true` Polygonzug

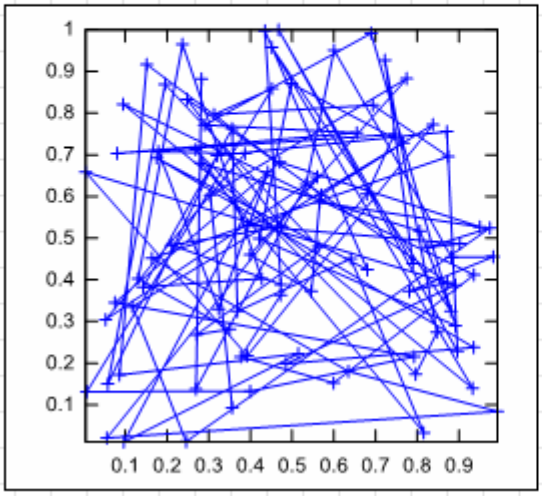
`impulses` Linien zur x-Achse oder zur xy-Ebene (bei 3D Punkten)

Bezeichnung	Kennzahl	Symbol
none	-1	keines
dot	0	·
plus	1	+
multiply	2	×
asterisk	3	★
square	4	□
filled_square	5	■
circle	6	∘
filled_circle	7	•
up_triangle	8	△
filled_up_triangle	9	▲
down_triangle	10	▽
filled_down_triangle	11	▼
diamant	12	◊
filled_diamant	13	♦

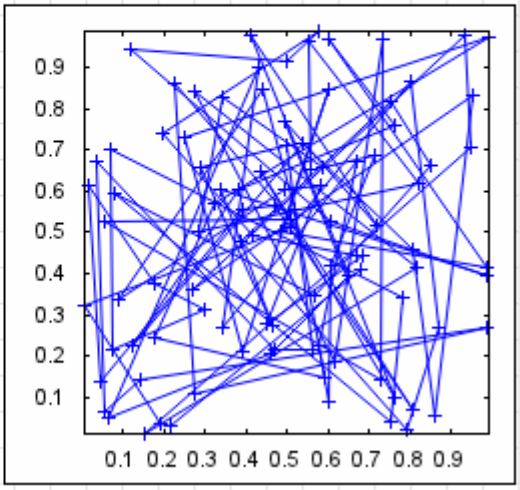
Tabelle 5.1.: Punktarten (Werte für `point_type`)

Die folgenden Diagramme zeigen verbundene Punkte mit zufälligen Koordinaten. Links die svg-Version und rechts die png-Version (nur durch Angabe eines Dateinamen mit Endung *png* erreichbar).

```
cm:=[points_joined=true, points(Random(100 ; 2)), proportional_axes=xy]
s:=[2 60, 2 40]
p1:=Draw2D(cm ; s)
p2:=Draw2D(cm ; "draw_points2.png" ; s)
```



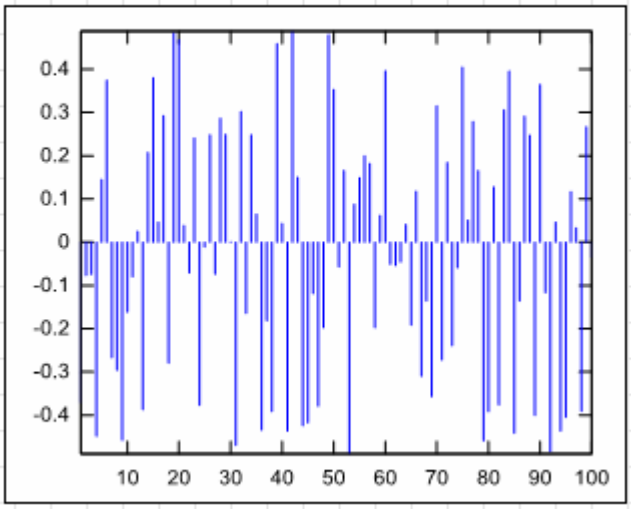
p1



p2

Punkte mit zufälligen y-Werten, als x-Wert wird die laufende Nummer verwendet.

```
p:=Draw2D([points_joined=impulses, points(Random(100 ; 1)-0,5)])
```



p

5.1.4. Funktionen 2D

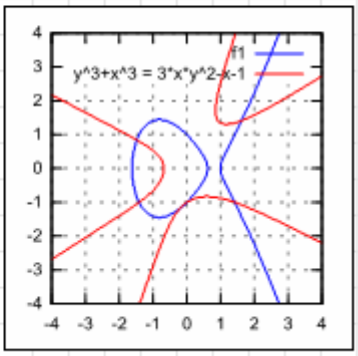
Die Funktion `explicit(f(x); x; xmin; xmax)` dient der Darstellung explizit gegebener Funktionen $y(x)$ innerhalb eines vorgegebenen Intervalls:

```
cmds:=[key="sin(x)"
       explicit(sin(x);x;-2·π;2·π)
       key="cos(x)"
       color=red
       explicit(cos(x);x;-2·π;2·π)
       grid=true]

p:=Draw2D(cmds;[400
                200])
```

p

Mit `implicit(f(x; y); x; xmin; xmax; y; ymin; ymax)` können implizit gegebene Funktionen dargestellt werden.

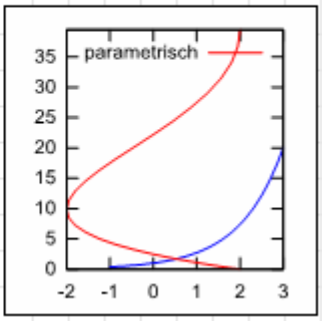
```
f1:=(y^2=x^3-2·x+1)        f2:=(x^3+y^3=3·x·y^2-x-1)

cm:=[key="f1"
     implicit(f1;x;-4;4;y;-4;4)
     key=string(f2)
     color=red
     implicit(f2;x;-4;4;y;-4;4)
     proportional_axes=xy
     grid=true]

p:=Draw2D(cm;[200
              200])
```



p

Parametrische Funktionen werden mit `parametric(x(t); y(t); t; tmin; tmax)` dargestellt.

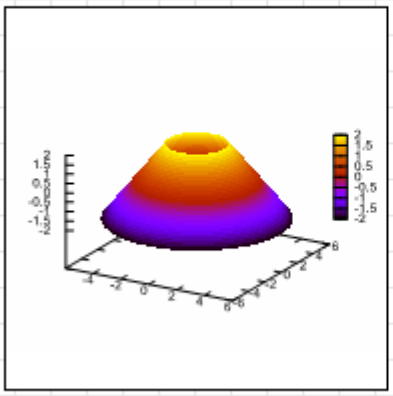
```
cm:=[explicit(e^x;x;-1;3)
     color=red
     key="parametrisch"
     parametric(2·cos(r);r^2;r;0;2·π)]

p:=Draw2D(cm;[160
              160])
```



5.1.5. Funktionen 3D

Die Funktion `cylindrical(r(z; φ); z; zmin; zmax; φ; φmin; φmax)` stellt Flächen dar, die als Funktion $\begin{cases} 60 \\ 30 \end{cases}$ der axialen Koordinate z und des Azimut-Winkels az gegeben sind.

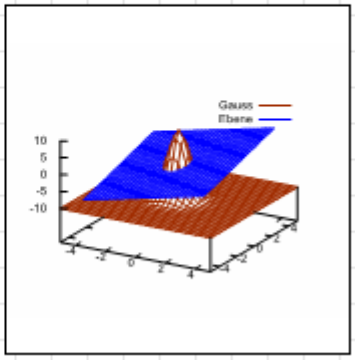
```
p:=Draw3D({enhanced3d=true
           cylindrical(4-z; z; -2; 2; az; 0; 2·π)})
```



```
p
```

Die Funktion `explicit(z(x; y); x; xmin; xmax; y; ymin; ymax)` dient der Darstellung explizit gegebener Funktionen $z(x, y)$ innerhalb eines vorgegebenen Intervalls:

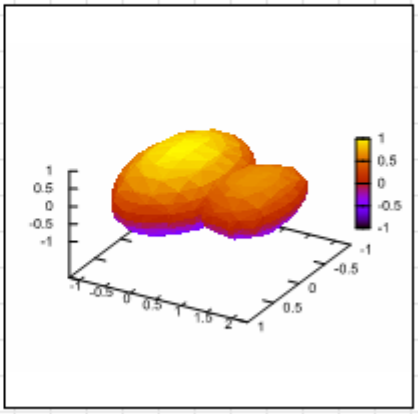
```
gauss:=explicit(20·e^(-x^2-y^2)-10; x; -5; 5; y; -5; 5)
ebene:=explicit(x+y; x; -4; 4; y; -4; 4)
cmds:={key="Gauss"
       color="#a02c00"
       gauss
       key="Ebene"
       color=blue
       ebene
       surface_hide=true
p:=Draw3D(cmds)
```




```
p
```

Mit `implicit(f(x; y; z); x; xmin; xmax; y; ymin; ymax; z; zmin; zmax)` können implizit gegebene Funktionen dargestellt werden:

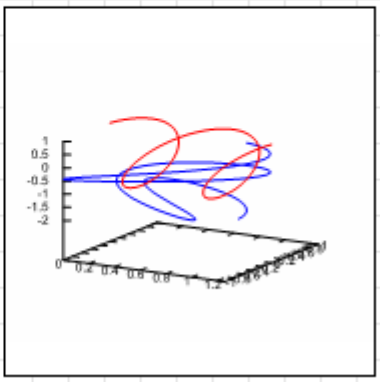
```
f:=((x^2+y^2+z^2-1)·(x^2+(y-1,5)^2+z^2-0,5)=0,015)

cm:={enhanced3d=true
     implicit(f;x;-1;1;y;-1,2;2,3;z;-1;1)
     view={50
           120
     cbtics=0,5

p:=Draw3D(cm)
```



p

Parametrische Kurven werden mit `parametric(x; y; z; t; tmin; tmax)` dargestellt:

```
p:=Draw3D({nticks=100
           parametric(cos(5·u)^2;sin(7·u);u-2;u;0;2)
           color=red
           parametric(0,1·t;sin(t);cos(t);t;0;4·π)
           view={70
                 30
```



p

Parametrische Flächen werden mit `parametric_surface(x; y; z; u; umin; umax; v; vmin; vmax)` dargestellt. Das folgende Beispiel ist von der Gnuplot-Beispiel-Seite inspiriert.

```
cm:=[user_preamble="set style fill transparent solid 0.65 border"
     "set view 120, 357, 1.75, 1.08"
     axis_3d=false
     colorbox=false
     enhanced3d=true
     surface_hide=true
     xu_grid=51
     yv_grid=51
     xtics=none
     ytics=none
     ztics=none
     wired_surface=true
     parametric_surface(fx; fy; fz; u; -4,5; 4,5; v; 0,05; π-0,05)]
```

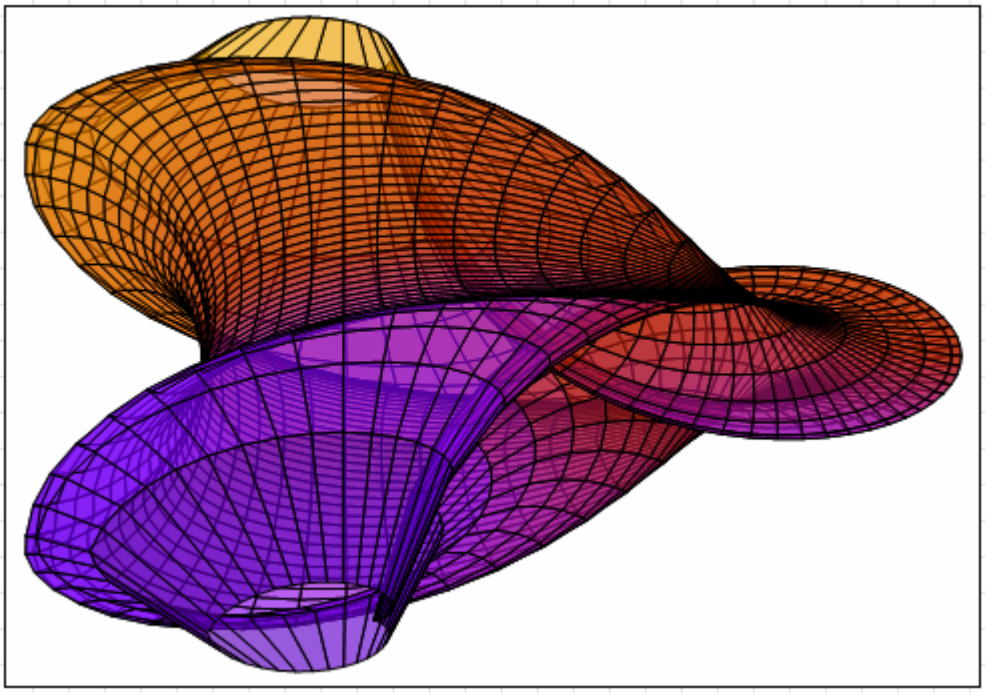
$$a := 1$$

$$fx := \frac{2 \cdot a \cdot (\cos(u) + u \cdot \sin(u)) \cdot \sin(v)}{1 + u^2 \cdot \sin(v)^2}$$

$$fy := \frac{2 \cdot a \cdot (\sin(u) - u \cdot \cos(u)) \cdot \sin(v)}{1 + u^2 \cdot \sin(v)^2}$$

$$fz := a \cdot \ln\left(\tan\left(\frac{v}{2}\right)\right) + \frac{2 \cdot \cos(v)}{1 + u^2 \cdot \sin(v)^2}$$

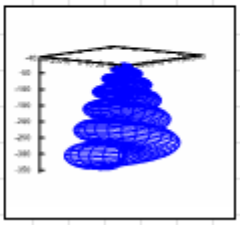
```
p:=Draw3D(cm; [1000
               700])
```



```
p
```

Hier noch eine parametrische Fläche aus dem Maxima-Handbuch:

```
fx:=u/2·cos(u)·(cos(v)+1)   fy:=u/2·sin(u)·(cos(v)+1)   fz:=u·sin(v)-((u+3)/8·π)^2-20
cm:=[user_preamble="set view 100, 320, 1, 1.5"
     xu_grid=150
     yv_grid=25
     surface_hide=true
     xyplane=0
     parametric_surface(fx; fy; fz; u; 0; 13·π; v; -π; π)]
p:=Draw3D(cm)
```



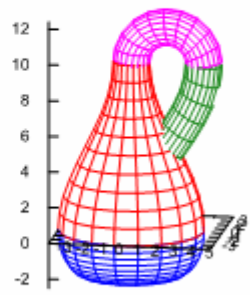
p

Mit `tube(x; y; z; r; p; min; max)` können Schlauchflächen durch die Lage der Mittellinie x, y, z und den Radius r als Funktionen eines Längenparameters p vorgegeben werden. Hier als Beispiel die Kleinsche Flasche. Zunächst werden die Funktionen definiert. Den einzelnen Abschnitten werden Farben zugeordnet, damit man sie später im Bild eindeutig identifizieren kann.

```
a:=2,5   b:=1,5   c:=2   d:=3
obj:=[color=blue
      tube(0; 0; (-a)·sin(t); a+b·cos(t); t; 0; π/2)
      tube(0; 0; (-a)·sin(t); a+b·cos(t); t; π/2; π)
      color=red
      tube(0; 0; d·t; a+b·cos(t); t; 0; π)
      color=forest_green
      tube(c-c·cos(t); 0; d·t; a-b; t; 0; π)
      color=magenta
      tube(c-c·cos(t); 0; d·π+c·sin(t); a-b; t; 0; π)]
cm:=[xu_grid=11
     yv_grid=20
     xyplane=0
     surface_hide=true
     proportional_axes=xyz
     obj
     view=[v1
           100-v2]]
p:=Draw3D(cm)
```

v1

v2



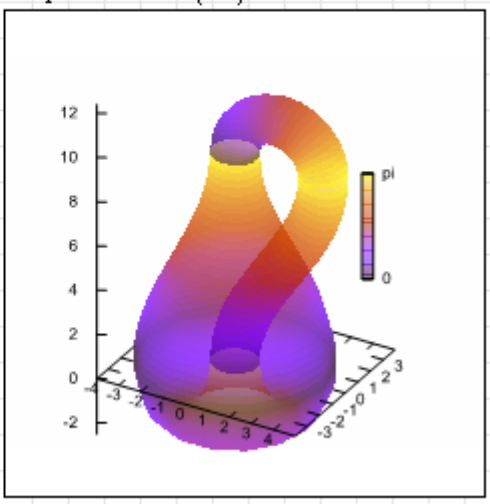
Darstellung als Gitternetz mit verdeckten Kanten

p

```
cm:= [ user_preamble = ["set style fill transparent solid 0.5 border"
                        "set view 60, 30, 1.4, 1.2"]
       xu_grid = 29
       yv_grid = 15
       enhanced3d = [t
                     t]
       xyplane = 0
       surface_hide = true
       proportional_axes = xyz
       cbtics = set([ "0"  ; "pi" ]
                    [ 0    ;  π   ])
       obj ]
```

Darstellung als transparente Fläche mit dem Längenparameter als Farbwert

```
p := Draw3D(cm)
```




```
p
```

5.1.6. Grafikobjekte

Dieser Abschnitt befindet sich im Aufbau...

`bars(b1; b2; b3...)` Säulendiagramm, die einzelnen Säulen sind durch Listen mit der Position x, der Höhe h und der Breite b gegeben:

$$\begin{cases} x \\ h \\ w \end{cases}$$

`elevation_grid(`$\underline{M}$`; `x_0`; `y_0`; `w`; `h`)` 3D-Darstellung einer $m \times n$ Matrix von z-Werten. M_{11} entspricht $x = x_0$ und $y = y_0 + h$, M_{mn} entspricht $x = x_0 + w$ und $y = y_0$.

`line_type` Linienart der Gitterlinien

`line_width` Linienbreite der Gitterlinien

`key` Legendeneintrag

`color` Linienfarbe

`wired_surface` (false) oder true: Gitternetz auf eingefärbten Flächen anzeigen oder nicht.

`ellipse(xc; yc; a; b; ang1; ang2;)` Ellipse, zentriert bei `xc`, `yc` mit den Halbachsen `a` und `b`, von Winkel `ang1` bis Winkel `ang2`

`nticks` Zahl der Punkte für die Randkurve

`transparent` ungefüllt oder gefüllt (hier kann keine Transparenz eingestellt werden)

`fill_color` Füllfarbe

`border` Rand zeichnen

`line_type` Linienart für den Rand

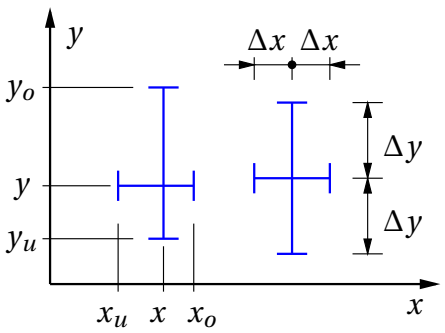
`line_width` Linienbreite für den Rand

`key` Legendeneintrag

`color` Linienfarbe

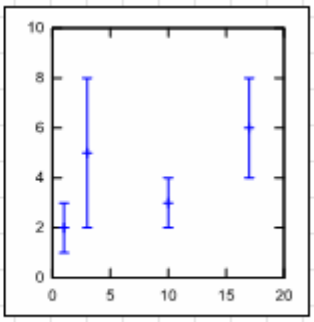
`errors(l1; l2;...)` Fehlerbalken. Abhängig vom Balkentyp sind die Argumente Listen der Form

error_type	Liste
x	$\begin{Bmatrix} x \\ y \\ \Delta x \end{Bmatrix}$ oder $\begin{Bmatrix} x \\ y \\ x_u \\ x_o \end{Bmatrix}$
y	$\begin{Bmatrix} x \\ y \\ \Delta y \end{Bmatrix}$ oder $\begin{Bmatrix} x \\ y \\ y_u \\ y_o \end{Bmatrix}$
xy,boxes	$\begin{Bmatrix} x \\ y \\ \Delta x \\ \Delta y \end{Bmatrix}$ oder $\begin{Bmatrix} x \\ y \\ x_u \\ x_o \\ y_u \\ y_o \end{Bmatrix}$

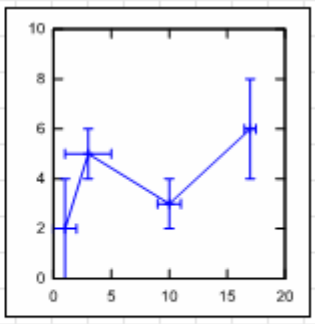


`error_type` Balkenart (`x`, `y`, `xy` oder `boxes`)

```
p := Draw2D( error_type = y,
             errors( args( [[1,2,1],[3,5,3],[10,3,1],[17,6,2]] ) ) : [200,200],
             xrange = [0,20],
             yrange = [0,10] )
```




```
p := Draw2D( error_type = xy,
             points_joined = true,
             errors( args( [[1,2,1,2],[3,5,2,1],[10,3,1,1],[17,6,1/2,2]] ) ) : [200,200],
             xrange = [0,20],
             yrange = [0,10] )
```



`explicit(expr; var; min; max)` explizit gegebene Funktionen darstellen. Das ist der Normalfall.

`histogram_description(data; option1; option2...)` Histogramm-Objekt. `data` ist eine Liste oder ein Vektor. Die Optionen müssen separat angegeben werden (können nicht zu Listen zusammengefasst werden).

`nclasses` (10) Klassenzahl des Histogramms oder eine Liste mit den Bereichsgrenzen und optional deren Anzahl:

$$\begin{cases} x_{\max} \\ x_{\max} \\ n_k \end{cases} \text{ oder } \begin{cases} x_{\max} \\ x_{\max} \end{cases}$$

`frequency` (`absolute`) Skalierung der Klassenwerte:

`absolute` Anzahl der Werte in der Klasse

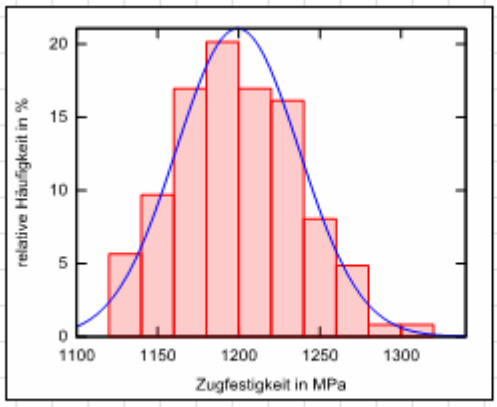
`relative` Relativer Anteil der Werte in der Klasse

`percent` Relativer Anteil in %

`density` Mittlere Verteilungsdichte (relativer Anteil durch Klassenbreite)

`htics` (`auto`), `endpoints`, `intervals` oder eine Liste mit Marken

```
cm:={ fill_density=0,2
    | histogram_description(D ; nclasses={x_min, x_max, n_k ; frequency=percent)_5
    | explicit(100·δ·pdf_Normal(x ; R_m_μ ; R_m_σ) ; x ; x_min ; x_max)
    | xlabel="Zugfestigkeit in MPa"
    | ylabel="relative Häufigkeit in %"
p:=Draw2D(cm)
```




```
p
```

Der Index an der Funktion histogram_description ist erforderlich, weil SMath nicht mit leeren Listen zurechtkommt.

`image()` Rasterbild

`implicit()` Implizit gegebene Funktionen darstellen

`ip_grid` Anzahl von Intervallen für die Funktionsabtastung in x- und y-Richtung

`ip_grid_in` Sekundäre Intervallteilung

`label(Liste)` Beschriftung, die Liste hat die Form

$$\begin{cases} \text{Text} \\ x \\ y \end{cases}$$

`label_alignment` Horizontale Ausrichtung (`center`, `left`, `right`)

`label_orientation horizontal` oder `vertical`

`parametric(x; y; t; tmin; tmax)` parametrisch gegebene Funktion in 2D. Die üblichen Optionen für Linienobjekte sind anwendbar.

`parametric(x; y; z; t; tmin; tmax)` parametrisch gegebene Funktion in 3D.

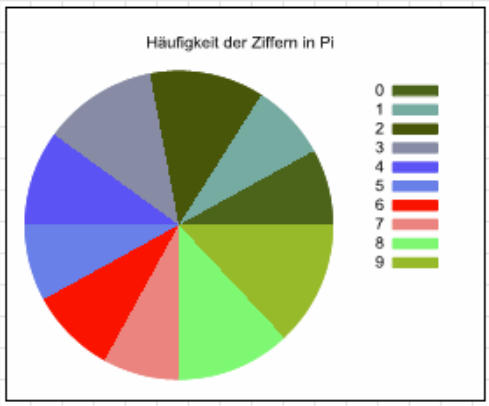
`piechart_description(data; option1; option2...)` Sektordiagramm-Objekt. `data` ist eine Liste oder ein Vektor. Die Optionen müssen separat angegeben werden (können nicht zu Listen zusammengefasst werden). Das Diagramm wird aus `ellipse()`-Objekten aufgabaut.

`sector_colors` (Zufallsfarben) Liste mit den Farben der Sektoren

`pie_center` ($\begin{cases} 0 \\ 0 \end{cases}$) Mittelpunkt

`pie_radius` (1) Sektorradius

```
msg:Maxima(load(descriptive))
S:Maxima(read_matrix(file_search("pidigits.data")))
p:Draw2D(title="Häufigkeit der Ziffern in Pi"
         piechart_description(S)_2
         proportional_axes=xy
         xrange={-1
                 1,8}
         axis_top=false
         axis_bottom=false
         axis_left=false
         axis_right=false
         xtics=false
         ytics=false)
```



p

`points()` Datenpunkte, optional mit Linien verbunden

`polygon(p1; p2; ...)` gefülltes Polygon. Die Argumente sind Koordinatenlisten für die einzelnen Punkte in der Form

$$\text{2D:} \begin{cases} x \\ y \end{cases} \quad \text{oder 3D:} \quad \begin{cases} x \\ y \\ z \end{cases}$$

Sie können mit der Maximafunktion `args()` bequem aus einer Koordinatenmatrix $\underline{M}$ erzeugt werden: `polygon(args(`$\underline{M}$`))`.

`transparent` ungefüllt oder gefüllt (hier kann keine Transparenz eingestellt werden)

`fill_color` Füllfarbe

`border` Rand zeichnen

`line_type` Linienart für den Rand

`line_width` Linienbreite für den Rand

`key` Legendeneintrag

`color` Linienfarbe

`quadrilateral()` Viereck

`rectangle()` Rechteck

`triangle()` Dreieck

`tube()` Schlauch, gegeben durch Mittellinie und Radius

`vector()` Pfeil

`head_angle()` Winkel zwischen Schaft und Spitze in °

`head_both()` ein (`false`) oder zwei Spitzen (`true`)

`head_length()` Spitzenlänge in Einheiten der x-Achse

`head_type()`

`unit_vectors()`

5.1.7. Optionen

Die Optionen für die Funktionen `Draw2D()` und `Draw3D()` sind als logische Gleichungen `Option=Wert` anzugeben.Genau wie die Grafikobjekte können sie beliebig in Listen verschachtelt werden und gelten immer für die nachfolgenden Grafikobjekte oder global für das Diagramm.

Option	(Voreinstellung): Bedeutung
`adapt_depth`	(10) Maximale Teilungstiefe der adaptiven Kurvenabtastung

`axis_3d`
 	
`axis_bottom/top`
`axis_left/right`
	(`true`): Achsen im 3D zeichnen
	`false`: keine Achse zeichnen
	analog für die Achsen im 2D

`azimuth`	(30) Drehwinkel um z in °
`background_color`	(`white`): Hintergrundfarbe
`border`	(`true`): Ränder von `polygon`, `ellipse` oder `rectangle` zeichnen
`cbrange`	(`auto`): Farbbereich (Liste mit min und max)
`cbtics`	(`auto`): Marken am Farbbalken
`color`	(`blue`): Farbe für Linien, Punkte, Kanten und Text, als Namen oder hexadezimal in der Form `"#rrggbb"`
`colorbox`	(`true`): Farblegende ohne Beschriftung zeichnen `false`: keine Farblegende zeichnen *text*: Farblegende mit Beschriftung *text* zeichnen
`contour`	(none) keine Höhenlinien `base`: Höhenlinien auf der xy-Ebene `surface`: Höhenlinien auf der Fläche `both`: Höhenlinien auf der xy-Ebene und auf der Fläche `map`: xy-Ebene und Blickrichtung senkrecht
`contour_levels`	(5) Zahl der Höhenlinien Dreier-Liste mit Minimum, Schritt und Maximum Satz (set) mit Werten

`data_file_name`	(`"data.gnuplot"`) Kurvendaten für Gnuplot
`delay`	(5) Zeitschritt in animierten gif-Dateien zwischen zwei Bildern
`dimensions`	(600x500) Liste mit Breite und Höhe
`draw_realpart`	(`true`) Realteil bei komplexen Funktionen darstellen.
`enhanced3d`	(`none`) keinen Farbverlauf verwenden, die Oberflächen werden als Gitter dargestellt (steuerbar mit `color` und `surface_hide`) $\begin{cases} f(x,y,z) \\ x \\ y \\ z \end{cases}$, $\begin{cases} f(x,y) \\ x \\ y \end{cases}$ Färbung anhand der Koordinaten, 2D-Fall für `explicit()` und `elevation_grid()` $\begin{cases} f(u,v) \\ u \\ v \end{cases}$, $\begin{cases} f(u) \\ u \end{cases}$ Färbung anhand der Objektparameter $\begin{cases} f(k) \\ k \end{cases}$ Färbung anhand der Reihenfolge (bei Punkten, `points()`) `true`: entspricht Färbung gemäß der z-Koordinate
`error_type`	(`y`) oder `x`, `xy` oder `boxes`: Fehlerbalkenformat
`file_name`	(`"maxima_out"`) mit Endung gemäß Ausgabeformat
`fill_color`	(`red`) Füllfarbe für Polygone und explizite Funktionen $y(x)$ im 2D
`fill_density`	(0) Deckungsgrad der Füllfarbe in `bars()`-Objekten, zwischen 0 und 1
`filled_func`	(`false`) explizite Funktionen $y(x)$ werden nicht gefüllt `true`: Füllung bis zur Unterkante des Diagramms $g(x)$ Füllung zwischen $g(x)$ und $y(x)$
`gnuplot_file_name`	(`"maxout.gnuplot"`) Kommandodatei für Gnuplot
`gnuplot_term`	() Ausgabe-Format-Spezifikation (Text nach „set terminal”)
`gnuplot_out_file`	(maxplot.xxx) Ausgabedateiname
`grid`	(`false`) kein Gitterraster zeichnen `true`: Gitter zeichnen
`head_angle`	(45) Pfeilwinkel an Vektoren (halber Öffnungswinkel in °)
`head_both`	(false) ob bei `vector()`-Obkjekten zwei Pfeilspitzen gezeichnet werden.
`head_length`	(2) Länge der Pfeilspitze in Einheiten der x-Achse
`head_type`	(`filled`) ▶, `empty` ▷ oder `nonfilled` >
`ip_grid`	$\begin{cases} 50 \\ 50 \end{cases}$ Primäres Gitter für die Darstellung impliziter Funktionen
`ip_grid_in`	$\begin{cases} 5 \\ 5 \end{cases}$ Sekundäres Gitter für die Darstellung impliziter Funktionen

`key`	(`""`) Legendeneintrag für das nachfolgende Grafikobjekt.
`label_alignment`	(`center`), `left` oder `right`: Ausrichtung von Texten
`label_orientation`	(`horizontal`) oder `vertical`: Ausrichtung von Texten
`line_type`	(`solid`) oder `dots`: Linienart
`line_width`	(1) Linienbreite, > 0
`logcb`	(`false`) oder `true`: logarithmische Farbskale
`logx/y/z`	(`false`) oder `true`: logarithmische x-, y- oder z-Achse
`nticks`	(29): Punktezahl für die Kurvenabtastung
`palette`	(color) Standard-Farbpalette `gray`: Graustufen $\left\{\begin{array}{l}\text{Farbe}_1\\ \text{Farbe}_2\\ \quad\vdots\\ \text{Farbe}_n\end{array}\right.$ `false`: Flächen nicht färben
`point_size`	(1) Größe von `points()`-Symbolen
`point_type`	(1) Symbol für `points()`
`points_joined`	(`false`), `true` oder `impulses`: Verbindungsart für `points()`
`proportional_axes`	(`none`), `xy` oder `xyz`: Achsen mit gleicher Skalierung
`surface_hide`	(`false`) oder `true`: Ausblendung verdeckter Flächen bei Linienrasterdarstellung
`terminal`	(`screen`): Separates Gnuplot-Fenster öffnet sich `png, jpg, gif, eps, svg, pdf, animated_gif`
`title`	(`""`) Überschrift für das Diagramm
`transform`	(`none`) Koordinatentransformation $\left\{\begin{array}{l}f_1(x,y)\\ f_2(x,y)\\ x\\ y\end{array}\right.$ für 2D oder $\left\{\begin{array}{l}f_1(x,y,z)\\ f_2(x,y,z)\\ f_3(x,y,z)\\ x\\ y\\ z\end{array}\right.$ für 3D
`transparent`	(`false`) oder `true`: ob die Rechtecke, Polygone oder Ellipsen gefüllt werden sollen (bei `false` werden sie gefüllt)
`tube_extremes`	($\left\{\begin{array}{l}\text{open}\\ \text{open}\end{array}\right.$) `open` oder `closed`: Endscheiben von `tube()`-Objekten
`unit_vectors`	(`false`) oder `true`: Vektoren in originaler Länge oder als Einheitsvektoren zeichnen.
`user_preamble`	(`""`) Zeichenkette oder Liste von Zeichenketten, die unmittelbar vor dem eigentlichen Diagrammbefehl an Gnuplot übergeben werden. Beispiele sind in Abschnitt 5.1.7.2 angegeben.
`view`	($\left\{\begin{array}{l}60\\ 30\end{array}\right.$) Blickrichtung

`wired_surface`	(`false`) oder `true`: Gitternetz auf eingefärbten Flächen anzeigen oder nicht.
`x/y/z_voxel`	(10) Bereichsteilungen bei implizit gegebenen Flächen
`x/y/zaxis`	(`false`) oder `true`: Anzeige der entsprechenden Achse
`x/y/zaxis_color`	(`black`) Farbe der entsprechenden Achse
`x/yaxis_secondary`	(`false`) oder `true`: Nutzung der zweiten x- oder y-Achse (oben oder rechts) für die nachfolgenden Grafikobjekte
`x/y/zaxis_type`	(`dots`) oder `solid`: Linienart der entsprechenden Achse
`x/y/zaxis_width`	(1) Linienbreite der entsprechenden Achse
`x/y/zlabel`	(`""`) Achsenbeschriftung
`x/y/zrange`	(`auto`) oder $\left\{ \begin{array}{l} \text{min} \\ \text{max} \end{array} \right.$: Achsenbereich
`x/yrange_secondary`	(`auto`) oder $\left\{ \begin{array}{l} \text{min} \\ \text{max} \end{array} \right.$: Achsenbereich der Zweitachsen
`x/y/ztics` `x/ytics_secondary`	(`auto`) automatische Achsmarken none: keine Achsmarken Zahl: Schrittweite der Achsmarken $\left\{ \begin{array}{c} \text{min} \\ \text{Schritt} \\ \text{max} \end{array} \right.$ Satz (set) von Zahlen: Explizite Vorgabe der Achsmarkenpositionen Satz (set) von Listen $\left\{ \begin{array}{l} \text{"text"} \\ \text{pos} \end{array} \right.$
`x/y/ztics_axis` `x/ytics_secondary_axis`	(`false`): Achsmarken am Rand `true`: Achsmarken an der Achse
`x/y/ztics_rotate`	(`false`) oder `true`: Drehen der Achsmarkenbeschriftungen um 90°
`x/ytics_rotate_secondary`	(`false`) oder `true`: Drehen der Achsmarkenbeschriftungen um 90°
`xu_grid`	(30) Werteanzahl in die erste Koordinatenrichtung
`xyplane`	(`false`): xy-Koordinatenebene automatisch plazieren Zahl: bei vorgegebenem z-Wert plazieren
`yv_grid`	(30) Werteanzahl in die zweite Koordinatenrichtung

5.1.7.1. Farbspezifikation

Die Farbwerte für die Optionen `background_color`, `color`, `palette` und `x/y/zaxis_color` können als Wort oder als RGB-Wert (Zeichenkette) im Format `"#rrggbb"` angegeben werden.

Die folgenden Farben sind vordefiniert. Die Wörter können als Zeichenketten oder direkt als Variablenname angegeben werden, solange diese Namen im Rechenblatt nicht definiert werden.

```
white black gray0 grey0 gray10 grey10 gray20 grey20 gray30
grey30 gray40 grey40 gray50 grey50 gray60 grey60 gray70
grey70 gray80 grey80 gray90 grey90 gray100 grey100 gray grey
light_gray light_grey dark_gray dark_grey red light_red dark_red
```

```
yellow light_yellow dark_yellow green light_green dark_green
spring_green forest_green sea_green blue light_blue dark_blue
midnight_blue navy medium_blue royalblue skyblue cyan light_cyan
dark_cyan magenta light_magenta dark_magenta turquoise
light_turquoise dark_turquoise pink light_pink dark_pink
coral light_coral orange_red salmon light_salmon dark_salmon
aquamarine khaki dark_khaki goldenrod light_goldenrod
dark_goldenrod gold beige brown orange dark_orange violet
dark_violet plum purple
```

5.1.7.2. Nützliche Befehle für `user_preamble`

Plazierung der Legende

```
set key top|bottom left|right
```

Skalierung von 3D-Diagrammen

```
set view zenit azimut plotscale zscale
```

5.2. 2D-Diagramme (Standard)

2D-Diagrammbereiche werden erzeugt mit

- @
- Hauptmenü> Einfügen> Grafik> Zweidimensional

Diagrammbereiche können vier Arten von Ausdrücken anzeigen:

- Funktionen, als Argument wird ein dimensionsloses x bereitgestellt, als Ergebnis wird eine dimensionslose Zahl erwartet.
- 2-spaltige Matrizen, die einen Polygonzug definieren ($x\,y$)
- 3-spaltige Matrizen, die zeilenweise einen Text (Zeichenkette) anzeigen (x y Text). Größe und Farbe haben dann die voreingestellten Standardwerte.
- 5-spaltige Matrizen, die zeilenweise einen Text anzeigen (x y Text Größe Farbe)

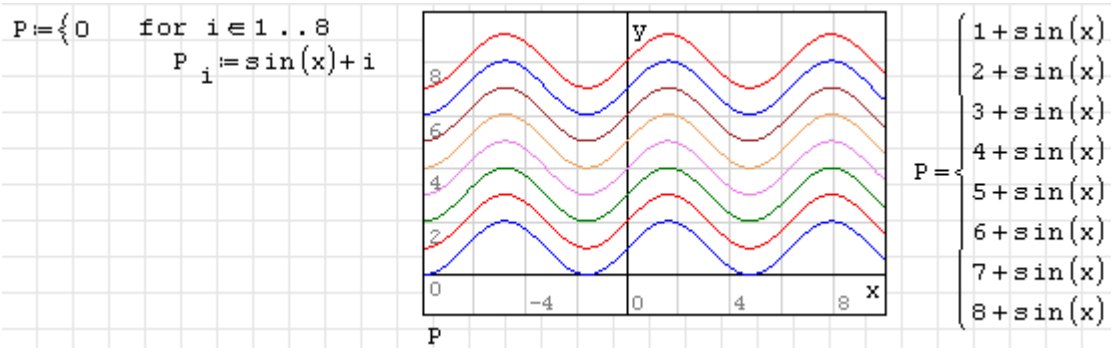
Alle zu plottenden Ausdrücke können mit der Variable t parametriert sein. Animierte Plots werden erzeugt, indem die Variable t die Werte eines beliebigen Vektors durchläuft, der im Rechenblatt definiert ist.

Wenn mehrere Ausdrücke in einem Bild dargestellt werden sollen, sind diese in einer Liste zu kombinieren.

5.2.1. Funktionen

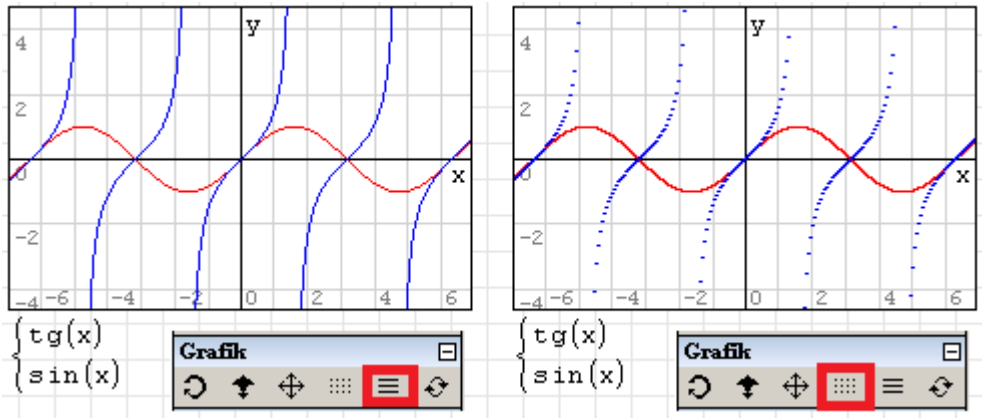
Funktionen werden als Polygonzug dargestellt, dessen x-Werte gleichmäßig über den Wertebereiche (sichtbarer Achsenbereich) verteilt sind und dessen y-Werte durch Aufruf der Funktion an diesen x-Werten berechnet werden.

Um mehrere Funktionen gleichzeitig darzustellen, ordnet man sie in einer Liste an. Für die Linien werden sechs Farben zyklisch verwendet.



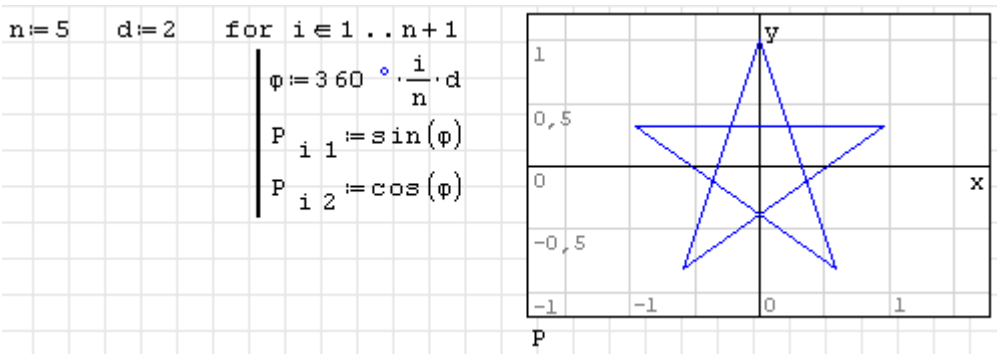
Leider gibt es keine Legendenfunktion. Man kann sich mit farbigen Textbereichen behelfen, dies ist allerdings fehleranfällige Handarbeit.

Das Aussehen der Graphen kann man durch Umschalten zwischen Punktdarstellung und Liniendarstellung im Panel „Grafik" der Seitenleiste beeinflussen:



5.2.2. Polygonzüge

SMath kann Polygonzüge darstellen, die als zweispaltige Matrizen gegeben sind, jede Zeile steht für einen Punkt. Die 1. Spalte enthält die x-Werte, die zweite Spalte die y-Werte.



$$P = \begin{pmatrix} 0{,}588 & -0{,}809 \\ -0{,}951 & 0{,}309 \\ 0{,}951 & 0{,}309 \\ -0{,}588 & -0{,}809 \\ -4{,}898 \cdot 10^{-16} & 1 \\ 0{,}588 & -0{,}809 \end{pmatrix}$$

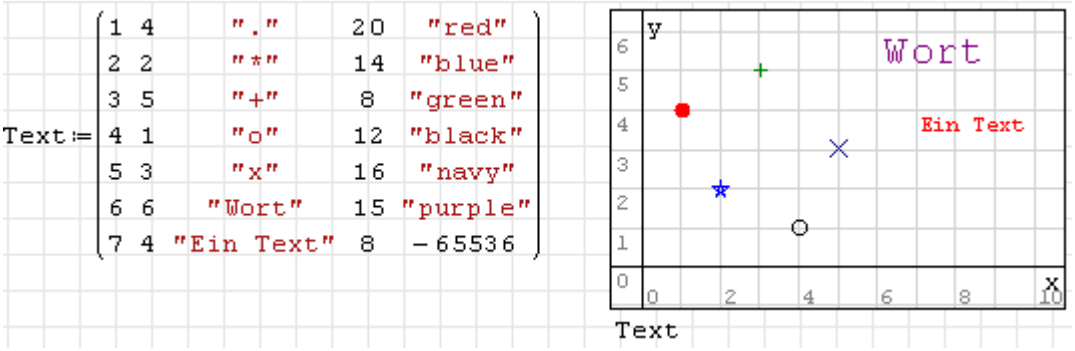
5.2.3. Text

Einzelne Zeichen oder Textzeilen können im Plot frei plaziert werden. Jede Zeile einer fünfspaltigen Matrix definiert einen Texteintrag:

$$[\ x \quad y \quad \text{Text} \quad s \quad \text{Farbe}\ ]$$

x, y	Koordinaten
Text	darzustellender Text (als Zeichenkette anzugeben)
s	Schriftgröße (Einheit Punkt)
Farbe	Textfarbe (als Zeichenkette anzugeben)

Die Koordinaten geben die Position der linken oberen Ecke des Textfeldes an. Die Zeichen `. * + o x` werden bei x, y zentriert, wenn sie einzeln vorkommen. Damit können sie auf einfache Weise als Punktsymbole in Plots verwendet werden.



5.2.4. Farben

Es gibt mehrere Wege, Farben für Textobjekte in Diagrammen zu spezifizieren:

1. Zeichenkette mit HTML-Farbnamen, z.B. `"red"`
2. Zeichenkette mit RGB-Hexcode, z.B. `"#FF0000"`. Je zwei Bytes codieren die Werte R, G und B.
3. Zeichenkette mit αRGB-Hexcode, z.B. `"880000FF"`. Je zwei Bytes codieren die Werte α (Deckkraft), R, G und B.
4. Dezimale 8-Byte-Ganzzahlen entsprechend den Hexcodes. Bei 6-Byte-Codes wird $\alpha = 1$ (`"FF"`) ergänzt. Auf diese Weise können Farbwerte auch berechnet werden.

SMath kennt die HTML-Farben, diese sind in Abbildung 5.3 dargestellt. Die Namen werden intern von Leerzeichen befreit und in Kleinbuchstaben umgewandelt, `"darkblue"`, `"DarkBlue"`, `"DARKBLUE"` und `"dark blue"` bezeichnen also alle die gleiche Farbe.

Das folgende SMath-Blatt[1] ordnet die Farbnamen jeweils passend eingefärbt in einer rechteckigen Matrix (Spaltenzahl c) an.

[1] Die Funktion `Floor()` ist im Plugin *Statistical Tools* enthalten, `importData.ODS()` im Plugin *Custom Functions*

```
F := importData_ODS("Farben.ods" ; "Farben" ; "A1" ; "A140")

c := 4   P := ⟨0

for i ∈ 1 .. length(F)        P_1 = (1 0 "aliceblue" 10 "aliceblue")
    x := mod(i ; c)
    y := -Floor(i / c)
    P_i := (x y F_i 10 F_i)
```

Abbildung 5.3 zeigt einen Plot der Liste P. Das Gitter und die Achsen wurden in diesem Bild ausgeblendet.

Abbildung 5.3.: Vordefinierte Farben für die Darstellung von Symbolen und Texten in 2D-Plots. Die Liste ist dem Wiki-Eintrag zu Grafik (englisch) entnommen. Die dortige Liste wurde in eine Textdatei kopiert. Leider gibt es keine Möglichkeit, Zeichenketten aus Textdateien einzulesen, die Funktion `importData()` kann nur Zahlenwerte lesen. Daher musste die Textdatei in *Farben.csv* umbenannt, mit Excel geöffnet und als ods-Datei wieder abgespeichert werden.

Umfangreicher Wiki-Eintrag zu Grafik (englisch)

5.2.5. Achsengrenzen ändern

Mauszeiger auf Diagrammbereich:

- Ziehen: Achsen verschieben
- Scrollen: Achsen skalieren
- ⇑ -Scroll: nur x-Achse skalieren
- Strg -Scroll: nur y-Achse skalieren

5.2.6. Animationen steuern

Wenn der darzustellende Ausdruck die Variable `t` enthält, kann SMath dieser nacheinander die Werte aus einer vorhandenen Liste (Spaltenvektor) zuweisen und die dadurch entstehende Bildsequenz als Animation abspielen.

Die Werteliste muss als Variable definiert sein. Im Kontextmenü (rechte Maustaste) des Plots kann unter „Animation" die Liste gewählt werden. Angeboten wird alles, was definiert ist.

Ebenfalls im Kontextmenü kann man unter „Bildfrequenz" die Grundeinstellungen für die Animation machen:

- Einstellung der Bildfrequenz (auf einzelne, vordefinierte Werte)
- Animationsmodus (unbegrenzt, keine Animation, einmalig)
  - unbegrenzt: Die Bildfolge wird beliebig oft wiederholt
  - einmalig: Die Bildfolge wird nur einmal wiederholt (nach jeder Neuberechnung), sie kann also mit  neu gestartet werden.
  - keine Animation: man kann mit Alt -→ und Alt -← vorwärts und rückwärts einzeln durch die Bilder schalten.

Ein Anwendungsbeispiel mit Erläuterungen zur Erstellung der Animation finden Sie in Abschnitt 10.5.

Video-Anleitung

5.3. Ebene Polygone

Ebene Polygone sind Grafikobjekte, die in SMath als speziell formatierte Matrizen dargestellt werden. Funktionen zum Umgang mit solchen Objekten werden vom Plugin *GPCPlugin* bereitgestellt. Die Installation ist in Abschnitt 9.4.6 beschrieben.

Das Plugin bindet eine Bibliothek GPC der Uni Manchester für 2D-Geometrieoperationen mit Polygonen (stückweise geradlinig berandete Flächen) ein. Die Bibliothek ist frei nutzbar für nichtkommerzielle Zwecke. Lizenzinformationen finden Sie hier. Der Leistungsumfang:

- Geometrieoperationen: Differenz, Schnittmenge, Exklusives Oder und Vereinigung
- Flächen können durch mehrere Randkonturen gebildet werden.
- Die Randpunkte können mit oder gegen den Uhrzeigersinn angegeben werden.

- Ränder können konvex, konkav oder selbstüberschneidend sein.
- Ränder können geschachtelt werden (d.h. Flächen können Löcher haben).
- Ergebnisformat als Randkurve (contour) oder Dreieckskette (triangle strip)
- Außen- und Innenränder im Ergebnis unterscheidbar
- Robust gegen doppelte Punkte und entartete Gebiete.

Polygone werden als Matrizen dargestellt:

$$\begin{bmatrix} r & p_1 & \dots & p_r & x_{11} & \dots & x_{1p_1} & \dots & x_{r1} & \dots & x_{rp_r} \\ p & f_1 & \dots & f_r & y_{11} & \dots & y_{1p_1} & \dots & y_{r1} & \dots & y_{rp_r} \end{bmatrix}$$

r	Zahl der Randkurven
p	Gesamtzahl der Eckpunkte
p_i	Zahl der Eckpunkte in Randkurve i
f_i	Schalter für Randkurve i: 1 = Innenrand (schneidet ein Loch), 0 = Außenrand
x_{ij}	x-Koordinate des j-ten Punktes von Randkurve i
y_{ij}	y-Koordinate des j-ten Punktes von Randkurve i

Die Matrix hat demzufolge $1 + r + \sum p_i$ Spalten.

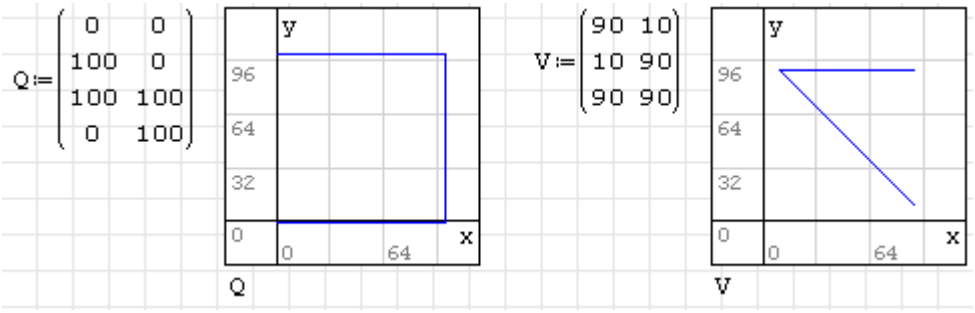
Die gleiche innere Darstellung wird für Dreiecksketten verwendet, dabei ist r die Zahl der Dreiecksketten und f_i ist stets gleich Null.

Beispiele (im russischen SMath-Forum)

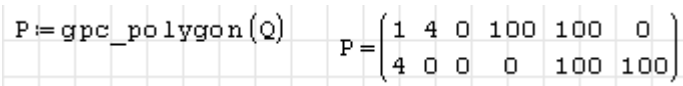
5.3.1. Polygon-Objekte erzeugen

Polygon-Objekte können aus Koordinatenmatrizen erzeugt werden, die die Randpunktfolge darstellen (Zeilen: Knoten, Spalte 1: x, Spalte 2: y). Die Randpunktfolge kann, muss aber nicht geschlossen sein.

Gegeben sind beispielsweise zwei Matrizen Q und V. Q enthält 4 Spalten und stellt ein Quadrat dar, V stellt die drei Ecken eines Dreiecks dar, das vollständig in das Quadrat hineinpasst:



Derartige Koordinatenlisten kann man mit `gpc_polygon()` zu einem Polygon-Objekt machen. Spalte 1 der Ergebnismatrix besagt, dass es einen Rand und 4 Punkte im ganzen Polygon gibt. Spalte 2 gibt an, dass der erste Rand 4 Punkte hat und ein Außenrand ist (Kennzahl 0, Innenrand wäre Kennzahl 1). Die weiteren Spalten sind die x, y-Koordinaten der jeweiligen Punkte.



Fügen wir nun das Dreieck als Innenrand hinzu. Die Funktion `gpc_add_contour(Polygon; Punktematrix; Kennzahl)` fügt je nach Kennzahl einen Außen- oder einen Innenrand hinzu.

```
T:=gpc_add_contour(P ; V ; 1)
```

$$T=\begin{pmatrix}2 & 4 & 3 & 0 & 100 & 100 & 0 & 90 & 10 & 90\\ 7 & 0 & 1 & 0 & 0 & 100 & 100 & 10 & 90 & 90\end{pmatrix}$$

Das Polygon T besteht nun aus 2 Rändern mit insgesamt 7 Punkten. Der erste Rand hat 4 Punkte und ist ein Außenrand (das erkennt man an Spalte 2), der zweite Rand (Info aus Spalte 3) enthält 3 Punkte und ist ein Innenrand. Es folgen die Punkte in der Reihenfolge der Ränder, also erst die vier Punkte des Außenrandes und dann die drei Punkte des Innenrandes.

5.3.2. Polygon-Objekte darstellen

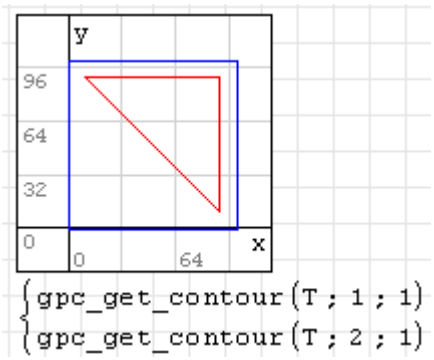
Das Ergebnis von Polygon-Operationen will man natürlich auch darstellen können. Im Plugin *GPCPlugin* gibt es die Funktion `gpc_get_contour(Polygon; Rand; Schalter)`, die einen beliebigen Rand wieder in eine plotbare Koordinatenmatrix umwandelt. Ist der Schalter 0, wird ein offener Rand erzeugt, ist er 1, wird der Rand geschlossen, d.h. der erste ist gleich dem letzten Punkt.

```
Randpunkte extrahieren
```

$$\text{gpc\_get\_contour}(T\ ;\ 2\ ;\ 0)=\begin{pmatrix}90 & 10\\ 10 & 90\\ 90 & 90\end{pmatrix}$$

```
Randpunkte für Plot extrahieren
```

$$\text{gpc\_get\_contour}(T\ ;\ 2\ ;\ 1)=\begin{pmatrix}90 & 10\\ 10 & 90\\ 90 & 90\\ 90 & 10\end{pmatrix}$$



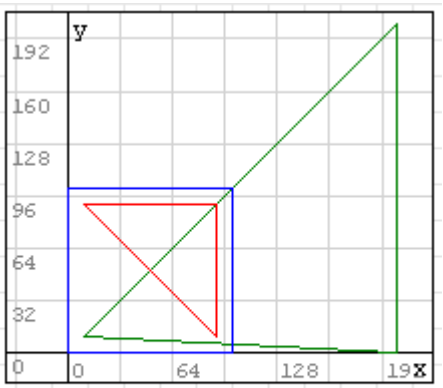
$$\begin{cases}\text{gpc\_get\_contour}(T\ ;\ 1\ ;\ 1)\\ \text{gpc\_get\_contour}(T\ ;\ 2\ ;\ 1)\end{cases}$$

5.3.3. Polygon-Operationen

Erzeugen wir zunächst ein weiteres Polygon N, welches beide Ränder von T schneidet und sich auch außerhalb von T erstreckt:

```
N:=gpc_polygon(...)
```

$$N=\begin{pmatrix}1 & 3 & 10 & 200 & 200\\ 3 & 0 & 10 & 200 & 0\end{pmatrix}$$




```
⎛gpc_get_contour(T ; 1 ; 1)
⎜gpc_get_contour(T ; 2 ; 1)
⎝gpc_get_contour(N ; 1 ; 1)
```

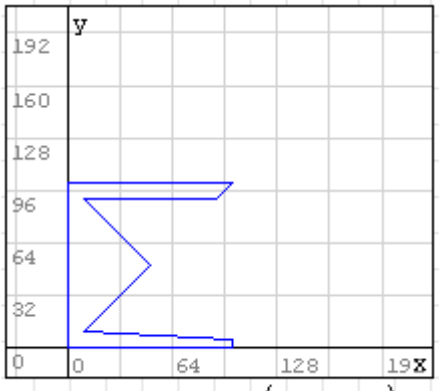
Polygon-Operationen werden mit der Funktion `gpc_clip(Kennzahl; Polygon1; Polygon2)` ausgeführt. Die Kennzahl steht für die auszuführende Operation:

Kennzahl	Operation
0	Differenz
1	Schnittmenge
2	Exklusives Oder
3	Vereinigung

Differenz: Polygon N wird von T abgezogen. Ergebnis sind die Flächen, die in T, aber nicht in N vorhanden sind.

```
M:=gpc_clip(0 ; T ; N)
```

$$M=\begin{pmatrix} 1 & 9 & 90 & 10 & 50 & 10 & 100 & 100 & 0 & 0 & 100 \\ 9 & 0 & 90 & 90 & 50 & 10 & 5{,}2632 & 0 & 0 & 100 & 100 \end{pmatrix}$$




```
gpc_get_contour(M ; 1 ; 1)
```

Schnittmenge: Ergebnis ist die gemeinsame Fläche beider Polygone.

```
M:=gpc_clip(1 ; T ; N)
```

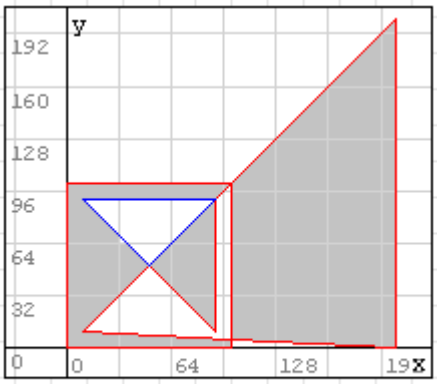
$$M=\begin{pmatrix} 1 & 6 & 100 & 10 & 50 & 90 & 90 & 100 \\ 6 & 0 & 5{,}2632 & 10 & 50 & 10 & 90 & 100 \end{pmatrix}$$

gpc_get_contour(M ; 1 ; 1)

Exklusives Oder: Ergebnis sind die Flächen, die nur in einem der beiden Polygone vorkommen. Im Diagramm sind die Gebiete grau gekennzeichnet (manuelle Nachbearbeitung)

M := gpc_clip(2 ; T ; N)

$$M = \begin{pmatrix} 2 & 3 & 13 & 90 & 10 & 50 & 200 & 100 & 100 & 0 & 0 & 100 & 90 & 90 & 50 & 10 & 100 & 100 & 200 \\ 16 & 1 & 0 & 90 & 90 & 50 & 0 & 5,2632 & 0 & 0 & 100 & 100 & 90 & 10 & 50 & 10 & 5,2632 & 100 & 200 \end{pmatrix}$$

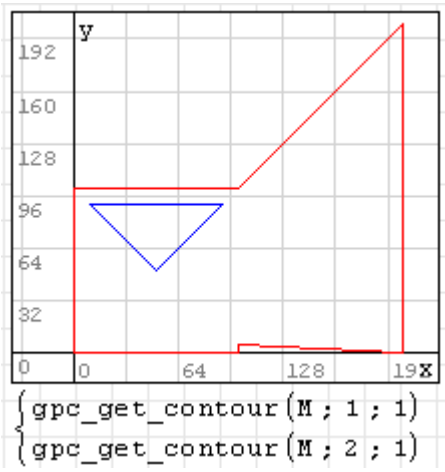


gpc_get_contour(M ; 1 ; 1)
gpc_get_contour(M ; 2 ; 1)

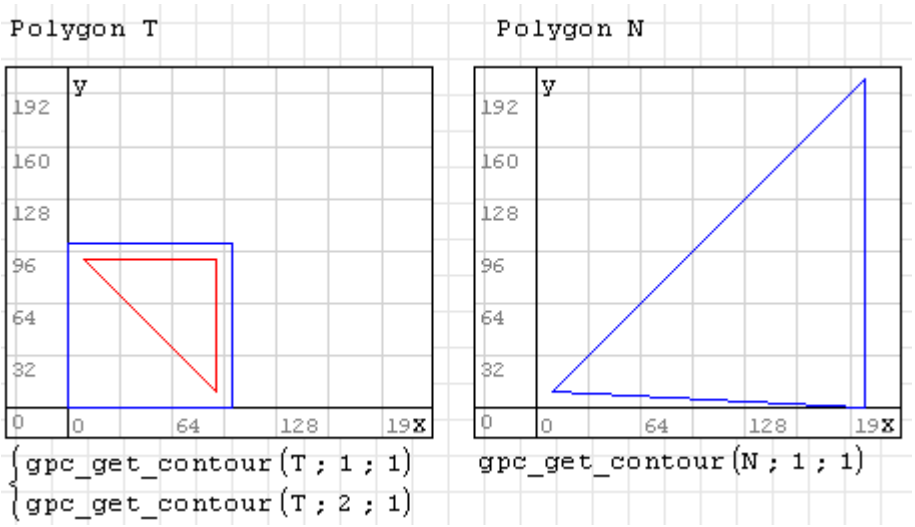
Vereinigung: Das sind die Flächen, die in mindestens einem der beiden Polygone vorhanden sind.

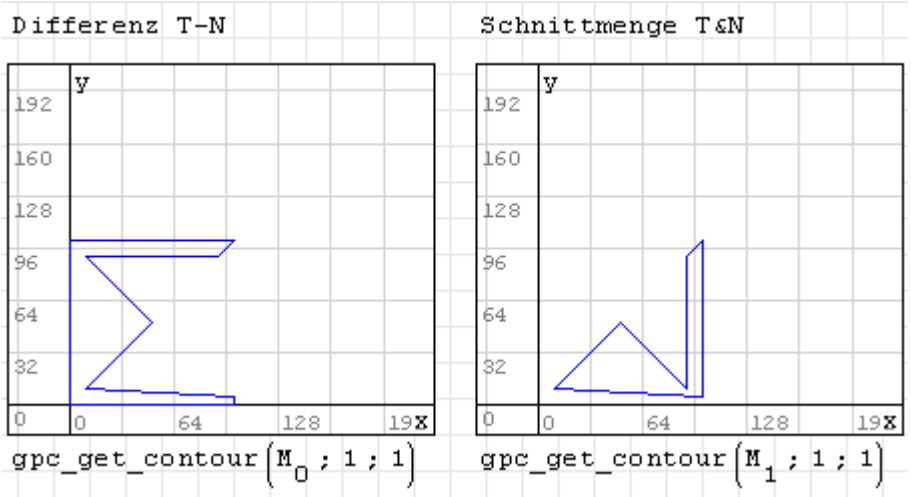
M := gpc_clip(3 ; T ; N)

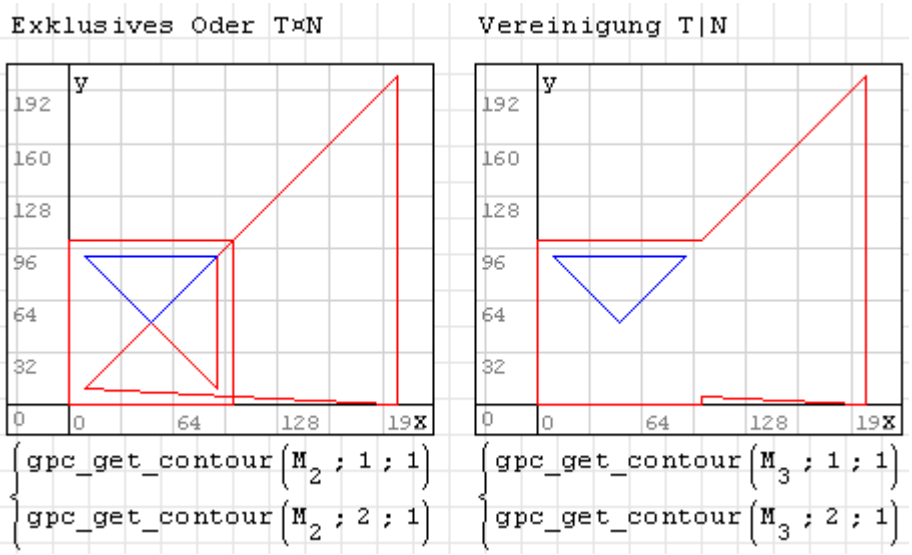
$$M = \begin{pmatrix} 2 & 3 & 7 & 90 & 10 & 50 & 200 & 100 & 100 & 0 & 0 & 100 & 200 \\ 10 & 1 & 0 & 90 & 90 & 50 & 0 & 5,2632 & 0 & 0 & 100 & 100 & 200 \end{pmatrix}$$



Hier noch mal zur Übersicht die Diagramme zusammengefasst:

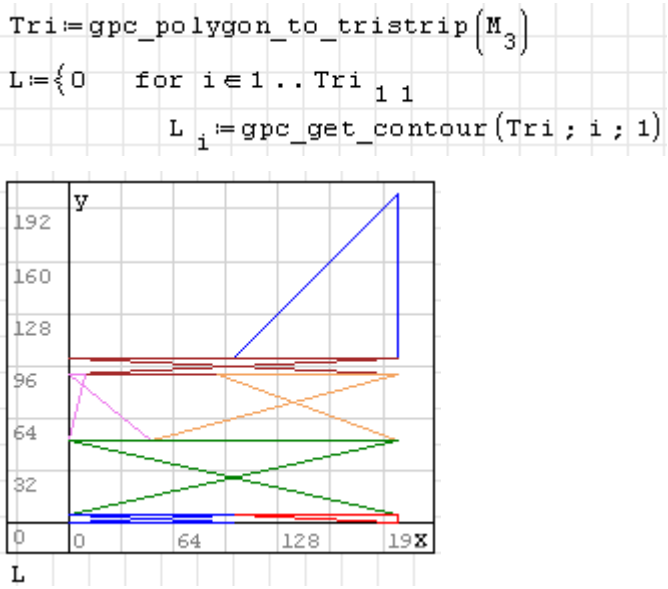






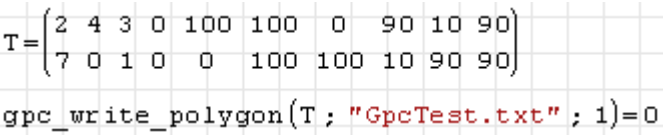
Dreieckszerlegung: Die Funktion `gpc_polygon_to_tristrip()` zerlegt die Polygonfläche in Dreiecksketten, also Linienzüge, bei denen zwei aufeinanderfolgende Geradenabschnitte immer ein Dreieck bilden. Zur Funktion `gpc_clip()` gibt es die Entsprechung `gpc_tristrip_clip(Kennzahl; Polygon1; Polygon2)`, die gleich Dreiecksketten liefert.

Das Matrixformat ist das gleiche wie bei der Randdarstellung, nur sind die Linienzüge nicht mehr als Rand, sondern als Dreieckskette zu interpretieren.



5.3.4. Polygon-Objekte lesen und schreiben

Das Plugin bietet auch zwei Funktionen zum Lesen und schreiben. Das letzte Argument ist jeweils der Schalter für die Randkennzahl. Steht er auf 1, wird die Randkennzahl (innerer oder äußerer Rand) mitgeschrieben und gelesen, ansonsten nicht. Verwenden Sie nur einen anderen Wert als 1, wenn Sie genau wissen, warum.



```
R:=gpc_read_polygon("GpcTest.txt"; 1)
```

$$R=\begin{pmatrix}2 & 4 & 3 & 0 & 100 & 100 & 0 & 90 & 10 & 90\\ 7 & 0 & 1 & 0 & 0 & 100 & 100 & 10 & 90 & 90\end{pmatrix}$$

Der Dateiname kann absolut oder relativ (bezogen auf das aktuelle Verzeichnis) sein.

5.4. Der Modeller-Bereich

Der Modeller-Bereich ist ein interaktiver Grafikbereich, der berechnete Grafikobjekte anzeigen und und interaktive Manipulationen an Grafikobjekten auswerten kann.

Gesteuert wird der Bereich über drei Variablen oder Funktionen:

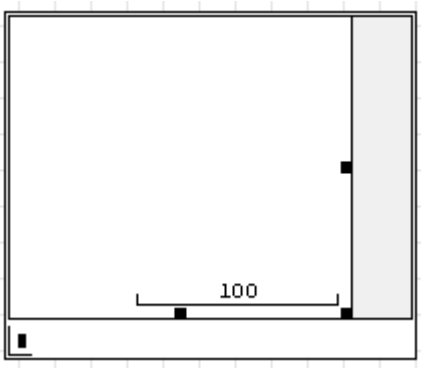
- Liste der Vorlagen (*templates*), welche die Beschreibung der Grafikobjekte enthalten, siehe Abschnitt 5.4.2,
- Vordefinierte Anordnung (*predefined layout*), siehe Abschnitt 5.4.3,
- Abhängige Anordnung (*dependent layout*)

In den Platzhalter wird eine Variable geschrieben. Diese enthält die Information über die vorhandenen Grafikobjekte, dies wird in Abschnitt 5.4.4 erläutert.

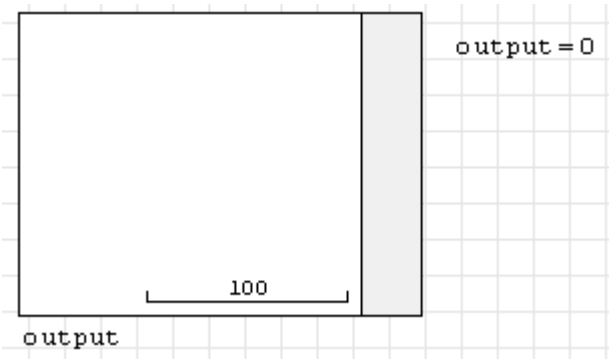
5.4.1. Erste Schritte mit dem Modeller

Er wird vom Plugin *Modeller Region* bereitgestellt (zum Download über die Erweiterungsverwaltung siehe Abschnitt 9.1).

Einfügen> Modeller

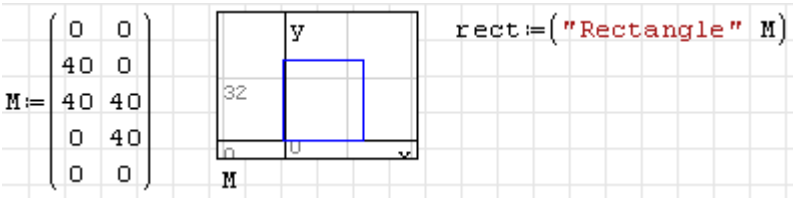


In den Platzhalter ist eine Ausgabevariable einzutragen. Solange noch keine Grafikobjekte vorhanden sind, ist ihr Wert Null.



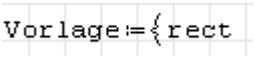
Um dem Modeller Grafikobjekte bekannt zu machen, müssen sie als Vorlage (*template*) bekannt gemacht werden. Eine Vorlage ist ein Zeilenvektor mit mehreren Parametern. Alle Vorlagen werden zu einer Liste zusammengefasst.

Die einfachste Vorlage enthält nur den Namen des Objekts und seine Randkurve. Hier wird ein Rechteck definiert:

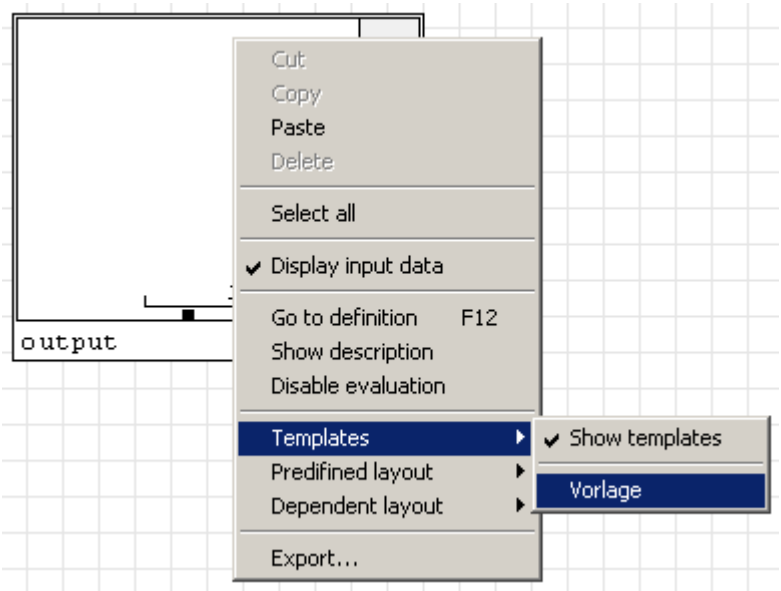


Wie man sieht, hat die Koordinatenmatrix der Randkurve das gleiche Format wie im 2D-Diagrammbereich.

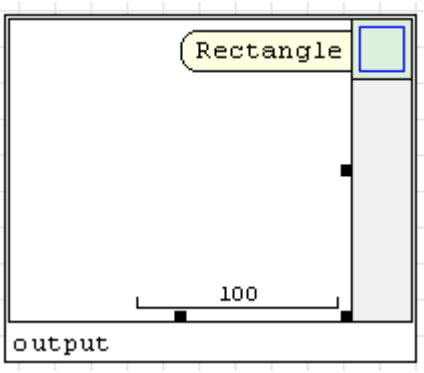
Die Vorlagenliste enthält in unserem Fall zunächst nur eine Vorlage:



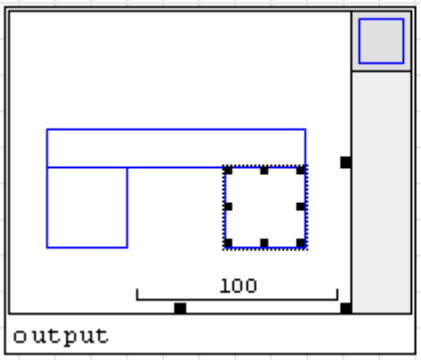
Im Kontextmenü des Modellerbereichs kann man die Liste als Vorlagenliste auswählen (angezeigt werden alle im Rechenblatt definierten Listen)



Nach Zuordnung der Vorlagenliste erscheint das Objekt in der Palette am rechten Bildrand. Ist der Bereich aktiviert, wird der Objektname angezeigt, wenn man mit dem Mauszeiger darübergeht.



Aus der Palette kann man nun das Objekt mit der Maus auf die Zeichenfläche ziehen. An den Platzhaltern kann man die Größe des Objekts ändern.



$$\text{output} = \begin{pmatrix} 1 & -47 & -22 & 40 & 40 & 0 & 0 & 230 & 0 & 0 \\ 1 & 42 & -22 & 40 & 40 & 1 & 0 & 241 & 0 & 0 \\ 1 & -2{,}5 & 7{,}5 & 129 & 19 & 0 & 0 & 247 & 0 & 0 \end{pmatrix}$$

Sobald Objekte auf der Zeichenfläche erscheinen, füllt sich die Ausgabevariable. Jede Zeile steht für ein Objekt, die Spalten stehen für Vorlagennummer (Position in der Vorlagenliste), Mittelpunktskoordinaten, Breite, Höhe, Auswahlstatus und weitere Daten, mehr dazu in Abschnitt 5.4.4.

5.4.2. Vorlagen

Alle Grafikobjekte werden durch Vorlagen beschrieben, alle Vorlagen sind in einer Liste enthalten. Die Vorlagenliste wird dem Modeller-Bereich unter dem Punkt „Templates" im Kontextmenü zugeordnet.

Eine Vorlage ist ein Zeilenvektor:

1. Name (Zeichenkette, wird als Hilfetext in der Vorlagenliste angezeigt)
2. $n \times 2$-Matrix mit Punktkoordinaten, beschreibt die Randkurve der Geometrie. Optional können Linienfarbe, Strichstärke und Füllfarbe angegeben werden, zusammengefasst zu einem Zeilenvektor.
3. Schalter, ob die Vorlage in der Seitenleiste des Modeller-Bereichs angezeigt werden soll (1: ja, 0: nein)
4. Spaltenvektor mit der Voreinstellung für die Objektabmessungen beim Einfügen aus der Seitenleiste.

In einer Vorlage können auch mehrere Randkurven enthalten sein, dann ist Punkt 2 nicht ein Zeilenvektor, sondern eine Matrix, in der jede Zeile einen Linienzug definiert.

5.4.3. Vordefinierte Anordnung

Damit können Objekte programmgesteuert angeordnet werden. Das kann entweder eine Liste oder eine Funktion sein. Variable mit passendem Inhalt müssen oberhalb des Modeller-Bereichs definiert werden und können im Kontextmenü unter „Predefined Layout" ausgewählt werden.

Definition mit einer Liste

Anzugeben ist eine Liste mit Zeilenmatrizen mit den Einträgen

1. Vorlagennummer
2. x-Koordinate des Objekts
3. y-Koordinate des Objekts
4. Breite
5. Höhe

Definition mit einer Funktion

Damit können auch die Skalierung und Position sowie der Vorlagenkontext gesteuert werden.

Es wird eine Funktion folgenden Aussehens definiert (Name ist egal)

`f(obj; ratio; x; y; id)` Rückgabewert ist eine Liste der statischen (nicht veränderbaren) Objekte, die zu zeichnen sind.

`obj` dargestellte Objekte. Variable kann verändert werden

`ratio` Skalierung des Koordinatensystems

`x,y` Koordinaten der Bildmitte

`id` Kontext für neue Einfügungen:

wenn positiv: ID des Objekts, welches als Container und aktueller Kontext benutzt werden soll (alle nachfolgend eingefügten Objekte sind Unterobjekte dieses Containers)

wenn negativ: Nummer der Vorlage, die als Kontext genommen wird.

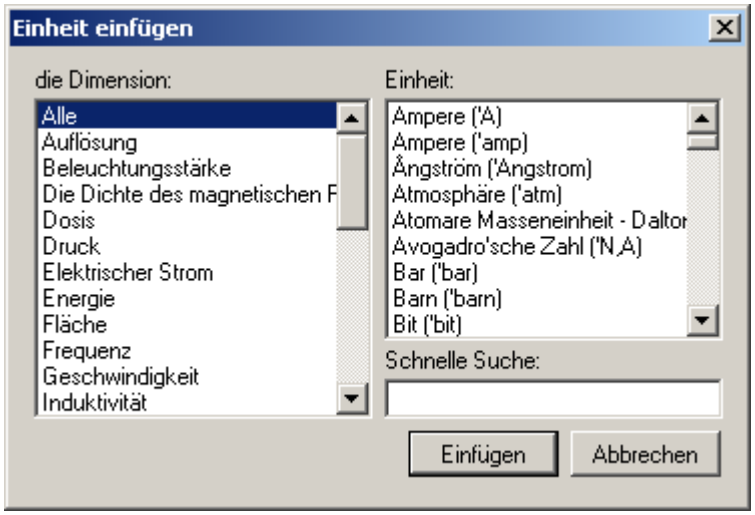
5.4.4. Ausgabe

Die Ausgabevariable wird in den Platzhalter des Modeller-Bereichs geschrieben und anhand der vorhandenen Grafikobjekte belegt. Geliefert wird eine Matrix mit einer Zeile für jedes Grafikobjekt, die Spalten haben die Bedeutung:

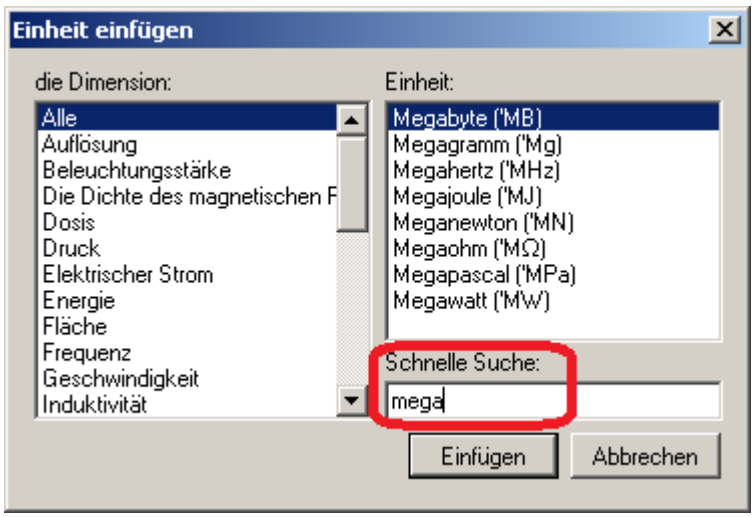
1. Vorlagennummer (Zeilennummer in der Vorlagenliste)
2. x-Koordinate des Objekts
3. y-Koordinate des Objekts
4. Breite
5. Höhe
6. Auswahlindikator (1: Objekt ist ausgewählt, 0: Objekt ist nicht ausgewählt)
7. Unbekannt
8. Objektnummer (ID)
9. Unbekannt
10. Unbekannt

6. Maßeinheiten

Eine besondere Stärke von SMath ist die Fähigkeit, mit Maßeinheiten zu rechnen. Jede Zahl kann mit einer Maßeinheit versehen werden. In SMath ist eine ganze Reihe von Einheiten verschiedener Dimensionen vordefiniert. Darüber hinaus sind auch einige physikalische Konstanten dabei. Einen Überblick verschafft man sich im Dialog **Hauptmenü> Einfügen> Einheit** (auch über Strg W oder mit  in der Werkzeugleiste erreichbar)



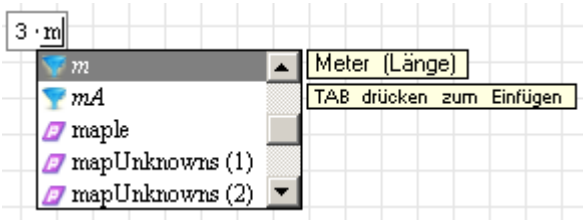
Im linken Fenster wählt man die Dimension, im rechten Fenster erscheinen dann die zugehörigen Maßeinheiten mit ihrem Namen und ihrer internen Darstellung, Im Feld „Schnelle Suche" kann man die Liste auf die Einheiten eingrenzen, die die angegebene Zeichenkette enthalten



Die interne Darstellung beginnt immer mit einem Hochkomma (z.B. `'m` für Meter).

Maßeinheiten können auch direkt mit der Tastatur eingegeben werden:

```
3m
```



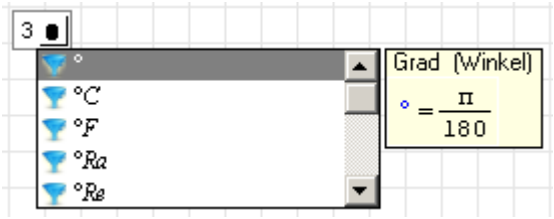
Sobald Sie einen Buchstaben in einem Formelbereich tippen, erscheint die dynamische Auswahl. Sie können weitere Buchstaben eintippen oder mit den Pfeiltasten aus der Liste auswählen. Mit ⇆ (Tabulator) wird bestätigt.

⇆

Maßeinheiten werden kursiv und blau dargestellt.

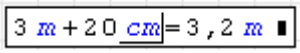
Tippt man vor dem Einheitennamen ein Hochkomma, schränkt sich die dynamische Auswahl direkt auf Maßeinheiten ein und der Platzhalter nimmt eine runde Form an:

3’



Hier die Summe zweier Längen:

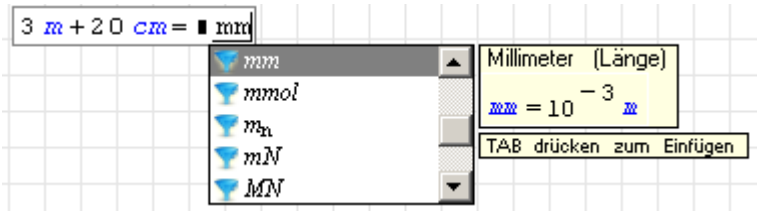
3m ⇆ +20cm ⇆ =



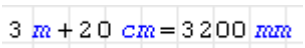
Die interne Darstellung diese Ausdrucks ist `3*’m+20*’cm=3.2*’m@#`

Um das Ergebnis in mm anzuzeigen, stellen Sie die Eingabemarke auf den Platzhalter hinter dem Ergebnis und tippen

mm

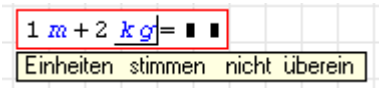


⇆ zur Bestätigung der Auswahl. Die Berechnung erfolgt erst nach Verlassen des Formelbereichs (↵ drücken oder irgendwo anders im Rechenblatt klicken, dann verschwindet der Rahmen um die Formel):

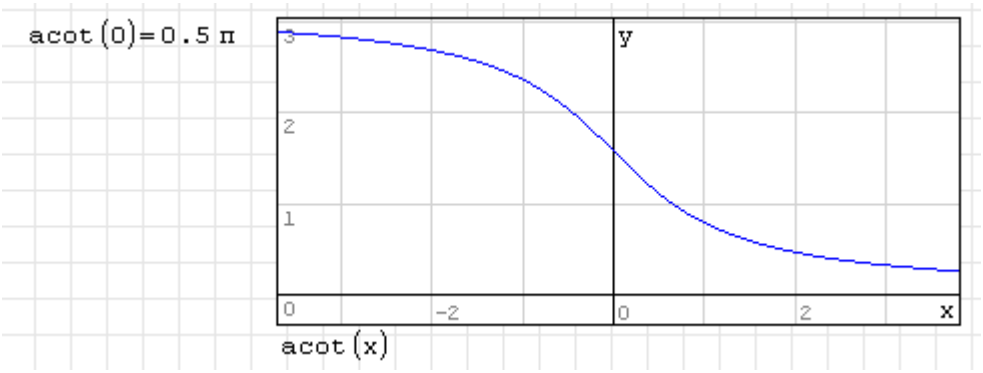


Intern rechnet SMath mit SI-Einheiten. Bei der Eingabe kann jeder Zahl eine Einheit zugeordnet werden. Diese legt die Dimension und den Vorfaktor gegenüber der SI-Einheit fest. Die Einheit mm kennzeichnet die Dimension Länge und hat den Faktor 0.001 in Bezug auf die SI-Einheit m.

Nur Größen gleicher Dimension können addiert, subtrahiert oder verglichen werden. Multiplikation und Division ist uneingeschränkt möglich.



Bei der Ergebnisanzeige (mit =) können nicht nur Maßeinheiten, sondern auch andere Faktoren, wie etwa π, abgespalten werden. Im linken Platzhalter erscheint dann einfach der verbleibende Restfaktor

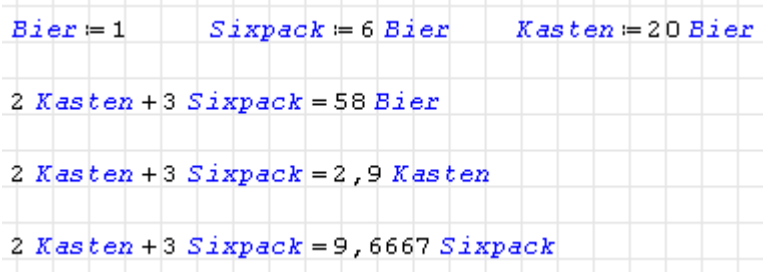


Als Argumente von Winkelfunktionen, Logarithmen als Exponenten sind Maßeinheiten nicht definiert. Entsprechend weigert sich SMath völlig korrekt, solche Berechnungen durchzuführen. Ausnahmen sind dimensionslose Maßeinheiten wie Grad (°) oder das Bogenmaß (rad).

Achtung: Der Vorsatz μ (Mikro) muss als `m` Strg - G eingegeben werden, also nicht das Zeichen AltGr -`m` eingeben. Die beiden Befehle scheinen unterschiedliche Zeichencodes zu erzeugen.

6.1. Definition eigener Maßeinheiten

Maßeinheiten können vom Anwender wie normale Variable definiert werden. Damit sie blau und kursiv erscheinen, stellt man dem Variablennamen ein Hochkomma (') voran, das in der Anzeige dann unsichtbar ist.



6.2. Vektoren und Matrizen mit Maßeinheiten

Elemente einer Matrix können verschiedene Einheiten haben. Matrixoperationen sind ausführbar, solange nicht Elemente unterschiedlicher Dimension addiert oder subtrahiert werden.

Bei der Anzeige ist es allerdings nicht möglich, auf die Maßeinheiten einzelner Elemente, Zeilen oder Spalten Einfluss zu nehmen. Wenn das erforderlich ist, müssen diese z.B. mit `row()`, `col()`, `submatrix()` oder Elementindex herausgegriffen und dann angezeigt werden.

Im Tabellenbereich ist es allerdings möglich, zeilenweise oder spaltenweise Anzeigeeinheiten vorzugeben.

6.3. Temperatureinheiten

SMath kennt die Temperatureinheiten Kelvin (K), Grad Celsius (°C), Grad Fahrenheit (°F), Grad Rankine (°Ra, die Absoluttemperatur passend zu Fahrenheit) und Grad Reamur (°Re).

Temperaturen können beliebig ineinander umgerechnet werden, wie dies bei anderen Dimensionen auch üblich ist. Die Nullpunktverschiebung wird dabei berücksichtigt.

Vorsicht ist **bei negativen Zahlenwerten** in nicht-absoluten Einheiten geboten!

$-10\ °C = -556{,}3\ °C$

Was ist hier passiert? Auf der linken Seite steht genau genommen: Null minus zehn Grad Celsius. Wegen Punktrechnung vor Strichrechnung rechnet SMath zunächst 10°C = 283,15 K, und wendet dann das Minuszeichen an. Die linke Seite wird also völlig korrekt ausgewertet zu −(10°C) = −(283,15 K). Bei der Anzeige in °C müssen vom Kelvin-Zahlenwert wieder 273,15 abgezogen werden, was dann den Wert von -556,3 ergibt.

Die Irritation ergibt sich daraus, dass das negative Vorzeichen (der unäre Minus-Operator) normalerweise höhere Priorität als alle anderen Rechenoperationen hat, in SMath aber als normale Strichrechnung betrachtet wird.

Das gewohnte Verhalten muss man daher mit Klammern erzwingen:

$(-10)\ °C = -10\ °C$

Temperaturdifferenzen dürfen nur mit Absolutskalen (also Kelvin oder Grad Rankine) angegeben und angezeigt werden!

$T_1 := (-10)\ °C$	$T_1 = 263{,}15\ K$	
$T_2 := 20\ °C$	$T_2 = 293{,}15\ K$	
$\Delta T := T_2 - T_1$	$\Delta T = 30\ K$	richtig
	$\Delta T = -243{,}15\ °C$	falsch

6.4. Winkel und Umdrehungen

Umdrehungen können als Ereignisse oder als Drehwinkel gezählt werden. Werden die Umdrehungen pro Zeit angegeben, dann handelt es sich um eine Frequenz f (Ereignisse oder Perioden pro Zeit).

Wird der Winkel pro Zeit angegeben (2π Radiant oder 360° für eine volle Umdrehung) dann handelt es sich um eine Winkelgeschwindigkeit ω oder gleichbedeutend damit um eine Kreisfrequenz.

Die Kreisfrequenz ω hat in Basiseinheiten den 2π-fachen Wert der Frequenz f.

Die Einheit rev (Umdrehung) in SMath ist als Winkeleinheit definiert:

$$rev = 2 \cdot \pi \qquad rev = 360\ °$$

Entsprechend führen die Einheiten rpm (Umdrehungen pro Minute) und rph (Umdrehungen pro Stunde) die Umrechnung in die Winkelgeschwindigkeit bereits durch:

$$rpm = \frac{\pi}{30\ s} \qquad rpm = 6\ \frac{°}{s} \qquad rpm = \frac{1}{60}\ \frac{rev}{s} \qquad rpm = \frac{\pi}{30}\ \frac{rad}{s}$$

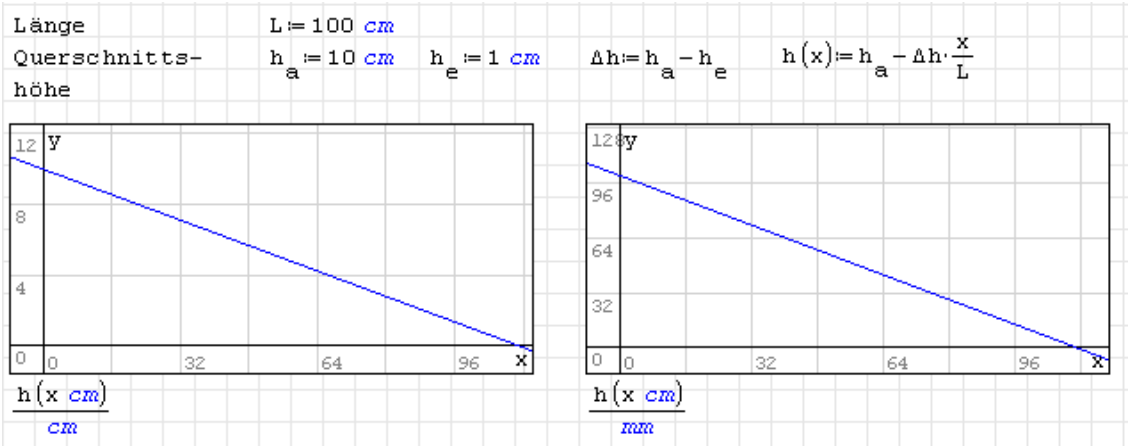
Das stiftet Verwirrung, denn eigentlich liegt nahe, dass Umdrehungen pro Zeit eher eine Frequenz als eine Kreisfrequenz meinen.

Als geeignete Frequenzeinheit für die Drehzahl kann man min^{-1} benutzen:

$$min^{-1} = \frac{1}{60\ s} \qquad min^{-1} = \frac{1}{60}\ Hz$$

6.5. Plotten mit Maßeinheiten

Die unabhängige Variable x in 2D-Plots (und y in 3D-Plots) ist dimensionslos. Die Plotfunktion erwartet dimensionslose Werte vom zu plottenden Ausdruck. Soll eine dimensionsbehaftete Funktion einer dimensionsbehafteten Größe geplottet werden, so multipliziert man das Argument mit der gewünschten Einheit der x-Achse und teilt das Ergebnis durch die gewünschte Einheit der y-Achse:



Das funktioniert natürlich auch mit mehreren Funktionen. Diese werden in einer Liste zusammengefasst und separat oder insgesamt auf die richtige Maßeinheit gebracht. Beim X-Y Plotbereich wird es genauso gemacht.

Die Maxima-Diagrammfunktionen Draw2D und Draw3D verhalten sich analog. Sollen die x- und y-Werte anders als in Basiseinheiten interpretiert werden, dann sind Argumente von Funktionen mit der gewünschten Einheit zu multiplizieren und das Ergebnis durch die gewünschte Einheit zu teilen.

6.6. Integration und Gleichungslösen

Bei symbolischer Integration und Gleichungslösung mit Maxima gibt es keine Besonderheiten zu beachten, da Maßeinheiten wir normale Variable betrachtet werden (allerdings wird stets ein positiver Wert unterstellt).

Ebenso unterstützen die numerischen Löser aus dem Plugin *NonLinear Solvers* einheitenbehaftete Größen. Dann müssen auch Startwerte und Toleranzen in den richtigen Einheiten angegeben werden müssen.

Die Integrationsvariable in bestimmten Integralen wird als dimensionslos betrachtet. Wie mit dimensionsbehafteten Integrationsvariablen umzugehen ist, wird in Abschnitt 4.12.2 erläutert.

6.7. Funktionen zum Umgang mit Maßeinheiten

Die Funktion `UoM(Ausdruck)` im Plugin *Custom Functions* (Abschnitt 9.4.3) liefert die zur Dimension des Ausdrucks passende Basiseinheit. Bei Matrizen oder Listen erfolgt dies elementweise.

Das Ergebnis hängt von der Auswertungseinstellung (Kontextmenü> Optimierung) ab.

Numerisches Ergebnis	**Symbolisches Ergebnis**
$\mathrm{UoM}(2)=1$	$\mathrm{UoM}(2)=1$
$\mathrm{UoM}\left(2\,\frac{kN}{m}\right)=1\,m\,Pa$	$\mathrm{UoM}\left(2\,\frac{kN}{m}\right)=\frac{kg}{s^2}$
$\mathrm{UoM}(2\,ft\,kip)=1\,J$	$\mathrm{UoM}(2\,ft\,kip)=\frac{kg\,m^2}{s^2}$
$\mathrm{UoM}\left(2\,\frac{N}{mm^2}\right)=1\,Pa$	$\mathrm{UoM}\left(2\,\frac{N}{mm^2}\right)=\frac{kg}{m\,s^2}$
$\mathrm{UoM}(2\,cm)=1\,m$	$\mathrm{UoM}(2\,cm)=m$
$\mathrm{UoM}\left(\begin{pmatrix}3\,kN\\4\,kN\end{pmatrix}\right)=\begin{pmatrix}1\,\frac{kg\,m}{s^2}\\1\,\frac{kg\,m}{s^2}\end{pmatrix}$	$\mathrm{UoM}\left(\begin{pmatrix}3\,kN\\4\,kN\end{pmatrix}\right)=\begin{pmatrix}\frac{kg\,m}{s^2}\\\frac{kg\,m}{s^2}\end{pmatrix}$
$\mathrm{UoM}\left(\begin{Bmatrix}3\,kN\\4\,kN\end{Bmatrix}\right)=\begin{Bmatrix}1\,\frac{kg\,m}{s^2}\\1\,\frac{kg\,m}{s^2}\end{Bmatrix}$	$\mathrm{UoM}\left(\begin{Bmatrix}3\,kN\\4\,kN\end{Bmatrix}\right)=\begin{Bmatrix}\frac{kg\,m}{s^2}\\\frac{kg\,m}{s^2}\end{Bmatrix}$

Mit dieser Funktion kann man beispielsweise Größen vorübergehend dimensionslos machen, etwa zur Anwendung von numerischen Verfahren.

6.8. Vordefinierte Einheiten und Konstanten

Konstanten und Maßeinheiten werden in der Datei *Installationsverzeichnis/Entries/Units.xml* definiert. Dort kann man auch eigene xml-Dateien einstellen, die Maßeinheitendefinitionen enthalten. In Vorbereitung ist ein Mechanismus, mit dem man eigene Einheitendefinitionen in der Online-Galerie als Erweiterung veröffentlichen kann. Vorläufig ist es für die Portabilität besser, solche Definitionen im SMath-Rechenblatt selbst vorzunehmen.

In SMath sind für einige Dimensionen Standardeinheiten definiert. Diese geben an, in welcher Einheit ein Ergebnis angezeigt wird, sofern man nicht selbst eine Einheit vorgibt.

Eine Liste aller vordefinierter Maßeinheiten findet sich in Anhang C.

7. Programmierung

Man kann in SMath Programme schreiben. Das wird unterstützt durch

- Benutzerdefinierte Funktionen
- Funktionen zur Ablaufsteuerung (Schleifen, Verzweigungen, Anweisungsblöcke)
- Funktionen für Zeichenketten
- Debug-Funktionen
- Zeitmessung (inoffizielle Funktion `time()`, siehe Seite 339)

7.1. Benutzerdefinierte Funktionen

Funktionen werden definiert, indem man dem Namen in Klammern eine Argumentliste anfügt. Ergebnis ist der mit den Argumentwerten ausgewertete Ausdruck auf der rechten Seite der Definition.

$f(x) := x^2 \qquad f(4) = 16$

Mehrere Argumente werden durch das Argumenttrennzeichen getrennt, das je nach Einstellung ein Komma oder ein Semikolon ist (der ebenfalls verfügbare Punkt ist eher unüblich). Im vorliegenden Handbuch wird stets das Semikolon benutzt, da das Komma Dezimaltrennzeichen ist.

$g(x ; y) := x + y \qquad g(1 ; 3) = 4$

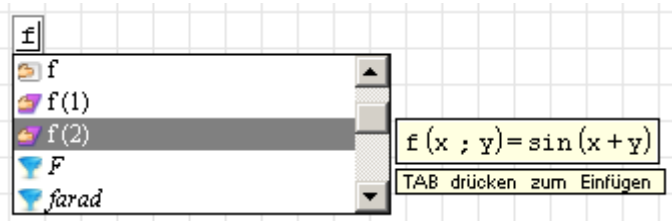
Einige Besonderheiten sind in SMath zu beachten:

- Variable und Funktionen mit gleichem Namen sind unterschiedliche Objekte, genauso wie Funktionen gleichen Namens mit unterschiedlicher Zahl von Argumenten.

  $f := 3 \qquad f(x) := x^2 \qquad f(x ; y) := \sin(x + y)$

  $f = 3 \qquad f(a) = a^2 \qquad f(y ; y) = \sin(2 \cdot y)$

  Die dynamische Hilfe kennzeichnet die verschiedenen Versionen durch die Zahl ihrer Argumente und zeigt die Definition des ausgewählten Objekts an.



- Alle Namen (Variable und Funktionen) im aufrufenden Kontext (das Rechenblatt oder das Innere einer Funktion, aus der der Aufruf erfolgt) sind in der aufgerufenen Funktion bekannt, wenn
- Die rechte Seite der Funktionsdefinition wird zum Zeitpunkt der Definition ausgewertet und das Ergebnis als Definition gespeichert. Die gespeicherte Form kann von der Optimierungseinstellung (numerisch, symbolisch, abgeschaltet) bei der Definition abhängen. Prüfen Sie jede Definition auf eine der im folgenden beschriebene Arten

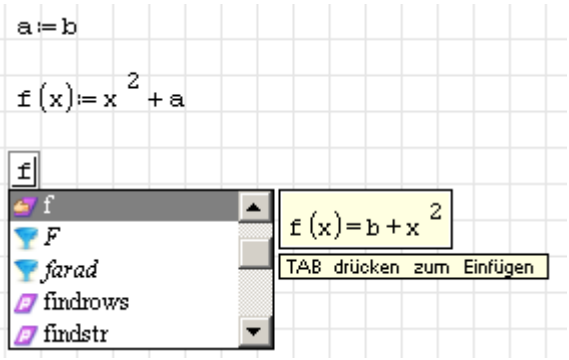
Es gibt drei Wege, die tatsächlich gespeicherte Definition einer Funktion anzuzeigen.

1. Anzeige des Ergebnisses der Definition selbst mit =
2. Maus über der Definition schweben lassen
3. Funktionsname eintippen, die dynamische Hilfe zeigt dann den Ausdruck an.

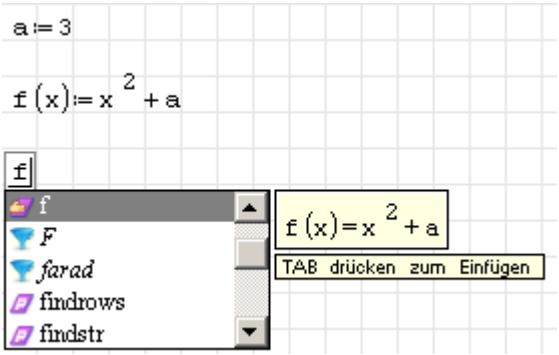
7.1.1. Symbolische/Numerische Optimierung, Verwendung von eval():

Funktionsdefinitionen können im Kontextmenü (rechte Maustaste) wie jeder Formelbereich auf symbolische oder numerische Optimierung umgestellt werden.

Optimierung symbolisch: Die rechte Seite wird symbolisch vereinfacht und erst dann als Definition verwendet. Eventuell bekannte globale Bezeichner werden berücksichtigt und durch ihre Werte ersetzt. Im angegebenen Beispiel wird der globale Bezeichner a durch den bekannten Wert b ersetzt. Beim Funktionsaufruf ist der aktuelle Wert von b maßgebend.



Optimierung numerisch: Es wird versucht, die rechte Seite numerisch auszuwerten. Bei Funktionsdefinitionen scheitert dies in der Regel, da ja die Argumente als Symbole erhalten bleiben müssen. In diesem Fall wird die rechte Seite unverändert als Definition verwendet. Auch bekannte numerische Werte für globale Bezeichner werden dann ignoriert. Eine Abfrage der Bezeichner erfolgt erst beim Funktionsaufruf, wenn der aufrufende Ausdruck auf Optimierung symbolisch gestellt ist. Im Beispiel wird der bekannte Zahlenwert von a zur Definitionszeit ignoriert, da die numerische Berechnung wegen unbekanntem x scheitert. Beim Funktionsaufruf sind dann der aktuelle Wert von a und natürlich das Argument x maßgebend.



eval()-Funktion: Ihre Verwendung hat zwei Folgen:

- Alles was innerhalb der `eval`-Klammer steht, wird entweder numerisch berechnet oder wenn dies scheitert, überhaupt nicht vereinfacht, auch wenn „Optimierung symbolisch" eingestellt ist.
- Eine symbolische Vereinfachung des Klammerinhalts bei Funktionsaufruf ist nicht möglich (Einstellung „Optimierung symbolisch" im aufrufenden Ausdruck wird ignoriert).

line()-Funktion (Anweisungsblock): Anweisungsblöcke sind zunächst nur aufeinanderfolgende Anweisungen, die mit der Funktion `line()` gruppiert werden.

Besonderheiten sind zu beachten, wenn der Anweisungsblock die rechte Seite einer Zuweisung bildet.

- Dann bildet der Block einen lokalen Namensraum.
- Alle außerhalb des Blocks definierten Variablen sind innerhalb sichtbar.
- Lokale Zuweisungen (innerhalb des Blocks) erzeugen stets lokale Variablen, die grundsätzlich nicht außerhalb sichtbar sind.
- Zuweisungen auf Bezeichner, die schon außerhalb verwendet wurden, erzeugen eine lokale Variable gleichen Namens, so dass die gleichnamige globale Variable innerhalb nicht mehr sichtbar ist, aber auch nicht global geändert wird.
- Ergebnis des Anweisungsblocks ist das Ergebnis der letzten Zeile im Block.
- Rekursion ist möglich, das heißt, die durch den Anweisungsblock definierte Funktion kann im Anweisungsblock selbst aufgerufen werden.

Variablennamen, die ausschließlich in der letzten Zeile eines Anweisungsblocks auftauchen, ohne dabei oder weiter oben i Block in Operationen verwendet zu werden, werden von SMath als undefiniert betrachtet. Das wird beispielsweise in der Funktion `Clear()` des Plugins *Custom Functions* benutzt.

7.1.2. Funktionsaufruf

Beim Funktionaufruf werden den Formalparametern der Funktion beliebige Ausdrücke (Argumente) zugeordnet.

Die Argumente werden zunächst alle im rufenden Kontext (außerhalb der aufgerufenenen Funktion) ausgewertet. Dieser Vorgang heißt *pre-processing* und besteht in symbolischer oder numerischer Optimierung je nach Einstellungen im Kontextmenü.

Anschließend werden die ausgewerteten Ausdrücke der Reihe nach von links nach rechts den Formalparametern zugewiesen. Das geschieht im Inneren der Funktion, der rufende Kontext bemerkt davon nichts.

Besondere Vorsicht ist geboten, wenn beim Funktionsaufruf Variablennamen benutzt werden, die auch als Namen der Formalparameter vorkommen. Betrachten Sie das folgende Beispiel:

$f(x;y):=\sin(x+y)$	Definition
$f(z;x)=\sin(2\cdot z)$	Aufruf
$x:=z=z$	Zuweisung von Wert z auf Parameter x
$y:=x=z$	Zuweisung von Wert x auf Parameter y
$\sin(x+y)=\sin(2\cdot z)$	Berechnung des Rückgabewerts

Der Variablenname x kommt sowohl im Aufruf (2. Argument) als auch in der Defintion als erster Parameter vor.

7.1.3. Parameterübergabe

Die Funktionsdefinition besteht aus

1. dem Namen der Funktion
2. den in Klammern eingeschlossenen und durch Argumenttrennzeichen getrennten Namen der Formalparameter (Argumente)
3. dem Funktionskörper, typischerweise ein Ausdruck, in dem die Formalparameter vorkommen.

Der Funktionsaufruf besteht aus dem Namen der Funktion und in Klammern eingeschlossenen Werten (Aktualparameter, Argumente) für die Formalparameter. Diese Werte (beliebige Ausdrücke) werden den Formalparametern zugewiesen und dann der Funktionskörper ausgewertet.

Die Zuordnung zwischen den Aktualparametern (Argumenten) und den Formalparametern erfolgt anhand der Position in der Parameterliste, das erste Argument wird dem ersten Formalparameter zugewiesen und so weiter.

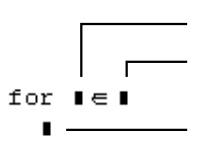
Verwenden Sie stets Namen für die Parameter, die weiter oben im Blatt noch nicht definiert wurden. Um Konflikte zu vermeiden, kann man Namen verwenden, die man im Blatt normalerweise nicht verwendet, z.B. `a_` oder `a#`.

Der Buchstabe i kann nicht als Parametername in Funktionsdefinitionen verwendet werden. Das wird intern verhindert, da sich SMath in diesem Fall unlogisch verhalten würde. Beim Aufruf wird i nicht durch den Aktualparameter ersetzt, sondern es wird die globale Konstantendefinition $i = \sqrt{-1}$ eingesetzt.

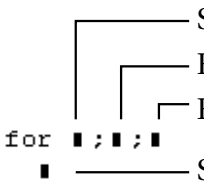
Beide Probleme haben ihre Ursache in unsauberer Trennung des globalen Namensraums vom lokalen Namensraum. Leider können die Formalparameter nicht uneingeschränkt als lokale Variable betrachtet werden.

7.2. Schleifen

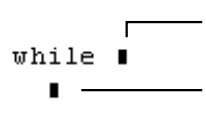
Schleifen dienen der wiederholten Ausführung von Anweisungen. Eine häufige Anwendung ist die elementweise Bearbeitung von Listen (Matrizen).



Mengenschleife, bei der die Schleifenvariable nacheinander alle Werte des Wertebereichs annimmt. Der Wertebereich ist meist ein Vektor von Zahlen, der mit der Funktion range erzeugt wird.

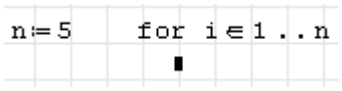


Kopfgesteuerte Schleife. Zuerst wird die Startanweisung ausgeführt. Ist die Fortsetzungsbedingung erfüllt, wird der Schleifenkörper ausgeführt und anschließend die Fortsetzungsanweisung ausgeführt und die Fortsetzungsbedingung erneut geprüft.

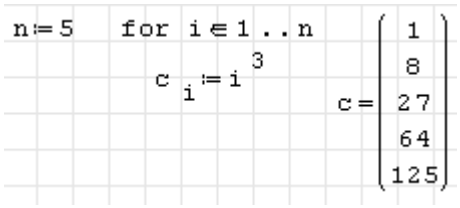


Kopfgesteuerte Schleife. Die Schleife wird durchlaufen, solange die Fortsetzungsbedingung erfüllt ist.

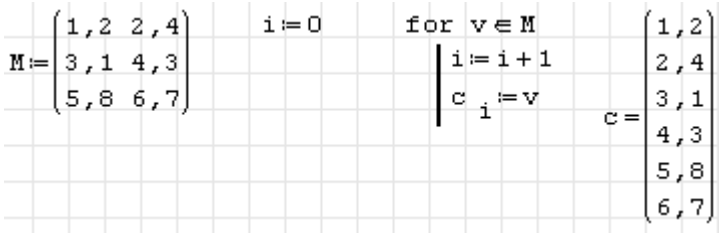
Wenn eine Schleifenvariable i ganzzahlige Werte von 1 bis n annehmen soll, dann schreibt man unter Verwendung der Funktion `range(2)`



So kann man beispielsweise einen Vektor der Kubikzahlen erzeugen. Beachten Sie, dass die Variable c nicht als Matrix vordefiniert werden muss. Sie wird dynamisch genau so groß dimensioniert, dass das Element mit dem höchsten Index noch reinpasst.



Als Wertebereich der Schleifenvariable kann jeder Vektor oder jede Matrix angegeben werden. Matrizen werden dabei mit nur einem Index indiziert, wobei die Elemente zeilenweise abgearbeitet werden. Im folgenden Beispiel ist v die Schleifenvariable, die nacheinander die Werte aus der Matrix $\underline{\underline{M}}$ annimmt. Die Variable i ist der Index für den Ergebnisvektor $\underline{c}$, in den die Werte der Schleifenvariable nacheinander geschrieben werden, so dass die Bearbeitungsreihenfolge (zeilenweise) sichtbar wird.



Bei der folgenden Schleife kommt es aber auf die Bearbeitungsreihenfolge nicht an.

$$M := \begin{pmatrix} 1{,}2 & 2{,}4 \\ 3{,}1 & 4{,}3 \\ 5{,}8 & 6{,}7 \end{pmatrix} \qquad s := 0 \qquad \text{for } i \in M \quad s := s + i \qquad s = 23{,}5 \qquad \sum M = 23{,}5$$

Gerade wenn man aus Funktionen Grafikobjekte erzeugen will, ist es oft bequemer, der Schleifenvariable direkt die x-Werte zuzuweisen. Dazu dient die Funktion range(3), die einen Vektor durch Angabe des ersten, des zweiten und des letzten Arguments definiert. Das folgende Beispiel zeigt, wie es funktionieren soll:

$$x := -\pi ; -\pi + \frac{\pi}{2} .. \pi \qquad x = \begin{pmatrix} -1 \\ -0{,}5 \\ 0 \\ 0{,}5 \\ 1 \end{pmatrix} \pi$$

Aus dem ersten und dem zweiten Wert wird das Inkrement (die Schrittweite) bestimmt. Die Schleife wird beginnend mit dem ersten Wert abgearbeitet. Dann wird die Schleifenvariable um das Inkrement erhöht. Wenn damit der Endwert noch nicht überschritten wurde, wird der Durchlauf wiederholt.

Es ist ein Grundgesetz sauberer Programmierung, dass die Gleichheit zweier Fließkommazahlen nie ohne eine sinnvolle Toleranz geprüft werden soll. Die exakte Übereinstimmung zwischen auf verschiedenen Wegen, also mit unterschiedlichen Rundungsfehlern, berechneten Fließkommazahlen ist reine Glückssache. Im Beispiel oben hat es funktioniert, in folgendem Beispiel funktioniert es nicht:

$$x := 0{,}03 ; 0{,}035 .. 0{,}05 \qquad x = \begin{pmatrix} 0{,}03 \\ 0{,}035 \\ 0{,}04 \\ 0{,}045 \end{pmatrix}$$

Die Schleife sollte eigentlich fünfmal durchlaufen werden, aber offenbar ist intern $0{,}045 + 0{,}005$ schon größer als der Endwert $0{,}05$. Das ist kein Bug in SMath, sondern ein normales Verhalten von Fließkommazahlen. Dennoch ist dieses Verhalten unschön. Man macht das Verhalten solcher Schleifen robust, indem man

- den Endwert um einen kleinen Betrag erhöht, der aber kleiner als das Inkrement sein muss, sonst erzeugt man einen zusätzlichen Durchlauf:

$$x := 0{,}03 ; 0{,}035 .. 0{,}05 + 10^{-16} \qquad x = \begin{pmatrix} 0{,}03 \\ 0{,}035 \\ 0{,}04 \\ 0{,}045 \\ 0{,}05 \end{pmatrix}$$

- ganzzahlige Schleifenvariable benutzt und die erforderlichen Fließkommazahlen daraus berechnet:

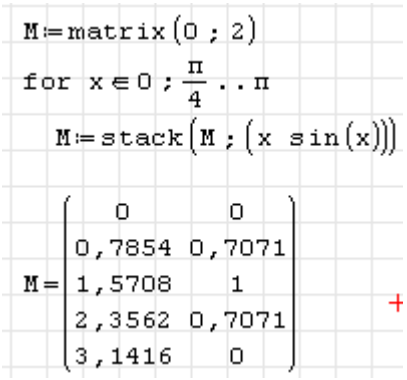
$$x := 0{,}03 + 0{,}005 \cdot (0 .. 4) \qquad x = \begin{pmatrix} 0{,}03 \\ 0{,}035 \\ 0{,}04 \\ 0{,}045 \\ 0{,}05 \end{pmatrix}$$

Ideal wäre eine Schleifenfunktion, die

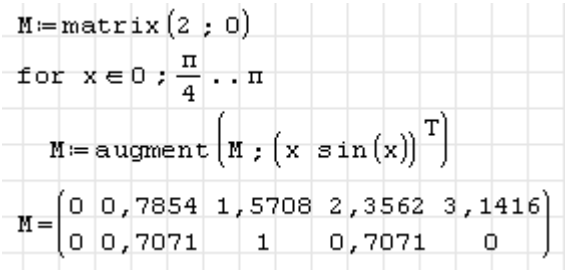
- die Argumente start, stop und schritt verwendet (damit man den Schritt direkt angeben kann, und nicht erst das zweite Schleifenelement aus dem Startwert und dem Schritt berechnen muss)
- die Abbruchbedingung robust prüft, indem z.B. als Toleranz die halbe Schrittweite verwendet wird (diese wird beim Vergleich auf den Endwert aufgeschlagen)
- den Wertebereich in der Form `start..stop by schritt` darstellt, wenn Schritt von Eins verschieden ist.

Anwendungsbeispiel: Zeilen- oder spaltenweiser Aufbau von Matrizen

Matrizen können mit der Funktion `stack()` um weitere Zeilen erweitert werden. Wenn man dies in einer Schleife tut, dann muss die Matrix vorher initialisiert werden, damit die Matrix im ersten Durchlauf schon existiert. Wenn dort aber die erste Zeile berechnet werden soll, muss man die Matrix mit leerer Zeilenzahl und passender Spaltenzahl initialisieren.



Spaltenweise geht das mit der Funktion `augment()` genauso:



7.3. Fehlersuche

Wenn Probleme mit Programmen auftreten, dann kann stehen einige Hilfsmittel zur Verfügung:

- die dynamische Auswahl: Sie zeigt für alle definierten Objekte die Belegung an.
- die Funktion `trace()` und das Ausgabefenster.

Die Funktion `trace()` kann in Anweisungsblöcke integriert werden, um zur Laufzeit die Belegung von Variablen anzuzeigen. Die Ausgabe erfolgt im Ausgabefenster, welches mit **Hauptmenü> Ansicht** ein- und ausgeschaltet werden kann.

Die Verwendungsmöglichkeiten und die Ausgabeformatierung werden im SMath Wiki erläutert.

7.4. Funktionen für Zeichenketten

Für Zeichenketten stehen folgende Funktionen zur Verfügung:

Zusammenfügen

```
concat("abc"; "def"; "ghi")="abcdefghi"
```

```
strjoin(", "; {"bread"; "milk"; "sugar"; "..."})="bread, milk, sugar, ..."
```

Teilketten herausgreifen

Startposition und Länge können angegeben werden.

```
substr("abcdefghijkdef"; 5)="efghijkdef"
substr("abcdefghijkdef"; 5; 3)="efg"
```

Aufteilen anhand von Trennzeichen:

```
strsplit("the QUICK bRoWn FoX..."; " ")={"the"; "QUICK"; "bRoWn"; "FoX..."}
strsplit("the QUICK bRoWn FoX..."; "o")={"the QUICK bR"; "Wn F"; "X..."}
```

Teilketten ersetzen oder löschen

```
strrep("abcdefghijkdef"; "def"; "123")="abc123ghijk123"
strrep("abcdefghijkdef"; "def"; "")="abcghijk"
```

Teilketten suchen

Die Funktion `findstr()` liefert einen Vektor mit den Anfangspositionen der Teilkette in einer gegebenen Zeichenkette. Wenn die Teilkette nicht vorkommt, liefert die Funktion den Wert -1.

```
findstr("abcdefghijkdef"; "def")={4; 12}
findstr("abcdefghijkdef"; "ab")=(1)
findstr("abcdefghijkdef"; "xy")=-1
```

Länge bestimmen

```
strlen("12345")=5
```

Prüfen, ob eine Zeichenkette vorliegt:

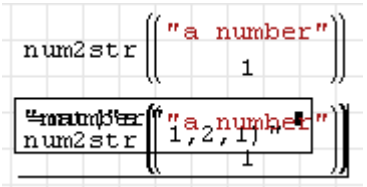
```
IsString("abc")=1
IsString(√2)=0
IsString({"abc"; √2})=0   Ausdruck ist vom Typ "Liste" (system)
```

Ausdruck in Zeichenkette umwandeln

Die Ausdrücke werden symbolisch ausgewertet und in die interne Textdarstellung umgewandelt. Man kann kein Format für die Zahlen vorgeben.

```
num2str(2,3)="23/10"
num2str(a+b)="a+b"
num2str({"a"; π})="sys("a",π,2,1)"
num2str([1 2; 3 4])="mat(1,2,3,4,2,2)"
num2str(∫ x^2 dx)="int(x^2,x)"
```

Der umzuwandelnde Ausdruck darf keine Zeichenketten mit Leerzeichen enthalten, sonst gibt es seltsame Fehler



Zeichenketten in Ausdrücke umwandeln

`str2num()` ist die Umkehrfunktion zu `num2str()`.

```
num2str({"abc"; √2})="sys("abc",sqrt(2),2,1)"
str2num(num2str({"abc"; √2})={"abc"; √2}   Optimierung symbolisch
str2num("sqrt(x^2+y^2)")=√(x^2+y^2)   Optimierung symbolisch
```

Groß- und Kleinschreibung

Diese Funktionen sind im Plugin *Custom Functions* (siehe Abschnitt 9.4.3) enthalten. Anders als die Standardfunktionen für Zeichenketten können sie auch auf Matrizen und Listen angewendet werden, die Zeichenketten enthalten.

Alles in Kleinbuchstaben umwandeln:

```
strtolower("the QUICK bRoWn FoX...")="the quick brown fox..."

strtolower(["the QUICK bRoWn FoX..."; √2; "another sTRINg"]) = ["the quick brown fox..."; √2; "another string"]
```

Alles in Großbuchstaben umwandeln:

```
strtoupper("the QUICK bRoWn FoX...")="THE QUICK BROWN FOX..."

strtoupper(["the QUICK bRoWn FoX..."; √2; "another sTRINg"]) = ["THE QUICK BROWN FOX..."; √2; "ANOTHER STRING"]
```

Anfangsbuchstaben der Zeichenketten in Großbuchstaben umwandeln:

```
ucfirst("the QUICK bRoWn FoX...")="The QUICK bRoWn FoX..."

ucfirst(["The QUICK bRoWn FoX..."; √2; "another sTRINg"]) = ["The QUICK bRoWn FoX..."; √2; "Another sTRINg"]
```

Anfangsbuchstaben aller Wörter in Zeichenketten in Großbuchstaben umwandeln:

```
ucwords("the QUICK bRoWn FoX...")="The QUICK BRoWn FoX..."

ucwords(["The QUICK bRoWn FoX..."; √2; "another sTRINg"]) = ["The QUICK BRoWn FoX..."; √2; "Another STRINg"]
```

Beschreibungstext von Definitionen abfragen

Nähere Erläuterungen in Abschnitt 3.12

```
Hier wird a definiert
a:=1,32
description(a)="Hier wird a definiert"
```

7.5. Code-Bausteine

Code-Bausteine, in der englischen Version „snippets", sind .sm-Dateien, die zum Einfügen in andere Dateien vorgesehen sind, um dort bestimmte Funktionen bereitzustellen. Das geschieht in den Schritten:

- Schreiben des gewünschten Inhalts in eine .sm-Datei.
- Definition, unter welchem Namen der Baustein in SMath bekannt sein soll. Dieser Name wird in der dynamischen Auswahl aufgelistet. Dafür wird unter **Hauptmenü> Datei> Eigenschaften** auf der Seite „Dateiattribute" das Feld „Baustein-Tastaturkürzel" ausgefüllt.
- Titel, Beschreibung, Autor und Firma aus der Dateibeschreibung werden in der Erweiterungs-Verwaltung angezeigt.
- .sm-Datei in den Ordner *snippets* der SMath-Installation verschieben (das geht nur bei der Installer-Version, nicht bei der portablen).
- Beim nächsten Neustart ist die Datei dem SMath-System bekannt.
- Bausteine können als Blattbereich eingefügt werden, der standardmäßig eingeklappt ist und den Namen des Bausteins als Titel hat.

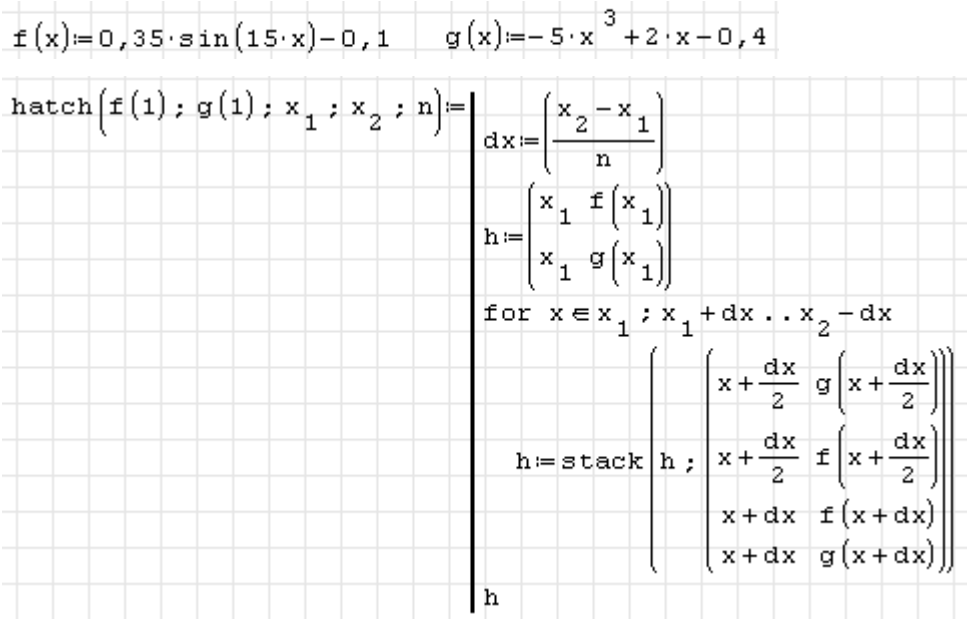
Diese Funktion wurde in SMath 0.90 eingeführt. Sehen Sie hier das Ankündigungsvideo.

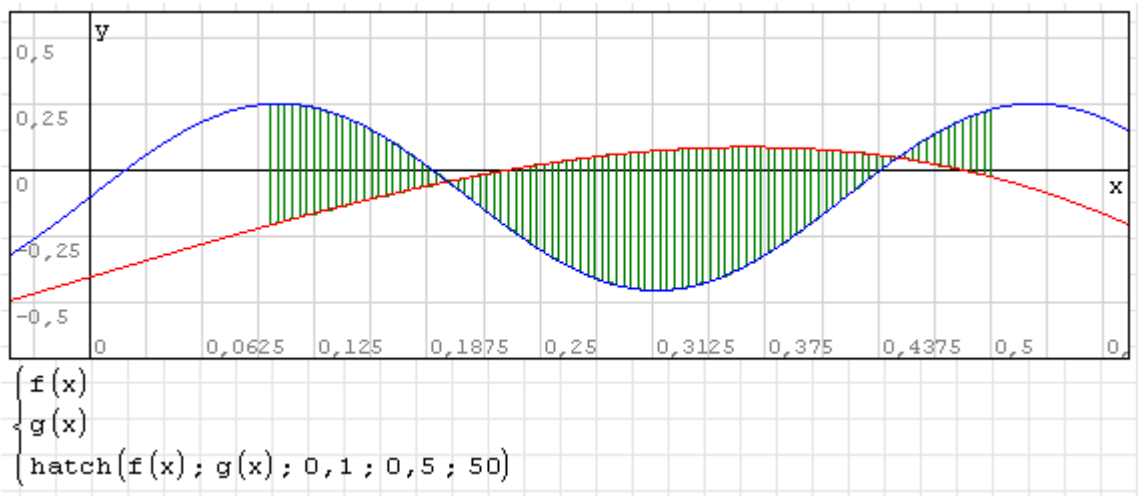
Eingefügte Bausteine sind völlig gleichwertig zu anderweitig eingefügten Anweisungen im Rechenblatt, es gibt also keinen getrennten Namensraum und keine Eingabe- und Ausgabeparameter. Das muss von den Funktionsdefinitionen im Baustein geregelt werden.

Code-Bausteine kann man auch aus der Online-Galerie laden oder dort einstellen, dies wird im Kapitel „Erweiterungen", Abschnitt 9.2 erklärt.

7.5.1. Erstellung von Bausteinen

Die Schritte werden anhand einer Funktion zum Schraffieren der Fläche zwischen zwei Kurven in 2D-Diagrammen erläutert. Das kann man beispielsweise folgendermaßen realisieren:

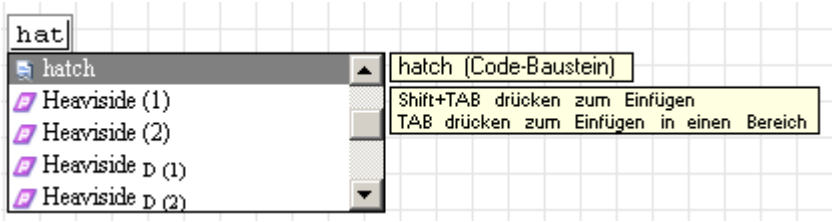




Grundsätzlich reicht es zunächst, die Funktion `hatch()` in eine .sm-Datei beliebigen Namens zu schreiben und diese im Ordner *snippets* der SMath-Installation abzulegen. Dann findet sie die Code-Baustein-Verwaltung beim nächsten Aufruf und von dort kann man den Baustein ins aktuelle Rechenblatt einfügen.

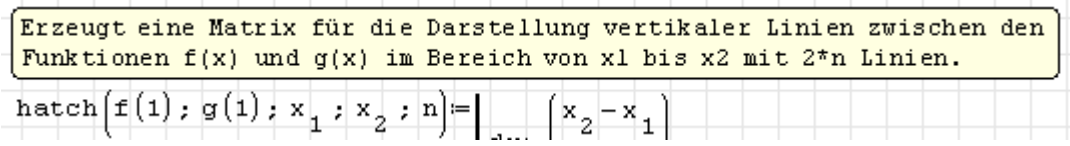
Gibt man in den Dateieigenschaften das „Tastaturkürzel" an,

dann kennt auch die dynamische Auswahl den Baustein:

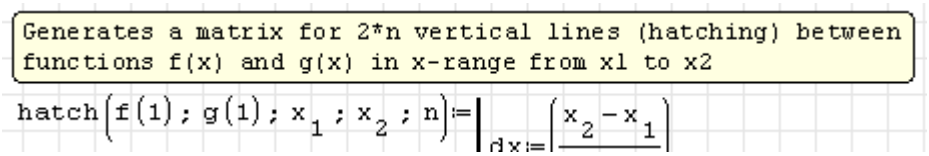


Wirklich wiederverwendbar wird der Baustein aber erst, wenn er ordentlich dokumentiert wird. Es gibt gegenwärtig keine Möglichkeit, irgendeine Baustein-Beschreibung für die dynamische Auswahl anzugeben. Was man bekommt, erfährt man hier also erst, wenn der Baustein eingefügt ist. Dann kennt die dynamische Auswahl alle enthaltenen Definitionen nebst deren Beschreibungen.

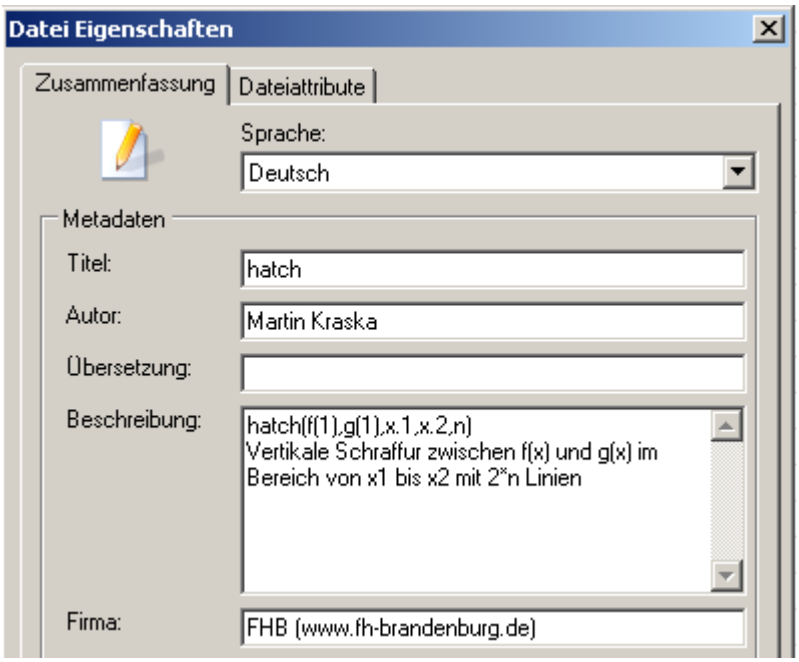
Man sollte es sich daher zur Gewohnheit machen, die Funktionen mit Beschreibungen zu versehen. In unserem Fall also beispielsweise so:



Bausteine, die Sie im SMath-Forum veröffentlichen wollen, sollten mindestens noch eine englische Beschreibung erhalten:



Bei Code-Bausteinen ist es sinnvoll, die Datei-Eigenschaften (gegebenenfalls auch mehrsprachig) auszufüllen. Diese Information kann man im Fenster „Datei-Eigenschaften" mehrsprachig angeben.



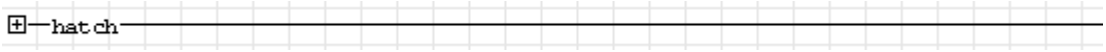
In der Baustein-Verwaltung werden der Titel, die Beschreibung, das Baustein-Tastaturkürzel und die Autor (Firma)-Information angezeigt, siehe nächster Abschnitt.

7.5.2. Verwendung von Bausteinen

Zunächst muss der Baustein ins Rechenblatt eingefügt werden. Dafür gibt es zwei Optionen:

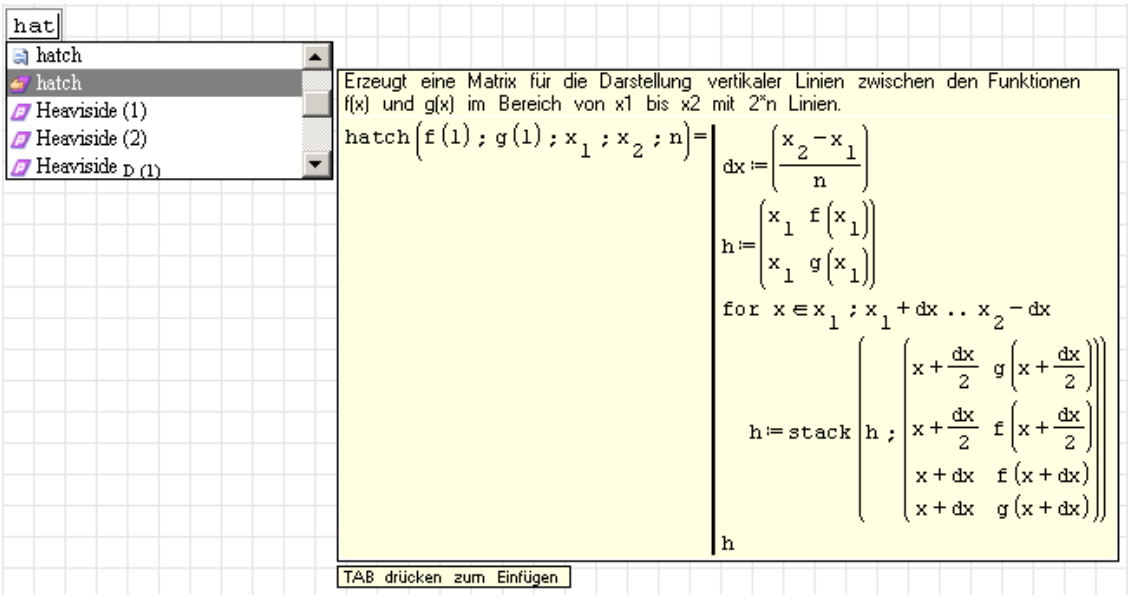
1. Durch Eintippen des Namens an der gewünschten Stelle des Rechenblatts. Dann erscheint die dynamische Auswahl. Der Baustein-Name kann dann mit ⇆ bestätigt werden, damit wird der Baustein in einem zugeklappten Blattbereich eingefügt.
2. **Hauptmenü> Extras> Code-Baustein-Verwaltung** ruft den Dialog „Erweiterungs-Verwaltung" auf. Dort kann der Baustein ausgewählt und per Einfügen ins aktuelle Rechenblatt eingefügt werden.

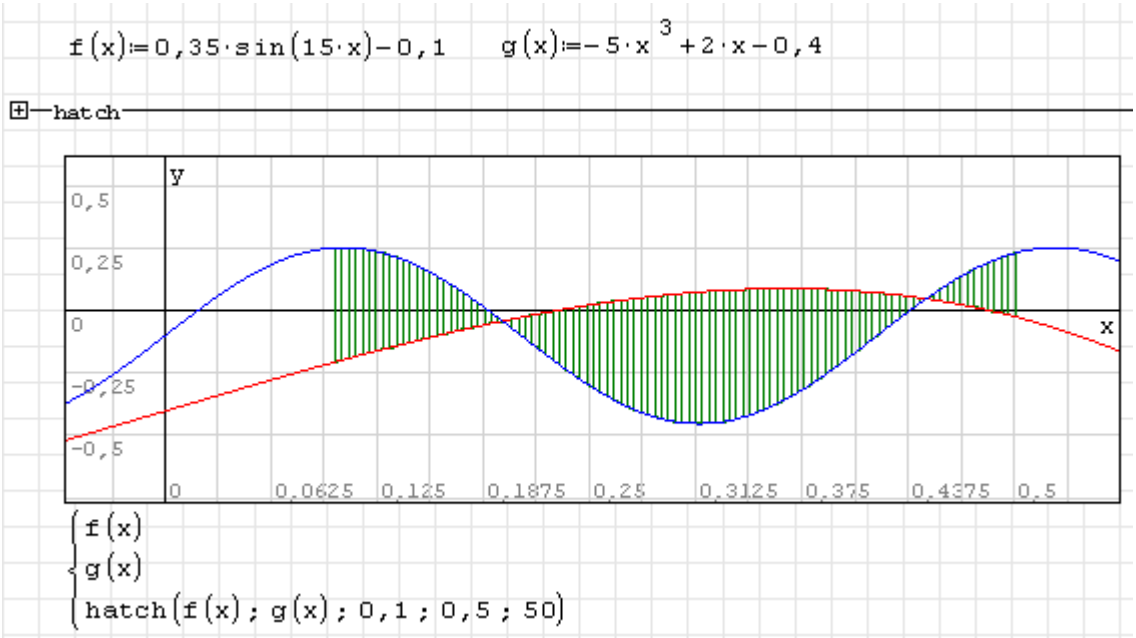
Wird in der dynamischen Auswahl mit ⇆ bestätigt oder in der Baustein-Verwaltung der Haken bei „Code-Baustein in einem Bereich einfügen" stehen gelassen, so wird der Inhalt des Bausteins in einen zugeklappten Blattbereich eingefügt. Der Titel wird gemäß Dateieigenschaften gesetzt.



Alternativ wird der Inhalt direkt ins Arbeitsblatt eingefügt (ohne einklappbaren Blattbereich).

In jedem Fall sind die enthaltenen Definitionen mit ihren Beschreibungen nun der dynamischen Auswahl bekannt und können verwendet werden.




f(x):=0,35·sin(15·x)−0,1
g(x):=−5·x^3+2·x−0,4
hatch
y
x
0,5
0,25
0
−0,25
−0,5
0
0,0625
0,125
0,1875
0,25
0,3125
0,375
0,4375
0,5
f(x)
g(x)
hatch(f(x); g(x); 0,1; 0,5; 50)

8. Der SMath Viewer

Der Inhalt dieses Kapitels basiert auf einem Forum-Beitrag des Entwicklers Andrey Ivashov.

Seit der Version 0.96 kann SMath das aktuelle Rechenblatt als interaktive Anwendung exportieren. Diese Anwendung beinhaltet ein komplettes SMath-System, welches das exportierte Rechenblatt in einem speziellen Modus anzeigt. Darin kann der Anwender Zahlenwerte ändern, hat aber keinen Zugriff auf das Rechenblatt selbst.

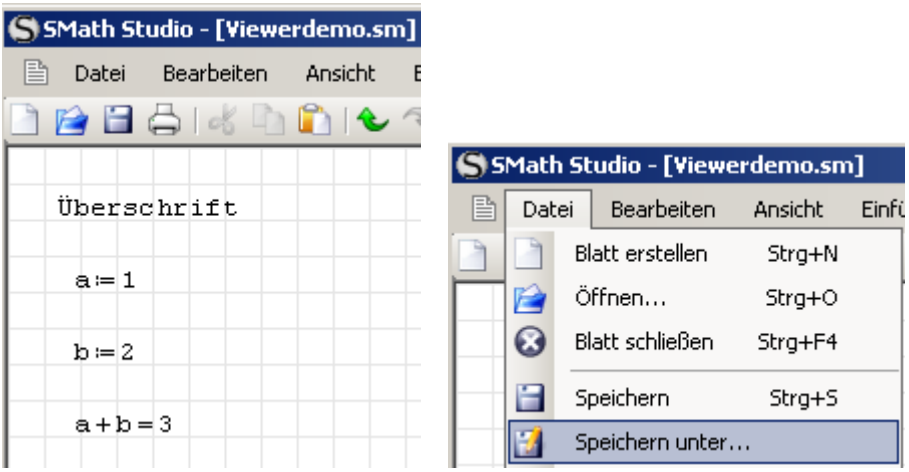
Der Grundgedanke ist, dass SMath Studio (wie auch Mathcad) eine hervorragende Anwendung ist, aber mitunter zu komplex für den normalen Ingenieur sein kann. SMath Studio wurde ursprünglich entwickelt, um die Erstellung beliebig komplexer Rechenblätter durch leistungsfähige Funktionen zu unterstützen. Anwender, die keine Rechenblätter erstellen, sondern dort nur komfortabel Eingaben machen und Ergebnisse ermitteln wollen, sind mitunter mit zu vielen Details in den Rechenblättern konfrontiert.

Im SMath Viewer werden den Anwendern nur die Daten gezeigt, die sie benötigen. Gleichzeitig erfolgt der Export direkt aus SMath Studio Rechenblättern, so dass existierende Anwendungen leicht exportiert werden können.

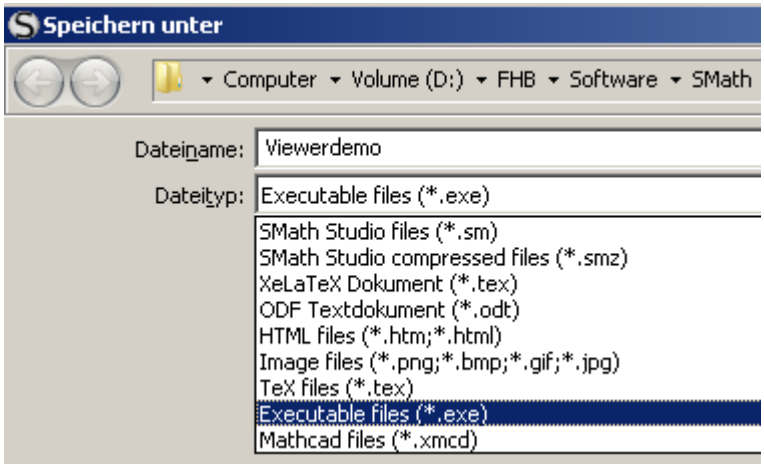
SMath Viewer ist ein Auslieferungsweg für Arbeitsblätter, und SMath Studio die zugehörige Entwicklungsumgebung.

8.1. Einführungsbeispiel

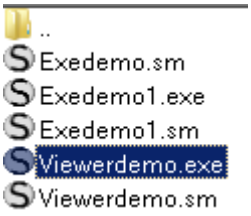
Erzeugen Sie ein beliebiges Arbeitsblatt mit ein paar Variablen, Formeln und etwas Text in SMath Studio und wählen Sie im Hauptmenü **Datei> Speichern unter**



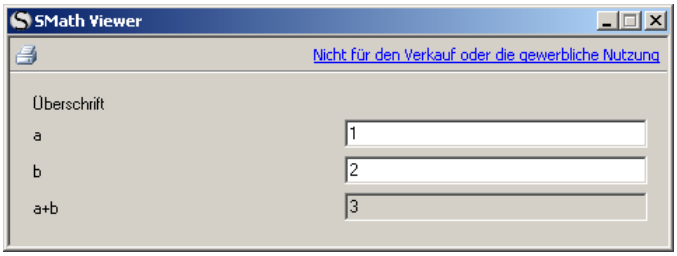
Wählen Sie „Executable files (*.exe)" als Dateityp im Dialog „Speichern unter" aus und drücken Sie Speichern.



In dem von Ihnen gewählten Verzeichnis erscheint nun die einsatzbereite ausführbare Anwendung.



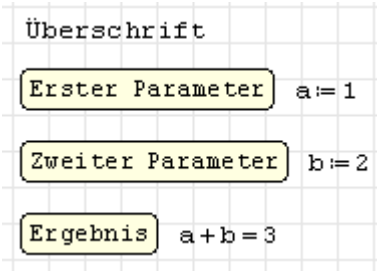
Starten Sie diese mit Doppelklick und Sie sehen die SMath Viewer-Oberfäche mit betriebssystemtypischen Bedienelementen.

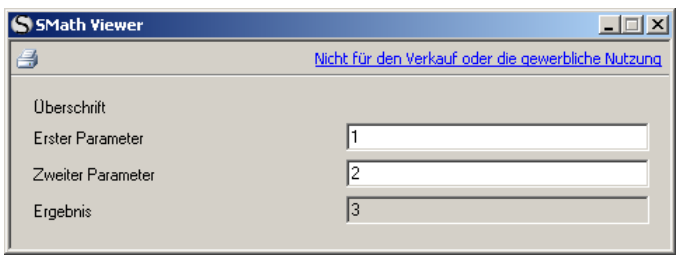


8.2. Gestaltungsmöglichkeiten

Damit das Programm besser bedienbar wird, brauchen wir einige Gestaltungselemente, die SMath Studio auch bereithält. Sie müssen nur ein paar Ergänzungen an Ihrem Rechenblatt vornehmen.

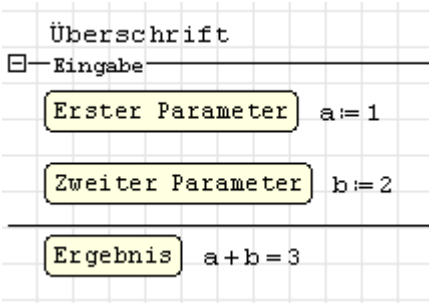
Einige Bereichstypen können mit Beschreibungstexten (siehe Abschnitt 3.12) versehen werden. Diese werden im Viewer anstelle der Formel oder des Variablennamens angezeigt:

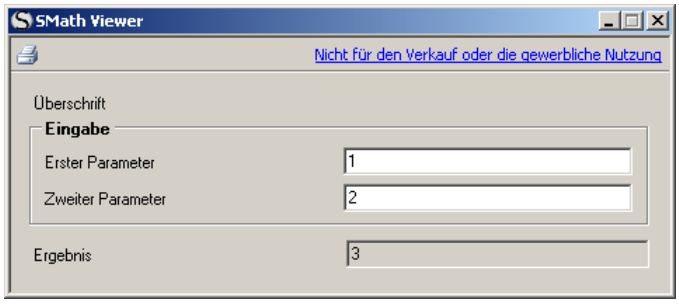




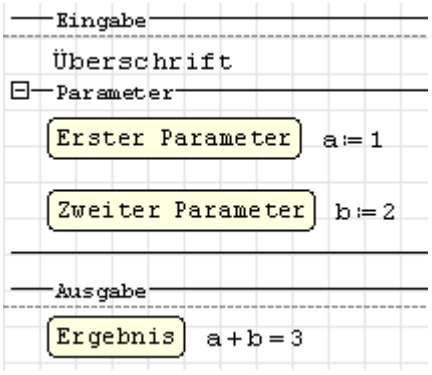
Von den Beschreibungstexten werden dabei die ersten 26 Zeichen und nur die jeweils erste Zeile benutzt.

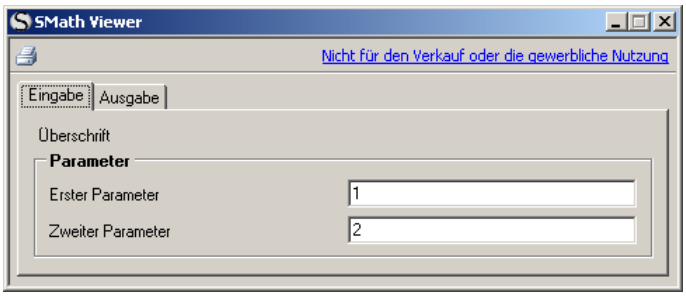
Benutzen Sie Blattbereiche, um deren Inhalt in der Vieweransicht zu gruppieren, die Gruppe kann auch einen Namen bekommen:



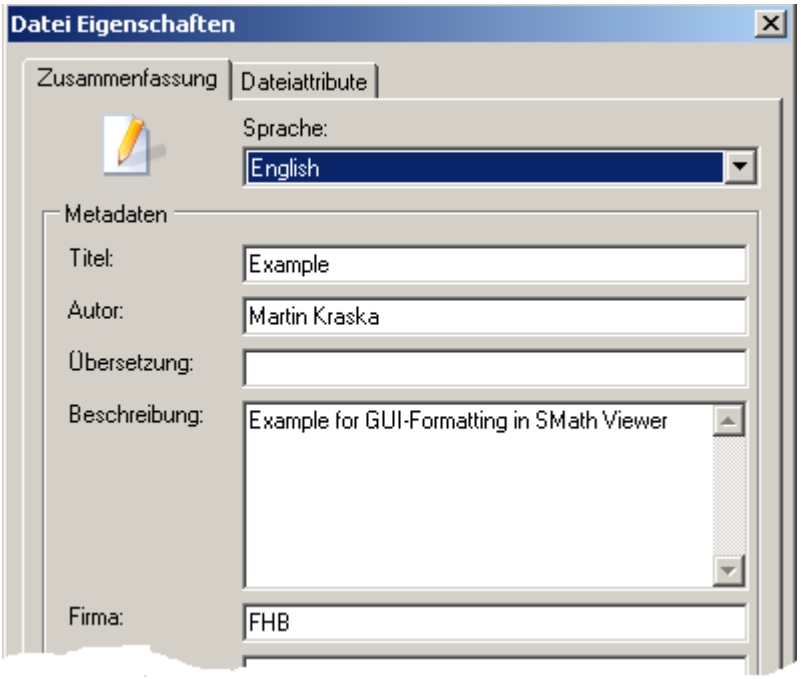


Mit Trennlinien können Sie Ihre Anwendung in mehrere Registerkarten gliedern. Trennlinientitel werden zu Namen der Registerkarten. Fehlt der Titel, werden Nummern vergeben.

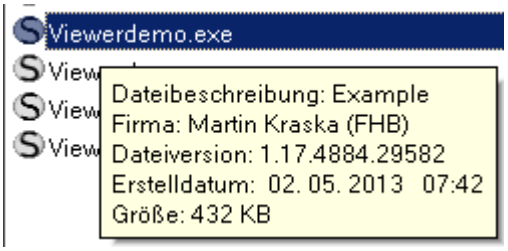




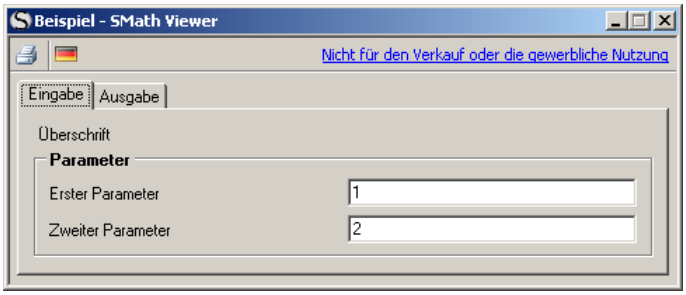
Unter **Hauptmenü> Datei> Eigenschaften** können Sie Angaben zu Ihrem Dokument in verschiedenen Sprachen machen.



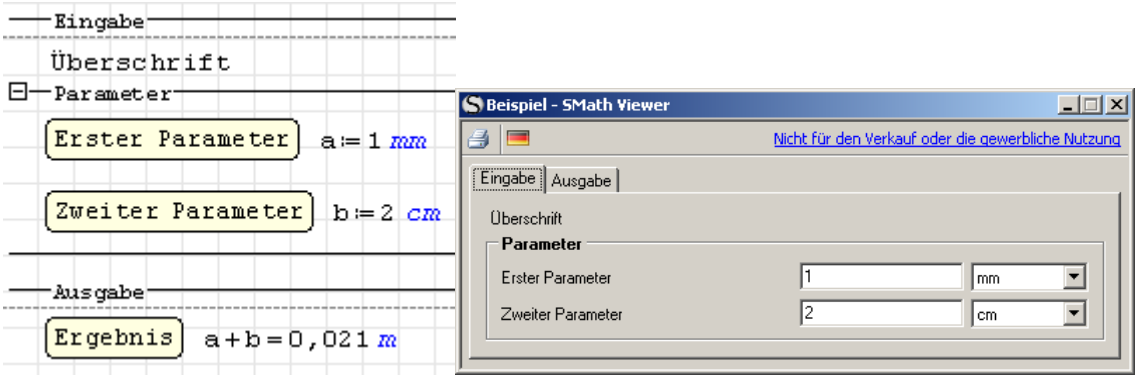
Die englischen Einträge werden im Dateimanager angezeigt.



Der Fenstertitel zeigt den Titel aus den Dateieigenschaften in der eingestellten Sprache an.



Maßeinheiten werden unterstützt. Sowohl bei Ein- als auch bei Ausgabe kann die Maßeinheit aus einer Liste gewählt werden. Diese ist beschränkt auf die dimensionsmäßig passenden Einheiten.



8.3. Anwendungshinweise

Vorteile der Weitergabe von Viewer-Dateien (.exe) gegenüber Rechenblättern (.sm)

- Der Anwender benötigt keine SMath-Installation und keine Erfahrung mit SMath,
- Sie können die Berechnungslogik vor dem Anwender verbergen
- Der Anwender kann die Berechnung nicht versehentlich beschädigen, es wird immer das ausgeführt, was Sie ausgeliefert haben.
- Es kann im Einzelfall zu Geschwindigkeitsvorteilen kommen, da bei Änderungen immer nur der Inhalt der gerade angezeigten Registerkarte neu berechnet wird.
- Trotz des komplett enthaltenen SMath-Systems sind die exe-Dateien vergleichweise kompakt (ab 0,5 MB, je nach Zahl und Größe der benutzten Plugins)

Plugin-Unterstützung.

Die exportierte Viewer-Anwendung enthält alle in der Rechnung benötigten Plugins. Das Verhalten erzeugter Bereiche (z.B. Plots) kann aber vom Verhalten im Rechenblatt abwei-

chen.

Das Maxima-Plugin wird leider nicht unterstützt.

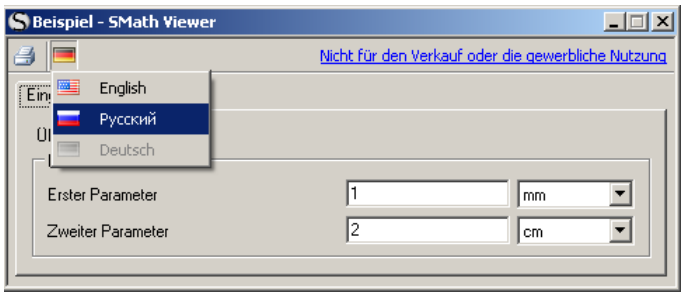
Werden keine externen Plugins benutzt, bleibt die ausführbare Datei unter 0,5 MB.

Sprachanpassung

Sie können Ihre Anwendung mehrsprachig machen, indem Sie bei allen Texten alternative Sprachvarianten angeben.

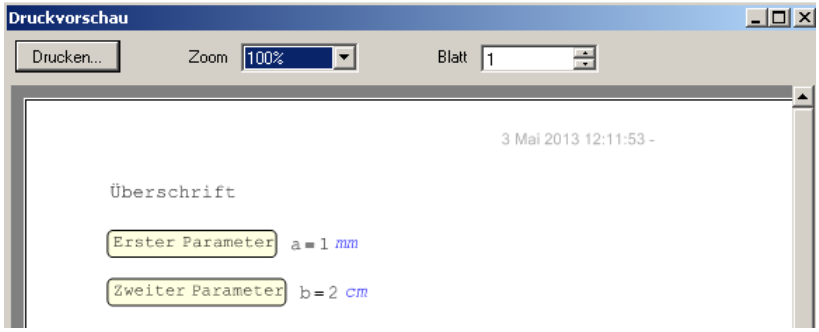
In der Regel kann die voreingestellte Sprache im Kontextmenü des jeweiligen Bereichs oder der Beschreibung geändert werden, so dass die Eingabe alternativer Texte möglich ist. Bei umfangreichen Übersetzungen kann es vorteilhaft sein, vorübergehend die Spracheinstellung in SMath zu ändern.

Der Viewer zeigt die Anwendung standardmäßig passend zur Systemkonfiguration des Benutzers an. Der Schaltknopf in der Werkzeugleiste bietet aber auch die anderen verfügbaren Sprachen an.



Druckansicht

In der Anwendung steht eine Druckansicht zur Verfügung, die der ursprünglichen Rechenblattansicht entspricht.



8.3.1. Lizenz

Der SMath Viewer is gegenwärtig (Mai 2013) für die private Nutzung und für Bildungszwecke kostenlos. Dabei ist der Verkauf oder die kommerzielle Nutzung der erzeugten Anwendungen nicht gestattet, ein entsprechender Hinweis findet sich auf der Benutzeroberfläche.

Der Autor von SMath Studio, Andrey Ivashov, bietet allerdings auch kommerzielle Lizenzen an, deren Konditionen im Einzelfall zu erfragen sind (z.B. über private Mail im Forum).

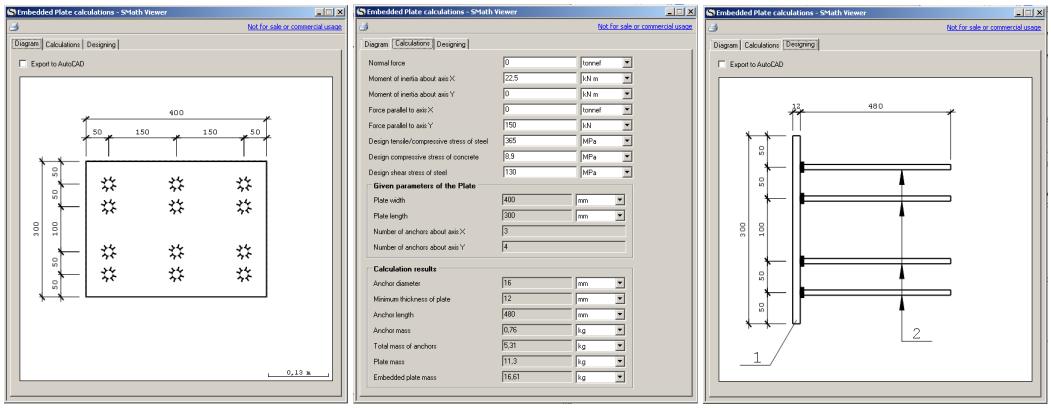
Besonderheiten bei kommerziellen Lizenzen:

- Verkauf und kommerzielle Nutzung von Viewer-Anwendungen ist erlaubt (Beschränkungshinweis in der Oberfläche entfällt)
- Signierung durch ein Code-Zertifikat. Dadurch wird die Datei gegen Modifikation durch Viren oder Hackerangriff geschützt und die sichere Ausführung gewährleistet.
- Das Rechenblatt wird verschleiert, so dass es nicht möglich ist, aus der exe-Datei das ursprüngliche Rechenblatt zu extrahieren.
- Das Anwendungs-Icon kann in jedes gewünschte geändert werden.
- Die Verschleierung des Rechenblatts und die Signierung erfordern eine volle Kontrolle über die eingesetzten Programme und den Kompilationsprozess, der daher auf dem SMath.info-Server online erfolgt und Plugins von Drittanbietern ausschließt.

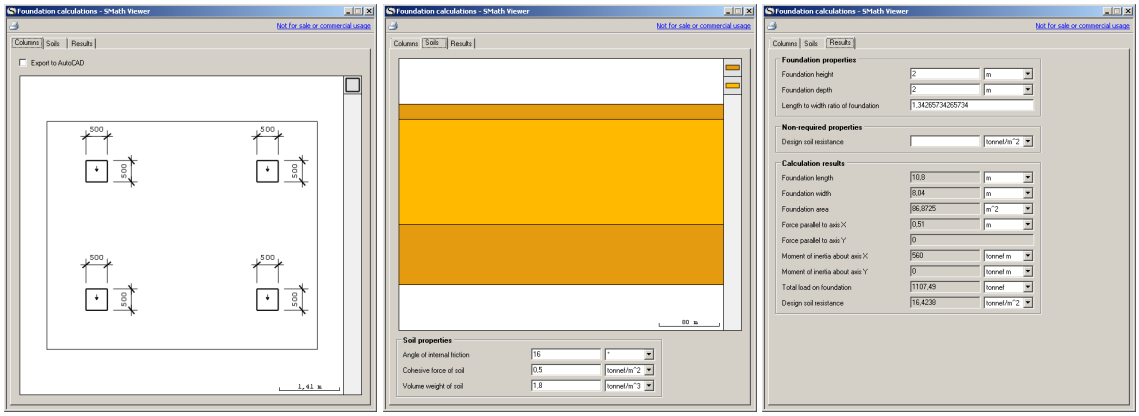
8.4. Beispiele im Netz

Zunächst einige Beispiele mit unveröffentlichtem Rechenblatt (.sm), die das Potenzial der Viewer-Anwendungen demonstrieren sollen.

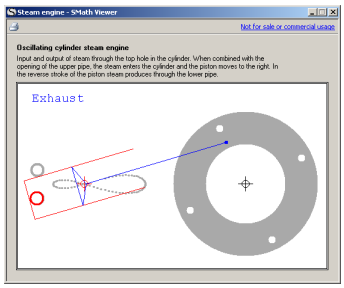
EmbeddedPlate (ENG, RUS) (ENG, RUS). Hier wird das Modeller-Plugin benutzt.



Foundation (ENG, RUS). Die erste Registerkarte enthält interaktive Bedienelemente.



Oscillating.exe (ENG, RUS) Beispiel mit Animation.


Steam engine - SMath Viewer
Not for sale or commercial usage
Oscillating cylinder steam engine
Input and output of steam through the top hole in the cylinder. When combined with the opening of the upper pipe, the steam enters the cylinder and the piston moves to the right. In the reverse stroke of the piston steam produces through the lower pipe.
Exhaust

9. Erweiterungen

Anwender können auf verschiedenen Wegen zur Erweiterung und Verbesserung von SMath beitragen:

- Handbücher schreiben. Der Mangel an ordentlicher Dokumentation ist ein ständiger Begleiter freier Software.
- Beispiele. Diese können die Benutzung bestimmter Funktionen oder die Lösung bestimmter Fragestellungen erläutern.
- Plugins (Anwendungserweiterungen). Das sind dynamisch ladbare Funktionsbibliotheken (dll), die den Funktionsumfang von SMath Studio erweitern.
- Anwendungen. Das sind Rechenblätter, die als eigenständige Programme (SMath Viewer) exportiert wurden.
- Code-Bausteine (sogenannte „snippets"). Diese kann der Anwender in sein Rechenblatt einfügen und dann die darin definierten Funktionen benutzen.
- Übersetzungen. Die Benutzeroberfläche von SMath Studio ist mehrsprachig. Wenn in Ihrer Sprachversion Übersetzungen fehlen oder falsch sind, können Sie das selbst ändern.

Anzeigen der verfügbaren Erweiterungen:

- Erweiterungsmanager (z.B. mit **Extras> Plugins**) und Umschaltung auf die Online Galerie)
- auf der SMath Internetseite unter http://smath.info/?extensions=SMathStudio_Desktop

9.1. Erweiterungsmanager

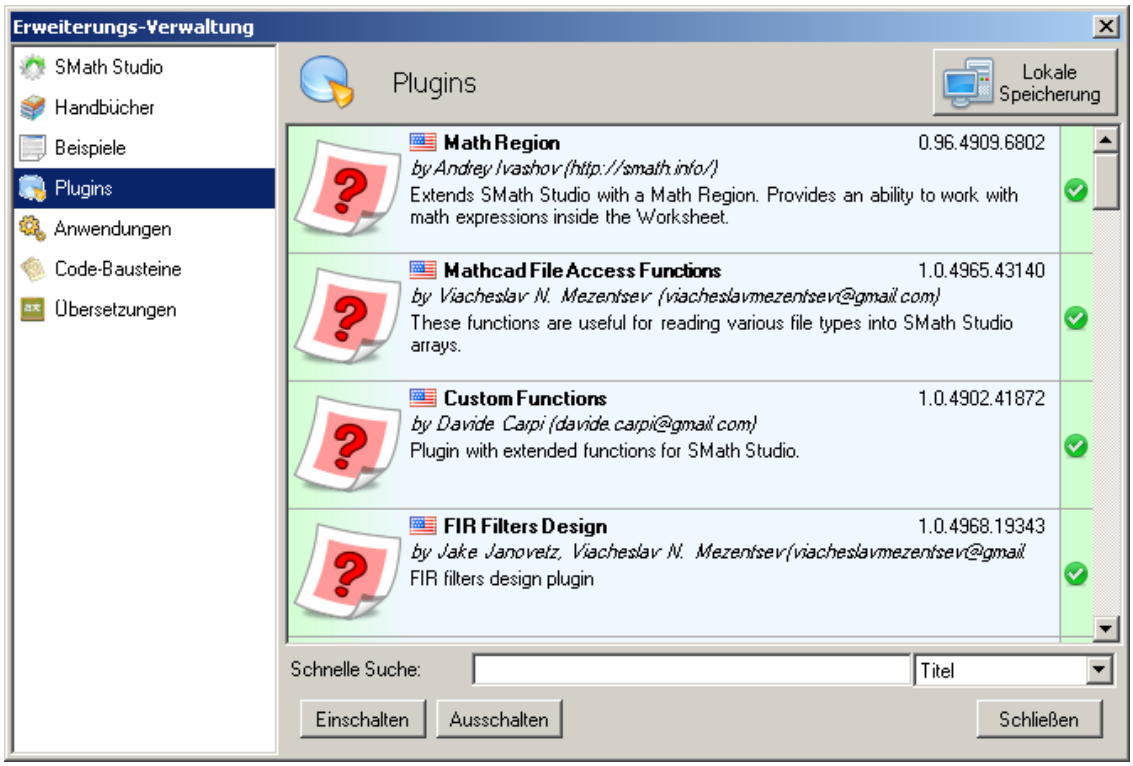
Der Zugriff auf Erweiterungen wird durch den Erweiterungsmanager verwaltet. Dieser wird aufgerufen durch:

Extras> Plugins

Extras> Code-Baustein-Verwaltung

Hilfe> Beispiele

Hilfe> Nach Aktualisierungen suchen



Der Erweiterungsmanager zeigt nach Aufruf die lokal auf Ihrem Rechner gespeicherten Erweiterungen an. Er kann aber auch die online verfügbaren Erweiterungen anzeigen. Zwischen beiden Ansichten schalten Sie mit dem Knopf rechts oben um:

In der Online Galerie werden Ihnen die Erweiterungen angezeigt, die bei Ihnen lokal nicht vorhanden sind oder aktualisiert werden können.

Die heruntergeladenen Erweiterungen werden im Einstellungsverzeichnis gespeichert (siehe Abschnitt 3.17). Sie sind dort unter kryptisch bezeichneten Unterverzeichnissen zu finden. In der Online-Hilfe (**Hilfe> Referenzmaterial> Handbuch> Anwendungserweiterungen (Plugins)**) finden Sie eine Seite mit aktiven Links zu den Plugin-Verzeichnissen.

9.2. Erweiterungen hochladen

Ablauf

1. Erweiterung in das erforderliche Format bringen und die Metadaten bereithalten
2. Registrierung im Portal http://smath.info/upload/Extensions.aspx. Dazu müssen Sie gegebenenfalls ein Nutzerkonto mit Name und Passwort anlegen.
   a) Erweiterungstyp wählen. Abhängig davon erscheinen rechts Hinweise. Diese sind unbedingt zu beachten.
   b) Metadaten ausfüllen (typabhängig).
   c) Datei wählen

d) Gegebenenfalls Passwort für den Zugriff über den Erweiterungsmanager setzen

e) Registrierung starten mit Register

3. Bei neuen Erweiterungen: Freigabe durch den Entwickler (Andrey Ivashov) abwarten. Updates benötigen keine Freigabe.
4. Die Erweiterung ist sichtbar unter Online Galerie im Erweiterungsmanager des Programms oder unter http://smath.info/?extensions=SMathStudio_Desktop

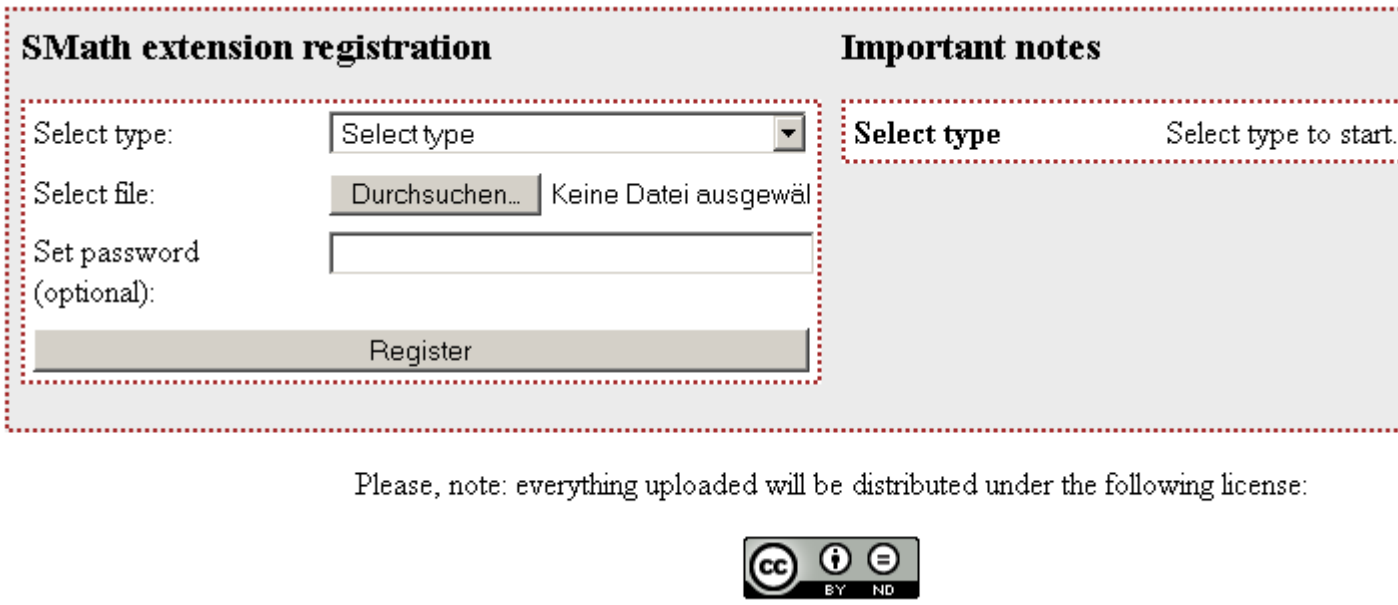


Abbildung 9.1.: Portal zum Registrieren von Erweiterungen: http://smath.info/upload/Extensions.aspx

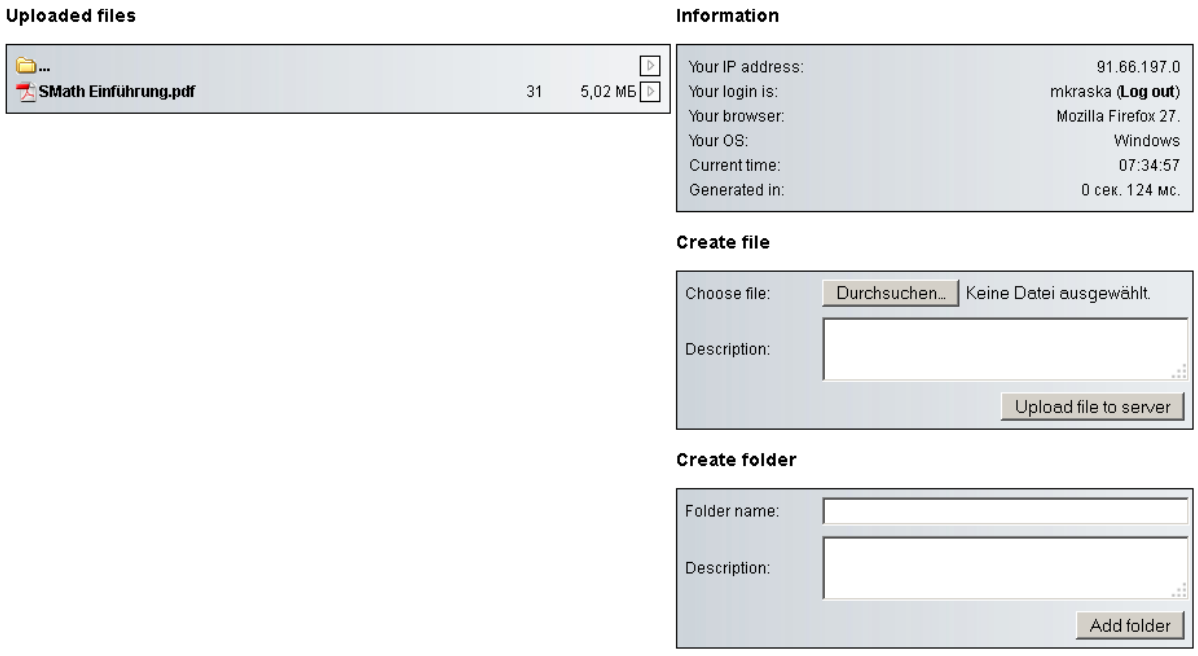


Abbildung 9.2.: Portal zum Verwalten der eigenen hochgeladenen Erweiterungen (z.B. Löschen oder Download-Statistik ansehen) http://smath.info/upload/default.aspx

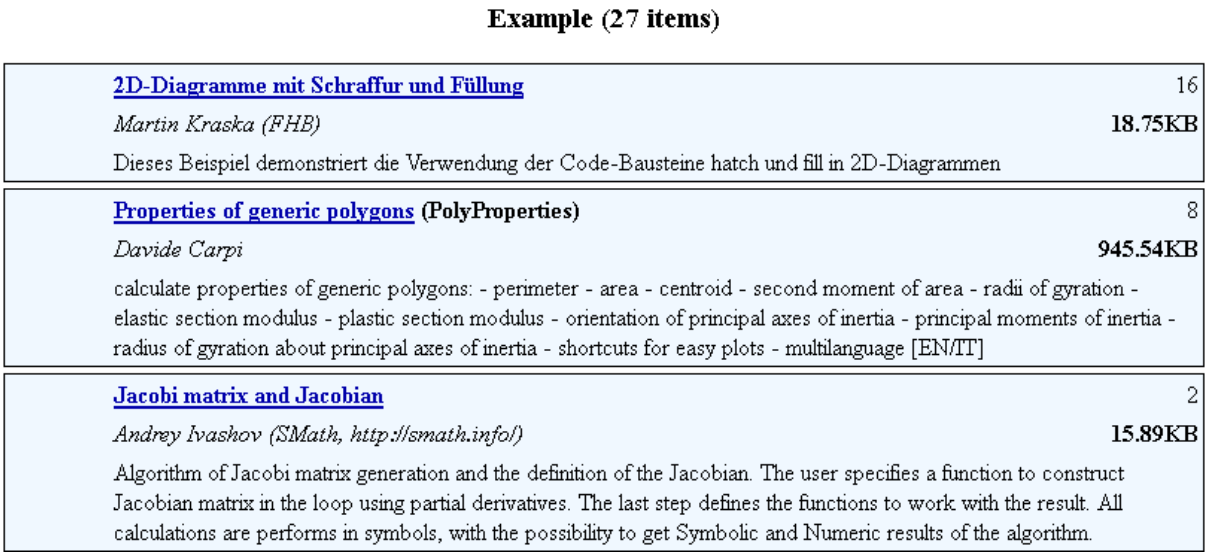


Abbildung 9.3.: Portal mit Links auf die aktuellen Versionen aller hochgeladenen Erweiterungen http://smath.info/?extensions=SMathStudio_Desktop

9.3. Dateiformate und Metadaten

Bei der Registrierung von Dateien gibt es je nach Dateityp unterschiedliche Wege, eine eindeutige Identität und Version zu erkennen.

Jedes Rechenblatt hat eine eigene Identifikationsnummer (ID) und Version (*revision*). Diese kann man in der .sm-Datei mit einem Editor anzeigen:

```
<regions>
  <settings>
    <identity>
      <id>94874e93-4882-4a0c-b0b6-4aba69703565</id>
      <revision>5</revision>
    </identity>
```

Die ID wird beim erstmaligen Abspeichern vergeben.

Achtung: Dateien, die durch **Datei> Speichern unter** oder im Betriebssystem kopiert werden, haben die gleiche ID!

Neue, eindeutige ID erzeugen:

1. Datei mit einem Editor öffnen, ganzen `<identity>...</identity>` Block löschen und Datei speichern.
2. Datei in SMath Studio laden und wieder speichern. Der Identifizierungsblock wird neu angelegt.

Die Version wird bei jedem Speichern in SMath Studio hochgezählt.

Example (Beispiele)

Dateiformat	Rechenblatt (.sm oder .smz)
Metadaten	*Titel, Autor* und *Beschreibung* unter **Datei> Eigenschaften** in SMath Studio in mindestens einer Sprache ausfüllen.
Identifikation	`id` im `identity`-Block in der .sm-Datei
Version	`revision` im `identity`-Block in der .sm-Datei

Plugin (Anwendungserweiterung)

Dateiformat	**Unkomprimierte** .zip-Datei mit beliebigen Ordnern und Dateien, mindestens eine .dll im Wurzelverzeichnis
SMath-Version	Auswahlliste, erforderliche SMath-Programmversion.
Metadaten	Eigenschaften in der Entwicklungsumgebung für die .dll setzen.
Identifikation	
Version	

Applications (Exportierte Anwendungen)

Diese werden als Rechenblatt hochgeladen und auf dem Server kompiliert. Dabei steht nur eine Untermenge der existierenden Plugins zur Verfügung (Näheres auf Anfrage beim Entwickler).

Dateiformat	Rechenblatt (.sm oder .smz)
Metadaten	Dateieigenschaften unter **Datei> Eigenschaften** in SMath Studio in mindestens einer Sprache ausfüllen.
Identifikation	`id` im `identity`-Block in der .sm-Datei
Version	`revision` im `identity`-Block in der .sm-Datei

Snippet (Code-Baustein)

Dateiformat	Rechenblatt (.sm oder .smz)
Metadaten	*Titel, Autor* und *Beschreibung* unter **Datei> Eigenschaften** in SMath Studio in mindestens einer Sprache ausfüllen. Dabei muss das *Tastaturkürzel* angegeben werden. Unter diesem Namen ist der Baustein in der dynamischen Hilfe bekannt.
Identifikation	`id` im `identity`-Block in der .sm-Datei
Version	`revision` im `identity`-Block in der .sm-Datei

Handbook (Handbuch)

Dateiformat	PDF
Metadaten	Dokumentsprache (Auswahlliste) Dokumenttitel (erscheint in der Galerie) Dokumentbeschreibung (unterer Textteil, wird umgebrochen) Autor (wird nicht umgebrochen)
Identifikation	GUID, erzeugt aus dem Dateinamen des Dokuments. Updates müssen daher den gleichen Namen haben.
Version	JJJJ.MM.TT.SS (Jahr.Monat.Tag.Stunde) beim Hochladen.

Book (Verlinkte Rechenblätter)

Dateiformat	**Unkomprimierte** .zip-Datei mit beliebigen Ordnern und Dateien, *contents.sm* muss im Wurzelverzeichnis liegen.
Metadaten	Dateieigenschaften unter **Datei> Eigenschaften** für *contents.sm* in SMath Studio in mindestens einer Sprache ausfüllen.
Identifikation	`id` im `identity`-Block in *contents.sm*
Version	`revision` im `identity`-Block in *contents.sm*

9.4. Plugins

Plugins sind Programmergänzungen, die als kompilierte Anwendungserweiterung (dynamisch anbindbare Bibliothek, .dll) bereitgestellt werden und den Funktionsumfang von SMath ergänzen. Sie können über den Erweiterungsmanager installiert werden.

Der Autor dieses Handbuchs hat eine inoffizielle portable Distribution zusammengestellt, die sowohl unter Linux als auch unter Windows funktionieren sollte und die meisten der in diesem Kapitel beschriebenen Plugins bereits enthält.

Was können Plugins bieten:

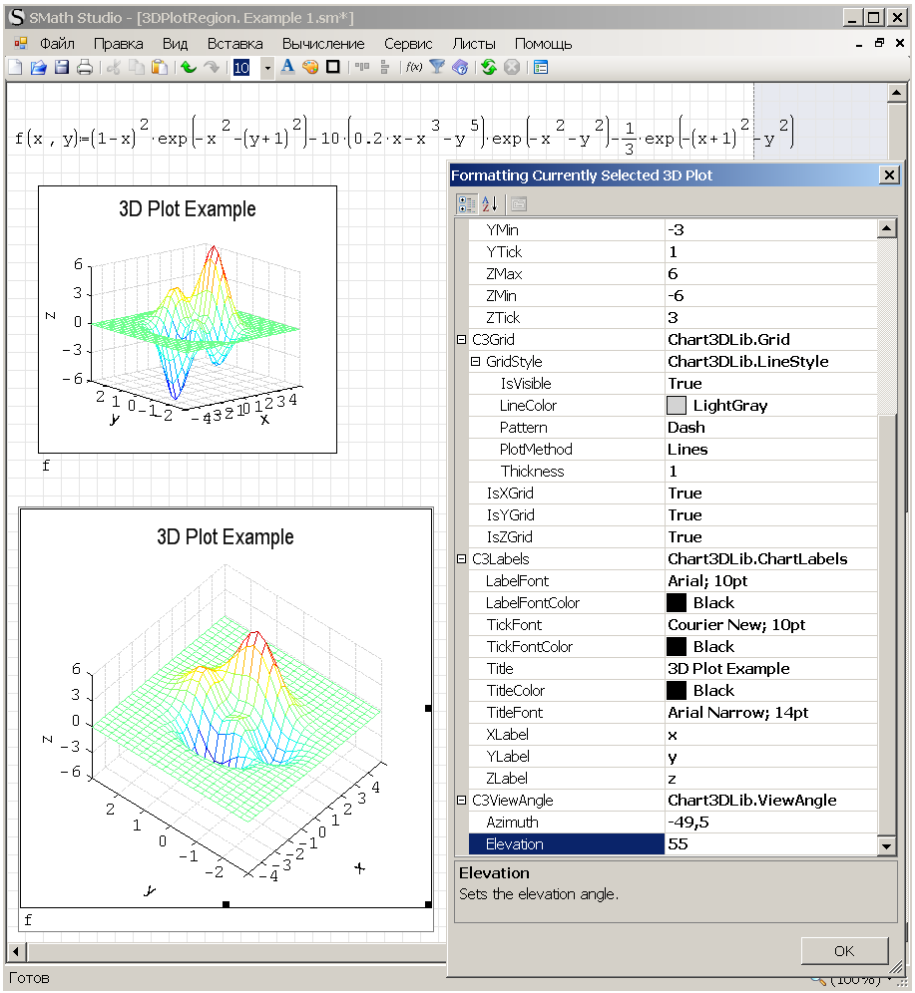
- Zusätzliche Funktionen. Diese werden in der dynamischen Hilfe aufgelistet, in der Regel mit einer Kurzbeschreibung.
- Zusätzliche Bereichstypen. Diese sind dann unter **Hauptmenü> Einfügen** verfügbar.
- Zusätzliche Exportformate. Diese erscheinen als mögliche Dateitypen unter **Hauptmenü> Datei> Speichern unter**

In der Regel ist die Dokumentation nur im SMath-Forum zu finden. Daher wird in diesem Abschnitt einiges zu den Funktionen gesagt, die von den Plugins bereitgestellt werden.

Die im folgenden erläuterten Plugins sind nicht Teil der Standardinstallation von SMath, sondern müssen mit dem Erweiterungsmanager (siehe Abschnitt) installiert werden, wenn Sie nicht die inoffizielle portable Distribution benutzen.

9.4.1. 3D Plot Region von Viacheslav N. Mezentsev

Das Plugin stellt einen 3D-Plotbereich zur Verfügung. Er befindet sich noch am Anfang der Entwicklung und ist momentan noch nicht wirklich nutzbar, insbesondere weil sich die Einstellungen der Bereiche (Achsenbereiche, Ansicht) nicht mit dem Rechenblatt speichern lassen.



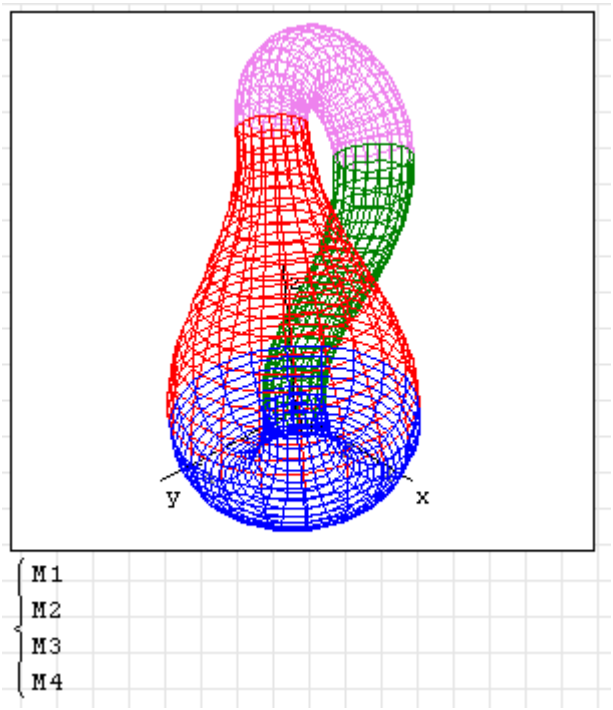
Die Funktion `CreateMesh()` erzeugt 3D Gitternetze aus parametrisch gegebenen Funktionen, so dass man man diese mit dem Standard-3D-Diagrammbereich darstellen kann. Hier zum Beispiel die Kleinsche Flasche:

$$a\coloneqq 2{,}5 \qquad b\coloneqq 1{,}5 \qquad c\coloneqq 2 \qquad d\coloneqq 3$$

$$S1(u\,;v)\coloneqq\begin{pmatrix}(a+b\cdot\cos(u))\cdot\cos(v)\\(a+b\cdot\cos(u))\cdot\sin(v)\\-a\cdot\sin(u)\end{pmatrix} \qquad S2(u\,;v)\coloneqq\begin{pmatrix}(a+b\cdot\cos(u))\cdot\cos(v)\\(a+b\cdot\cos(u))\cdot\sin(v)\\d\cdot u\end{pmatrix}$$

$$S3(u\,;v)\coloneqq\begin{pmatrix}c-c\cdot\cos(u)+\sin(v)\\\cos(v)\\d\cdot u\end{pmatrix} \qquad S4(u\,;v)\coloneqq\begin{pmatrix}c+(c+\cos(v))\cdot\cos(u)\\\sin(v)\\d\cdot\pi+(2+\cos(v))\cdot\sin(u)\end{pmatrix}$$

$$M1\coloneqq \mathrm{CreateMesh}(S1(u\,;v)\,;0\,;\pi\,;0\,;2\cdot\pi\,;20\,;20)$$
$$M2\coloneqq \mathrm{CreateMesh}(S2(u\,;v)\,;0\,;\pi\,;0\,;2\cdot\pi\,;20\,;20)$$
$$M3\coloneqq \mathrm{CreateMesh}(S3(u\,;v)\,;0\,;\pi\,;0\,;2\cdot\pi\,;20\,;20)$$
$$M4\coloneqq \mathrm{CreateMesh}(S4(u\,;v)\,;0\,;\pi\,;0\,;2\cdot\pi\,;20\,;20)$$



Beispiele, Download

9.4.2. AlgLib-Schnittstelle von Viacheslav N. Mezentsev

Alglib ist eine plattformunabhängige und quelloffene Bibliothek für numerische Verfahren und Datenanalyse. Das Plugin erschließt beispielhaft einige wenige Funktionen dieser Bibliothek für die Anwendung in SMath.

Das sind im einzelnen:

- `al_airy()` Airy-Funktionen und ihre Ableitungen $Ai(x)$, $Ai'(x)$, $Bi(x)$ und $Bi'(x)$.
- `al_beta()` Beta-Funktion
- `al_convr1d()` Eindimensionale Faltung (Konvolution) zweier reeller Vektoren
- `al_convr1dinv()` Inverse eindimensionale Faltung (Entfaltung, Dekonvolution) zweier reeller Vektoren
- `al_fftc1d()` Schnelle Fourier-Transformation (komplex)
- `al_fftc1dinv()` Inverse Schnelle Fourier-Transformation (komplex)
- `al_neqsolve()` Nichtlinearer Löser nach dem Levenberg-Marquardt-Verfahren

9.4.3. Custom Functions - Funktionspaket von Davide Carpi

Das Plugin stellt diverse Funktionen bereit. Diskussionsforum für Fragen und Fehlermeldungen.

Funktion	Seite	Verwendung
`Clear()`	267	Definition von Variablen löschen
`Diag()`	270	Diagonalmatrix erzeugen

eOoM()	272	Größenordnung (*order of magnitude*)
eOoM.10()		10 hoch Größenordnung (*order of magnitude*)
GetType()	285	Variablentyp feststellen
InterpBilinear()		Bilineare Interpolation
Max()	306	Maximum beliebig verschachtelter Objekte
mat2sys()	305	Matrix in Liste umwandeln
mat2sys.1()	305	Matrix in Liste umwandeln (nur oberste Ebene)
Min()	309	Minimum beliebig verschachtelter Objekte
Ones()	314	Eins-Matrix
OoM()	315	Größenordnung, Variante (*order of magnitude*)
OoM.10()		10 hoch Größenordnung, Variante (*order of magnitude*)
SettingsDirectory()	332	Einstellungsverzeichnis ermitteln
strtolower()	337	Zeichenkette in Kleinbuchstaben umwandeln
strtoupper()	337	Zeichenkette in Großbuchstaben umwandeln
sys2mat()	339	Liste in Matrix umwandeln
ucfirst()	340	Ersten Buchstaben in Großbuchstaben umwandeln
ucwords()	340	Wortanfänge in Großbuchstaben umwandeln
UoM()	340	SMath Basiseinheit (*unit of measurement*)
Zeros()	344	Null-Matrix

9.4.4. Data Exchange von Davide Carpi

Download

Das Plugin ergänzt den Menüpunkt Datei> Speichern unter: um das OpenOffice Textformat (odt) und LaTeX (tex). Das odt-Format kann auch mit MS Word gelesen werden, allerdings wird die Formatierung von Formeln nicht gut übernommen.

Das Plugin stellt Funktionen zum Datenexport in csv (Comma Separated Values) und zum Import und Export von ods (Open Office Calc) Dateien bereit. Ebenfalls enthalten ist eine Funktion zum Export von Ausdrücken in das Open Office Formel-Format (odf).

Import und Export des xlsx-Formats wird durch ein separates Plugin (siehe Abschnitt 9.4.13) realisiert, welches nur für Windows zur Verfügung steht.

Die Optionen zum Datenaustausch werden im Zusammenhang mit Matrizen in Abschnitt 4.9.7 erläutert. Eine Übersicht über alle Import- und Exportfunktionen findet sich in Abschnitt 3.18.

Funktion	Seite	Verwendung
`ExportData.CSV()`	275	Export von Ausdrücken als csv (Comma Separated Values)
`ExportData.ODF()`	276	Export von Ausdrücken als odf (Open Office Formel)
`ExportData.ODS()`	277	Export von Ausdrücken als ods (Open Office Calc)
`ImportData.ODS()`	298	Import von ods-Daten (Open Office Calc)

Die XLSX-Funktionen stehen unter Linux nicht zur Verfügung.

9.4.5. Excel Collaboration von Andrey Ivashov

Das Plugin erlaubt das Lesen und Schreiben von Zellen in Excel-Dateien.

Download, Beispiele

Funktion	Seite	Verwendung
`ExportCell()`	275	Schreibt Werte in Zellen existierender xls-Dateien (MS Excel)
`ImportCell()`	276	Liest Werte aus Zellen in xls-Dateien (MS Excel)

9.4.6. Functions Extensions von Davide Carpi

Das Plugin stellt zusätzliche Funktionen mit eigenen Operatorformen bereit.

- `at()` Substitutionsfunktion (Seite 253)
- `cases()` Fallunterscheidung (Seite 256)
- ternäre Vergleichsoperatoren der Form $a < b < c$

9.4.7. Image Region von Viacheslav N. Mezentsev

Das Plugin stellt einen Bereichstyp zur Anzeige von Pixeldaten aus Bilddateien oder aus Matrizen bereit. Sehen Sie dazu auch den Abschnitt 3.7. Die zugehörige Forumdiskussion finden Sie hier.

9.4.8. Maple-Schnittstelle von Viacheslav N. Mezentsev

Das Plugin ist eine Schnittstelle zur DOS-Version des Computeralgebrasystems MapleV, die im Download mitgeliefert wird. Die symbolischen Fähigkeiten von MapleV sind wesentlich besser als die von SMath selbst.

Vorteil gegenüber dem Maxima-Plugin: Die erforderliche MapleV-Version ist Bestandteil der Installation über die Erweiterungsverwaltung, es ist muss also keine extra Software installiert werden.

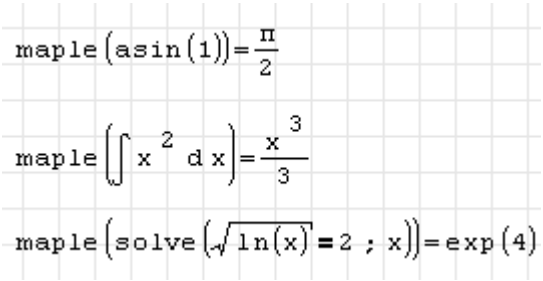
Nachteil: Die Rechtefrage zur benutzten MapleV-Version ist zumindest zweifelhaft, daher ist das Plugin auch nicht Bestandteil der inoffiziellen portablen SMath-Distribution.

Das Plugin stellt die Funktion `maple(Ausdruck)` bereit, die je nach Ergebnistyp symbolisch oder numerisch (wenn das Ergebnis keine unbekannten Bezeichner enthält) ausgewertet wird.

Ablauf beim Funktionsaufruf

1. Der Ausdruck wird in Zeilenform gebracht und auf Maple-Syntax transformiert.
2. Die so erhaltene Zeile wird in eine temporäre Textdatei geschrieben.
3. Maple wird mit der Textdatei als Parameter aufgerufen.
4. Maple schreibt das Ergebnis in eine Textdatei.
5. Die Ergebnisdatei wird gelesen, auf SMath Syntax transformiert und an SMath zurückgegeben.

Beispiele



9.4.9. Mathcad File Access von Viacheslav N. Mezentsev

Das Plugin stellt Mathcad-kompatible Funktionen für den Zugriff auf Datenfiles bereit.

Funktion	Seite	Verwendung
`READBIN()`	327	Lesen von Binärdaten
`WRITEBIN()`	343	Schreiben von Binärdaten
`READWAV()`	328	Lesen von .wav-Dateien
`WRITEWAV()`	343	Schreiben von .wav-Dateien
`GETWAVINFO()`	285	Lesen von .wav-Dateiinformationen
`READ_IMAGE()`	326	Einlesen eines Bildes als Grauwertmatrix
`READ_RED()`	326	Einlesen des Rot-Anteils eines Bildes
`READ_GREEN()`	326	Einlesen des Grün-Anteils eines Bildes
`READ_BLUE()`	325	Einlesen des Blau-Anteils eines Bildes
`READBMP()`	327	Einlesen eines .bmp-Bildes als Grauwertmatrix
`READRGB()`	328	Einlesen eines Bildes als drei Matrizen mit RGB-Werten
`GetFolderPath()`	284	Ermittelt Systempfade

`CurrentDirectory()`	268	Ermittelt oder setzt das aktuelle Verzeichnis
`DocumentDirectory()`		Verzeichnis des aktuell offenen Dokuments

9.4.10. Non Linear Solvers von Davide Carpi

Das Plugin stellt nichtlineare Löser sowie Optimierungsalgorithmen für Gleichungen und Gleichungssysteme bereit. Die meisten Algorithmen kommen auch im Klartext als SMath-Programme, was für den Test der Plugin-Funktionen benutzt wird.

Die Löser und Optimierer erwarten als erstes Argument einen Satz von Funktionen (oder eine einzelne Funktion). Als unabhängige Variable werden in der Funktionsdefinition alle vorkommenden Variablen betrachtet, die nicht definiert sind, und zwar in alphabetischer Reihenfolge. Weitere Argumente sind je nach Verfahren:

$\underline{x}_0$	Vektor der Startwerte für die unabhängigen Variablen (in alphabetischer Reihenfolge)
$\underline{x}_L$	Vektor der Bereichsuntergrenzen für die unabhängigen Variablen (in alphabetischer Reihenfolge)
$\underline{x}_H$	Vektor der Bereichsobergrenzen für die unabhängigen Variablen (in alphabetischer Reihenfolge)
$\underline{\varepsilon}$	Vektor der Genauigkeit für die Funktionen. Bei den Funktionen mit Bereichsangaben ($\underline{x}_L$, $\underline{x}_H$) ist das eine Liste aus Genauigkeit für die Funktionswerte und einer optionalen Genauigkeit für die Variablen.
$n_{\max}$	Maximale Anzahl der Iterationen
h	Schrittweite für den zentralen Differenzenquotienten bei Verfahren mit dem Zusatz „CD”

Ein letztes optionales Argument zeigt an, ob die Zahl der Iterationen als Ergebnis mitgeliefert werden soll (es muss dann vorhanden und von Null verschieden sein).

Beim Umgang mit diesen Funktionen ist zu beachten:

- Die Löser finden die Nullstellen von Ausdrücken $\underline{f}$ (diese sind stets als erstes Argument anzugeben). Diese Funktionen sind als erstes Argument anzugeben.
- Aufgelöst wird nach allen enthaltenen, nicht definierten Variablen. Diese werden automatisch identifiziert und alphabetisch sortiert (Vektor $\underline{x}$). Man kann dies mit der Funktion `Unknowns()` prüfen. Es ist also nicht erforderlich, die zu suchenden Variablen anzugeben. Man kann aber die Startwerte oder die Untergrenzen in der Form `Variable=Wert` (boolsches Gleichheitszeichen) angeben. Dann wird die eingebettete Zuweisung aktiviert, die Lösung wird auf die betreffende Variable zugewiesen. In diesem Fall ist auch die Reihenfolge der Variablen unerheblich.
- Es gibt Löser, die einen Startpunkt $\underline{x}_0$ benötigen und solche, die ein Intervall benötigen, begrenzt durch $\underline{x}_L$ und $\underline{x}_H$.
- Startwerte oder Intervallgrenzen werden durch Vektoren oder Listen angegeben, deren Werte den Unbekannten in alphabetischer Reihenfolge ihrer Namen zugeordnet werden. Die Werte sind einheitenmäßig passend zur jeweiligen Variable anzugeben.
- Die geforderte Genauigkeit $\underline{\varepsilon}$ ist als Toleranz auf den Ausdruck anzugeben, muss also einheitenmäßig zum Ausdruck passen. Wird die Maßeinheit weggelassen, dann wird die Basiseinheit der entsprechenden Dimension verwendet.

- In fast allen Lösern kann die maximale Zahl der Interationen $n_{\max}$ angegeben werden. Ausnahme: `Bisection()`, diese Funktion arbeitet mit dem Halbteilungsverfahren, dessen Iterationszahl durch die geforderte Genauigkeit festliegt.
- Die Funktion `NewtonRaphson.CD()` verwendet den zentralen Differenzenquotienten als Näherungswert für den Gradienten. In diesem Fall muss die Schrittweite h des Differenzenquotienten angegeben werden.
- Ein optionales letztes Argument s steuert die Ausgabe des Ergebnisses. Ist der Wert von Null verschieden, dann wird zusätzlich die Zahl der benötigten Iterationen (Funktionsaufrufe) ausgegeben.

Folgende Funktionen werden bereitgestellt:

Nichtlineare Löser		
`BDQRF()`	$\underline{x}_L$, $\underline{x}_H$,$\underline{\varepsilon}$, $n_{\max}$	255
`Bisection()`	$\underline{x}_L$, $\underline{x}_H$,$\underline{\varepsilon}$	255
`Brent()`	$\underline{x}_L$, $\underline{x}_H$,$\underline{\varepsilon}$, $n_{\max}$	256
`Broyden()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	256
`FindRoot()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	278
`HRE.B()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	291
`HRE.NR()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	291
`HRE.RK()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	291
`NewtonRaphson()`	$\underline{x}_0$, $\underline{\varepsilon}$, $n_{\max}$	312
`NewtonRaphson.CD()`	$\underline{x}_0$, h, $\underline{\varepsilon}$, $n_{\max}$	313
`Ridder()`	$\underline{x}_L$, $\underline{x}_H$,$\underline{\varepsilon}$, $n_{\max}$	330
`Secant()`	$\underline{x}_L$, $\underline{x}_H$,$\underline{\varepsilon}$, $n_{\max}$	331
Optimierer		
`GaussNewton()`	$\underline{x}_0$, ε, $n_{\max}$	281
`GaussNewton.CD()`	$\underline{x}_0$, h, ε, $n_{\max}$	283
`GaussNewton.CDGSS()`	$\underline{x}_0$, h, ε, $n_{\max}$	283
`GaussNewton.GSS()`	$\underline{x}_0$, ε, $n_{\max}$	284
`GoldenSectionSearch.max()`	$\underline{x}_L$, $\underline{x}_H$,?, $n_{\max}$	285
`GoldenSectionSearch.min()`	$\underline{x}_L$, $\underline{x}_H$,?, $n_{\max}$	286
`GradientAscent()`	$\underline{x}_0$, ε, $n_{\max}$	288
`GradientAscent.GSS()`	$\underline{x}_0$, ε, $n_{\max}$	288
`GradientDescent()`	$\underline{x}_0$, ε, $n_{\max}$	289
`GradientDescent.GSS()`	$\underline{x}_0$, ε, $n_{\max}$	289
`LevenbergMarquardt()`	$\underline{x}_0$, ε, $n_{\max}$	303
`LevenbergMarquardt.CD()`	$\underline{x}_0$, h, ε, $n_{\max}$	303
`NCGM()`	$\underline{x}_0$, ε, $n_{\max}$	311
`NCGM.CD()`	$\underline{x}_0$, h, ε, $n_{\max}$	311
`NewtonMethod()`	$\underline{x}_0$, ε, $n_{\max}$	311
`NewtonMethod.CD()`	$\underline{x}_0$,h, ε, $n_{\max}$	311
`NewtonMethod.CDGSS()`	$\underline{x}_0$,h, ε, $n_{\max}$	302
`NewtonMethod.GSS()`	$\underline{x}_0$, ε, $n_{\max}$	312
Hilfsfunktionen		
`Diag()`	Diagonalmatrix	270
`Gradient()`	Gradient	288
`Hessian()`	Hesse-Matrix	290

`Hessian.CD()`	Hesse-Matrix mit Zentralem Differenzenquotienten	291
`Jacobian()`	Jacobi-Matrix	302
`Jacobian.CD()`	Jacobi-Matrix mit Zentralem Differenzenquotienten	302
`mapUnknowns()`	Undefinierte Variablen identifizieren	305

9.4.11. ODE Solvers von Viacheslav N. Mezentsev

Das Plugin stellt das Runge-Kutta-Verfahren zur Lösung von Systemen gewöhnlicher Differenzialgleichungen bereit.

Beispiele

Funktion	Seite	Verwendung
`Rkadapt()`	330	Runge-Kutta-Fehlberg-Verfahren mit adaptiver Schrittweite
`rkfixed()`	331	Runge-Kutta-Verfahren mit fester Schrittweite

9.4.12. Statistical Tools von Davide Carpi

Das Plugin bietet eine umfangreiche Funktionssammlung. Es gliedert sich in die Gruppen

- Deskriptive Statistik
- Verteilungsfunktionen
- Dichtefunktionen
- Zufallszahlen
- Hilfsfunktionen

Die Funktionen sind alle in der Funktionsreferenz Kapitel D enthalten. Das Zusammenspiel der Funktionen wird im Abschnitt 4.15 erläutert.

9.4.13. XLSX Import/Export von Davide Carpi

Das Plugin stellt Funktionen zum Import und Export von xlsx (MS Excel) Dateien bereit.

Die Optionen zum Datenaustausch werden im Zusammenhang mit Matrizen in Abschnitt 4.9.7 erläutert. Eine Übersicht über alle Import- und Exportfunktionen findet sich in Abschnitt 3.18.

Funktion	Seite	Verwendung
`ExportData.XLSX()`	278	Export von Ausdrücken als xlsx (MS Excel)
`ImportData.XLSX()`	299	Import von xlsx-Daten (MS Excel)

Dieses Plugin funktioniert nicht unter Linux.

9.4.14. X-Y Plot Region von Viacheslav N. Mezentsev

Download im SMath-Forum. Dieses Plugin stellt eine Alternative zum eingebauten 2D-Diagrammbereich zur Verfügung. Es befindet sich noch in der Entwicklung. Näheres findet sich in Abschnitt F.3.

10. Anwendungen

10.1. SMath in Berichten und Abschlussarbeiten

SMath dient in solchen Fällen der Dokumentation von verwendeten Daten, Berechnungswegen und Ergebnissen. Die Einbindung eines kleinen SMath-Rechenblatts ist in jedem Fall einer mit dem Formeleditor nachgebauten Handrechnung vorzuziehen, da man in der Regel davon ausgehen kann, dass das was SMath darstellt, auch das ist, was gerechnet wird. Diese Sicherheit hat man beim Formeleditor nicht.

Im Standardumfang von SMath ist nur der Export als Grafik vorgesehen. Gegenüber einem Screenshot (z.B. mit dem Windows 7 Snipping Tool) ist die Größe nicht begrenzt, denn auch mehrseitige Dokumente werden in eine einzige Grafikdatei exportiert.

Das Plugin *Data Exchange* (Seite 215) erweitert die Exportmöglichkeiten auf das OpenDocument Textformat (odt) und auf XeLaTeX (tex). Das odt-Format kann mit Word, OpenOffice oder LibreOffice weiterverarbeitet werden.

Mit dem Plugin kann man ebenfalls einzelne Formeln (Ausdrücke) als OpenOffice-Formeln (odf) exportieren.

Beim Export geht aber die zweidimensionale Struktur des Rechenblatts verloren, alle Bereiche werden untereinander angeordnet. Ein Beispiel finden Sie in Abschnitt 3.18.3.

10.2. Regeln für SMath-Dokumente

Bei kleineren Übungsaufgaben zum Beispiel aus der Technischen Mechanik muss man die SMath Rechnung nicht in ein anderes Dokument einbinden, sondern kann alle erforderliche Information gleich im Rechenblatt mitliefern. Dabei gelten folgende Regeln:

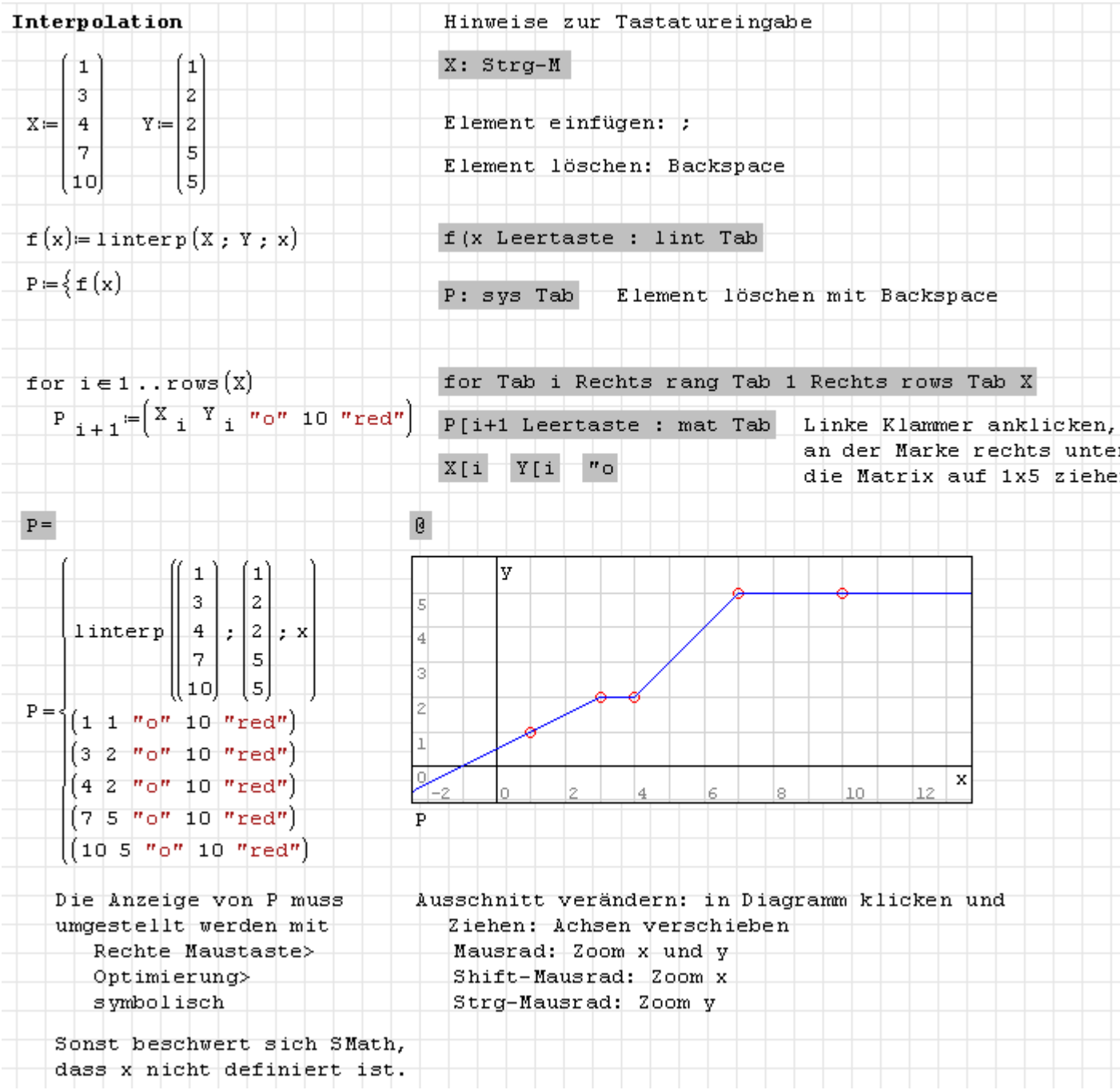
- Die Aufgabenstellung muss im Rechenblatt enthalten sein (z.B. Scan oder Screenshot).
- Verwenden Sie für alle vorkommenden Größen geeignete Variablennamen und weisen Sie diesen die gegebenen Werte zu. Verwenden Sie Namen, die so weit wie möglich an der natürlichen Schreibweise sind, also z.B. F_A statt FA
- Verwenden Sie in Formeln stets die Variablennamen, nicht die Zahlenwerte.
- Benutzen Sie grundsätzlich die SMath-Maßeinheiten. Geben Sie Ergebnisse mit sinnvollen Maßeinheiten an, so dass die Zahlenwerte möglichst zwischen 1 und 1000 liegen und drei signifikante Stellen angegeben werden.
- Benennen Sie (mit Textbereichen) die definierten Größen und die verwendeten Formeln sowie die berechneten Ergebnisse, soweit dies nicht bereits aus der Aufgabenstellung hervorgeht.

- Seien Sie mit Formatierungen genauso sparsam wie bei Bleistiftrechnungen auf dem Papier. Zum Hervorheben von Textbereichen, z.B. für Zwischenüberschriften, müssen Fettschrift und Unterstreichung reichen. Ergebnisse können Sie einrahmen. Manchmal ist es sinnvoll, gegebene Daten hervorzuheben, z.B. hellgelb zu hinterlegen. Das ist besonders dann sinnvoll, wenn Sie ein Rechenblatt erstellen, das andere dann mit ihren eigenen Daten benutzen sollen.
- Bei Mechanikaufgaben muss jede Kraft, die Sie in einer Formel verwenden, auch in einer Skizze auftauchen. Skizzen können Sie von Hand erstellen und als Bild einfügen (fotografieren oder scannen). Alternativ benutzen Sie Zeichenprogramme und fügen Screenshots oder exportierte Bilder ein.
- Jedes Koordinatensystem, das Sie in der Rechnung verwenden oder auf das Sie sich beziehen, muss in einer Skizze gekennzeichnet sein.
- Wenn Sie externe Daten verwenden (Bilder, Tabellen), dann geben Sie deren Namen im Rechenblatt als Kommentar mit an, damit Sie diese später wiederfinden, falls Sie etwas ändern wollen.
- Wenn Sie Berechnungen manuell anstellen müssen (z.B. symbolische Umformungen, die Sie mit SMath nicht hinbekommen), dann benutzen Sie Formelbereiche einfach als Formeleditor ohne Rechenfunktion. Benutzen Sie dabei anstelle des einfachen Gleichheitszeichens (Auswertungsoperator) das logische (boolsche) Gleichheitszeichen.

10.3. Dateninterpolation

Interpolation ist die Berechnung von Funktionswerten in den Zwischenräumen zwischen gegebenen Datenpunkten (x, y-Paaren). Wenn man Punkte sucht, die außerhalb des Bereichs der gegebenen Werte liegen, dann spricht man von Extrapolation, die aber häufig von Interpolationsverfahren gleich mit erledigt wird.

Im folgenden Rechenblatt können Sie die Eigenschaften der Interpolation nachvollziehen. Die Daten sind als Vektoren für die x-Werte und y-Werte gegeben. Die Werte werden mit der Funktion `linterp()` linear interpoliert und die Funktion gemeinsam mit den Datenpunkten dargestellt.



Hinweis: SMath stellt neben der linearen Interpolation zwei weitere Funktionen zur Verfügung:

- `cinterp()` kubische Spline-Interpolation
- `ainterp()` kubische Spline-Interpolation nach Akima Erläuterung des Verfahrens (TU Wien). Das Verfahren berechnet vor der Interpolation an den Stützstellen die Neigung aus einer 5-Punkte-Umgebung. In unserem Beispiel funktioniert das nur für den mittleren Punkt.

10.4. Multilineare Regression

Dieses Beispiel ist der SMath-Einführung von Bernard V. Liengme (Kanada) entnommen. Die Aufgabe besteht darin, die Koeffizienten einer Funktion $f(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_n f_n(x)$ so zu bestimmen, dass ein Datensatz aus zugeordneten x- und y-Werten möglichst gut dargestellt wird.

Eine Möglichkeit dafür ist die Methode der kleinsten Fehlerquadrate, die den mittleren quadratischen (vertikalen) Abstand zwischen den Funktionswerten $f(x)$ und den Datenwerten $y(x)$ minimiert. Die Funktion kann als Skalarprodukt des Koeffizientenvektors $\underline{c}$ und dem Funktionsvektor $\underline{f}$ dargestellt werden:

$$f(x) = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} = \underline{c}^\top \underline{f}$$

Man baut nun eine Matrix $\underline{\underline{X}}$ auf, in der die Spalten gebildet werden durch Anwendung der Funktionen f_i auf den ganzen Datenvektor $\underline{x}$. Die Koeffizienten berechnen sich dann wie folgt:

$$\underline{c} = \left(\underline{\underline{X}}^\top \underline{\underline{X}}\right)^{-1} \underline{\underline{X}}^\top \underline{y}$$

Diese Berechnung soll nun in SMath ausgeführt werden. Zunächst die Definition der Daten:

$$x := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \qquad y := \begin{pmatrix} 0{,}9 \\ 2{,}4 \\ 7{,}0 \\ 17{,}4 \\ 30{,}5 \\ 48{,}3 \end{pmatrix}$$

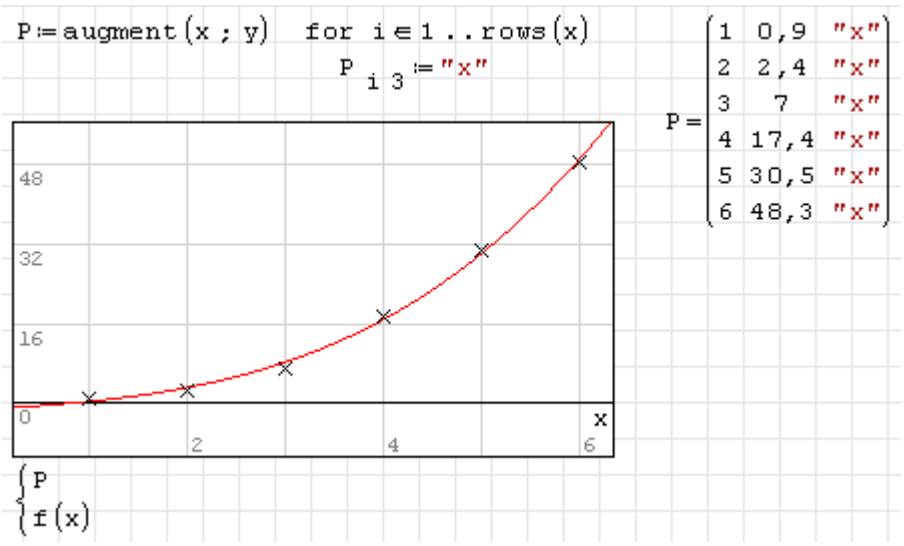
Dann geben wir den Vektor $\underline{f}$ vor und berechnen die Matrix $\underline{\underline{X}}$:

$$f(x) := \begin{pmatrix} 1 \\ x \\ x^3 \end{pmatrix} \qquad \begin{array}{l} \text{for } i \in 1..\text{rows}(x) \\ \quad \text{for } j \in 1..\text{rows}(f(x)) \\ \quad\quad X_{i,j} := f(x_i)_j \end{array} \qquad X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 8 \\ 1 & 3 & 27 \\ 1 & 4 & 64 \\ 1 & 5 & 125 \\ 1 & 6 & 216 \end{pmatrix}$$

Nun berechnen wir die Koeffizienten und bilden daraus die Funktion $f(x)$. Die Funktion `eval()` sorgt dafür, dass die Koeffizienten bei späteren Funktionsauswertungen nicht ständig neu berechnet werden. Beachten Sie, dass das Matrixprodukt $\underline{c}^\top \underline{f}$ in SMath eine 1×1 Matrix ist. Um daraus eine plotbare skalare Funktion zu machen, muss das erste Element durch den Adressindex 1 extrahiert werden.

$$c := \text{eval}\left(\left(X^T \cdot X\right)^{-1} \cdot X^T \cdot y\right) \qquad c = \begin{pmatrix} -1{,}3552 \\ 1{,}4015 \\ 0{,}1932 \end{pmatrix} \qquad f(x) := \left(c^T \cdot f(x)\right)_1$$

Nun soll noch die so bestimmte Funktion mit den Datenpunkten verglichen werden.



10.5. Pendelschwingungen bei großen Auslenkungen

Das folgende Beispiel zeigt, wie Differenzialgleichungen mit dem Runge-Kutta-Verfahren integriert werden können. Wir betrachten ein mathematisches Pendel (Massenpunkt an starrer, masseloser Stange) mit gegebener Länge r. Gesucht ist die Bewegungsfunktion $\varphi(t)$ für die Anfangsgeschwindigkeit v_0 und den Anfangswinkel φ_0.

Radius	$r := 1\ m$
Anfangswinkel	$\varphi_0 := 90\ °$
Angangsgeschwindigkeit	$v_0 := 2\ \frac{m}{s}$
Erdbeschleunigung	$g = 9{,}8066\ \frac{m}{s^2}$

Die Bewegungsgleichung kann man beispielsweise aus der Drehimpulsbilanz um den Aufhängungspunkt gewinnen (SMath dient hier nur der Dokumentation, die Gleichungen sind nicht aktiv)

$$m \cdot r^2 \cdot \frac{d^2}{dt^2}\varphi = -r\, g \cdot m \cdot \sin(\varphi)$$

$$\frac{d^2}{dt^2}\varphi = -\frac{1}{r}\, g \cdot \sin(\varphi)$$

Das ist eine gewöhnliche Differenzialgleichung zweiter Ordnung mit der unabhängigen Variable t und der abhängigen Variable φ.

Als Lösungsverfahren steht uns das Runge-Kutta-Verfahren zur Verfügung, welches Systeme von Differenzialgleichungen erster Ordnung integriert. Wir können z.B. die Funktion `rkfixed()` aus dem Plugin *ODE Solvers* (ODE = Ordinary Differential Equation, gewöhnliche Differenzialgleichung) benutzen.

Dafür müssen wir die Differenzialgleichung zweiter Ordnung in ein System von Differenzialgleichungen erster Ordnung umwandeln. Das geht immer, indem man für die niedrigeren

Ableitungen neue Variable einführt. Im vorliegenden Fall wird die Winkelgeschwindigkeit ω als erste Ableitung des Winkels eingeführt. Die zweite Ableitung des Winkels nach der Zeit ist dann gleich der ersten Ableitung der Winkelgeschwindigkeit nach der Zeit.

$$\omega = \frac{\mathrm{d}}{\mathrm{d}t}\varphi, \quad \frac{\mathrm{d}^2}{\mathrm{d}t^2}\varphi = \frac{\mathrm{d}}{\mathrm{d}t}\omega = -\frac{g}{r}\sin(\varphi)$$

Die abhängigen Variablen und ihre ersten Ableitungen werden zu einem Vektor zusammengefasst:

$$\underline{y} = \begin{bmatrix} \varphi \\ \omega \end{bmatrix}, \quad \underline{y}' = \begin{bmatrix} y_2 \\ -\frac{g}{r}\sin(y_1) \end{bmatrix}.$$

Der Ableitungsvektor muss als Funktion der unabhängigen und der abhängigen Variable bereitgestellt werden.

$$D(t\ ;\ y) := \begin{pmatrix} y_2 \\ -\frac{g}{r}\cdot\sin\left(y_1\right) \end{pmatrix}$$

Außerdem wird der Vektor der Anfangsbedingungen benötigt:

$$y_0 := \begin{pmatrix} 90\ ° \\ \frac{v_0}{r} \end{pmatrix}$$

Die Funktion `rkfixed()` integriert die abhängigen Variablen über einen Zeitraum von t_0 bis t_e unterteilt in n gleichgroße Schritte.

$$t_0 := 0\ s \qquad t_e := 2{,}6\ s \qquad n := 100$$

$$L := \mathrm{rkfixed}\left(y_0\ ;\ t_0\ ;\ t_e\ ;\ n\ ;\ D(t\ ;\ y)\right)$$

Zurückgeliefert wird eine Matrix mit den Werten der unabhängigen Variable in der ersten Spalte und den abhängigen Variablenwerten in den weiteren Spalten. Jede Zeile entspricht einem Zeitpunkt. Maßeinheiten sind im Ergebnis nicht mehr enthalten, dies ist ein Mangel des verwendeten Plugins. Die Endzeit t_e wird so bestimmt (durch Ausprobieren), dass sich in der Animations-Endlosschleife ein flüssiger Bewegungsablauf ergibt.

$$t_0 := 0\ s \qquad t_e := 2{,}45\ s \qquad n := 100$$

$$L := \mathrm{rkfixed}\left(y_0\ ;\ t_0\ ;\ t_e\ ;\ n\ ;\ D(t\ ;\ y)\right)$$

$$\mathrm{row}(L\ ;\ 2) = (0{,}0245\ \ 1{,}6169\ \ 1{,}7598)$$

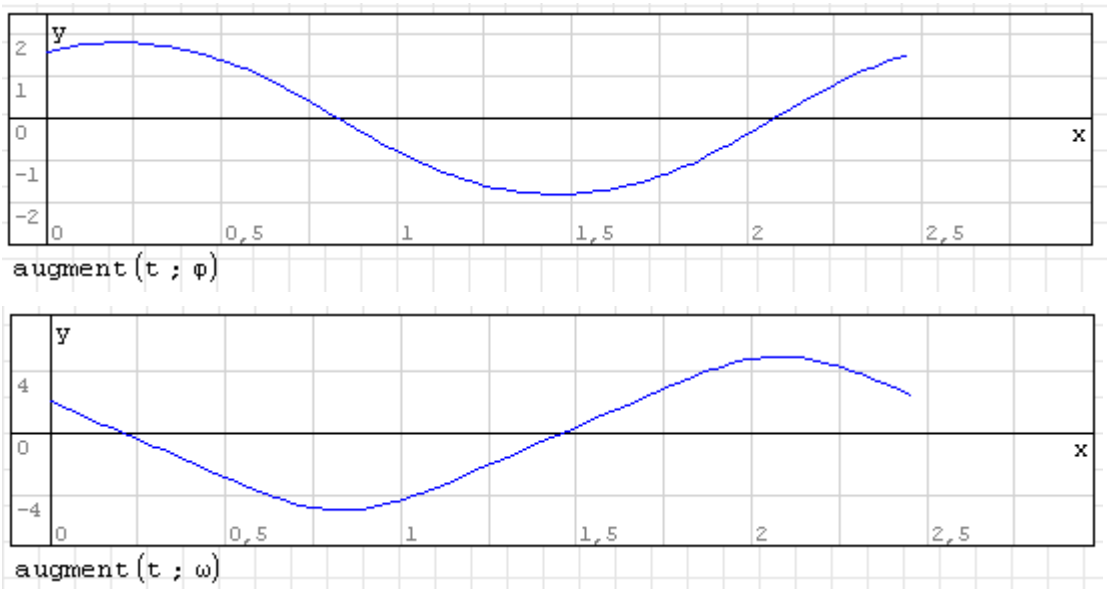
Die Ergebnisgrößen werden als Vektoren extrahiert und können dann grafisch dargestellt werden.

$$t := \mathrm{col}(L\ ;\ 1) \qquad \text{unabhängige Variable}$$

$$\varphi := \mathrm{col}(L\ ;\ 2) \qquad \text{abhängige Variable 1}$$

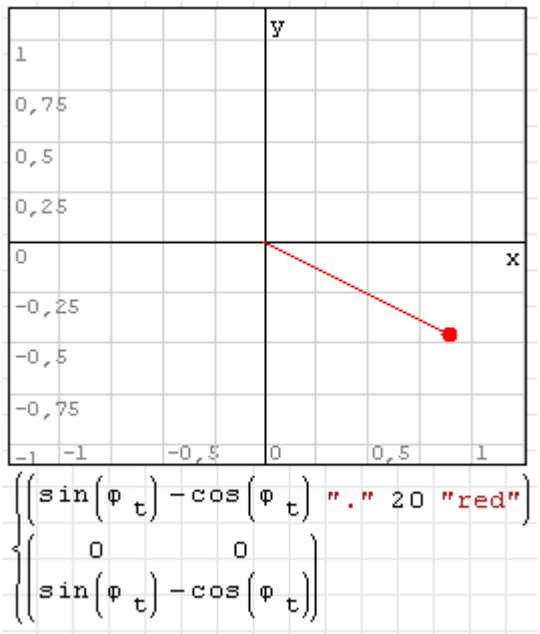
$$\omega := \mathrm{col}(L\ ;\ 3) \qquad \text{abhängige Variable 2}$$

$$k := 1\ ..\ \mathrm{rows}(L) \qquad \text{Indexvektor für die Animation}$$

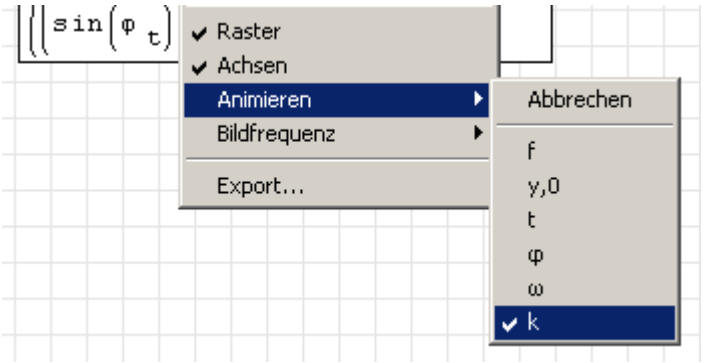


Besonders interessant ist die Möglichkeit, die Bewegung des Pendels animiert darzustellen. Die Koordinaten des Massenpunkts ergeben sich aus dem Winkel $\varphi(t)$ und dem Radius r:

$$x = \sin\varphi, \quad y = -\cos\varphi$$



Die Animation wird erzeugt, indem im Kontextmenü (rechte Maustaste) der oben erzeugte Indexvektor k als Wertebereich für t ausgewählt wird. t entspricht hier nicht der Zeit, sondern einem Zähler, der die berechneten Datensätze durchläuft.



10.6. Lorentz-Attraktor

Dieses Beispiel zeigt:

- die Integration von Systemen gewöhnlicher Differenzialgleichungen
- die Verwendung des X-Y Plot-Bereichs

Attraktoren sind Gebiete im Phasenraum dynamischer Systeme, die auf die Umgebung anziehend wirken. Der Lorentz-Attraktor ist der Attraktor des folgenden Differenzialgleichungssystems:

$$\dot{x} = a(y - x)$$
$$\dot{y} = x(b - z) - y$$
$$\dot{z} = xy - cz$$

Dieses System kann mit dem Runge-Kutta-Verfahren integriert werden, geeignet ist dafür z.B. die Funktion `Rkadapt()` aus dem Plugin *ODE Solvers*.

Im Beispiel ist t die unabhängige Variable, die von 0 bis 50 Sekunden läuft. Auszugeben sind 2000 Zeitschritte. Die abhängigen Variablen x, y, z sind Komponenten des Vektors $\mathbf{x}$, dessen Anfangswert ist $\mathbf{x}_0$.

$$D(t\ ;\ x) := \begin{pmatrix} a\cdot(x_2 - x_1) \\ x_1\cdot(b - x_3) - x_2 \\ x_1\cdot x_2 - c\cdot x_3 \end{pmatrix} \qquad x_0 := \begin{pmatrix} 0,1 \\ 0,1 \\ 0,5 \end{pmatrix} \qquad t_0 := 0 \qquad t_{max} := 50 \qquad n := 2000$$

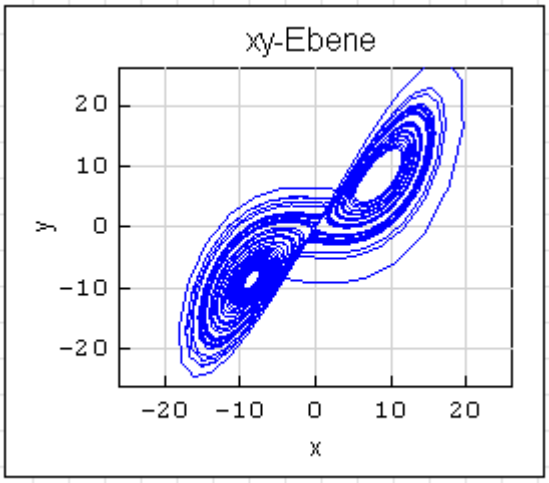
Im folgenden Abschnitt werden die Systemparameter angegeben, das Integrationsverfahren aufgerufen und die Spalten der Ergebnismatrix extrahiert.

$$a := 10 \qquad b := 28 \qquad c := \frac{8}{3}$$

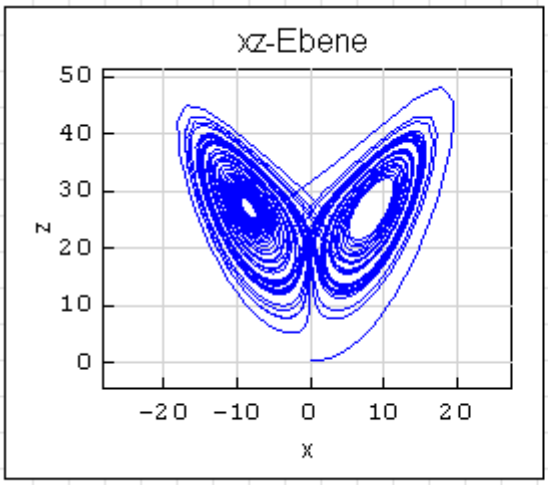
$$L := \mathrm{Rkadapt}\left(x_0\ ;\ t_0\ ;\ t_{max}\ ;\ n\ ;\ D(t\ ;\ x)\right)$$

$$x := \mathrm{col}(L\ ;\ 2) \qquad y := \mathrm{col}(L\ ;\ 3) \qquad z := \mathrm{col}(L\ ;\ 4)$$

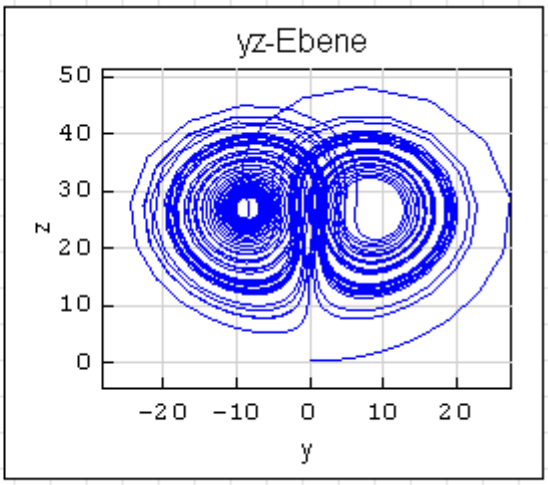
Die xy-, xz- und yz- Phasenebenen werden mit dem X-Y Plot-Bereich aus den Plugin *X-Y Plot Region* dargestellt.



augment(x ; y)



augment(x ; z)



augment(y ; z)

A. Operatoren

Hier finden Sie eine vollständige Liste aller Operatoren in SMath Studio. Die meisten dieser Operatoren sind auch für Dritte, die SMath nicht kennen, verständlich.

Allerdings ist es häufig besonders für Anfänger schwer erkennbar, wie die Operatoren einzugeben sind. Solange die Paletten der Seitenleiste hier nicht alles abdecken oder die Funktionsnamen statt der Operatoren verwenden, ist diese Liste eine Hilfe.

Darstellung	Eingabe	Bedeutung
m	'm	Maßeinheit
a_b	a.b a,b	Bezeichner mit Textindex
:=	:	Zuweisung, Definition
=	=	Numerische Auswertung
=	Strg - .	Symbolische Auswertung
▪|▪	at ⇆	Substitution (Plugin *Functions Extensions*)
▪|▪	at ⇆ ;	Substitution (Plugin *Functions Extensions*)
{▪	sys ⇆	Liste
(▪ ▪)	mat ⇆	Matrix
M(▪)	maxi ⇆	Auswertung mit Maxima (Plugin *Functions Extensions* und *Maxima*)
Arithmetik		
+▪, ▪+▪	+	Positives Vorzeichen (reine Optik) oder Addition
−▪, ▪−▪	-	Negatives Vorzeichen oder Subtraktion
▪±▪	%	Plus/Minus
∓▪	Insert> Operator	Minus/Plus
▪·▪	*	Multiplikation
▪!	!	Fakultät
▪/▪	/	Division
▪^▪	^	Potenzierung
log▪(▪)	log ⇆	Logarithmus mit angebbarer Basis

$\lvert\blacksquare\rvert$	abs ⇆	Betrag einer Zahl
$\sqrt[\blacksquare]{\blacksquare}$	Strg-^	n-te Wurzel
$\sqrt{\blacksquare}$	\	Quadratwurzel
$\lceil\blacksquare\rceil$	ce ⇆	nächstgrößere Ganzzahl (Plugin *Functions Extensions*)
$\lfloor\blacksquare\rfloor$	fl ⇆	nächstkleinere Ganzzahl (Plugin *Functions Extensions*)

Matrizen und Vektoren

$\begin{Bmatrix}\blacksquare\\\blacksquare\end{Bmatrix}$	sys ⇆	Liste
$\begin{pmatrix}\blacksquare&\blacksquare\\\blacksquare&\blacksquare\end{pmatrix}$	mat ⇆	Matrix
$\blacksquare\times\blacksquare$	Strg-8	Kreuzprodukt (Vektoren)
$\lvert\blacksquare\rvert$	det ⇆	Determinante einer Matrix
$\blacksquare^{T}$	tran ⇆	Transposition einer Matrix
v_j	[	Elementadressierung in Feldern
$\blacksquare_{\blacksquare\,\blacksquare}$	[→ ;	mit zwei Indizes
$\sum\blacksquare$	sum ⇆	Summe aller Elemente
$\sum_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare$	sum ↓ ⇆	Summe
$\prod_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare$	pr ⇆	Produkt
$\blacksquare..\blacksquare$	rang ⇆	Wertebereich (Vektor)
$\blacksquare;\blacksquare..\blacksquare$	rang ⇆ ;	Wertebereich, 2. Wert angebbar

Logische und Vergleichsoperatoren (Boolesch)

$\blacksquare=\blacksquare$	~	Logisches Gleich
$\blacksquare<\blacksquare$	<	Kleiner als
$\blacksquare>\blacksquare$	>	Größer als
$\blacksquare\le\blacksquare$	Strg-9	Kleiner oder gleich
$\blacksquare\ge\blacksquare$	Strg-0	Größer oder gleich
$\blacksquare\neq\blacksquare$	Strg-3	Ungleich
$\neg\blacksquare$	Panel Boolesch	Logische Negation
$\blacksquare\wedge\blacksquare$	&	Logisches Und
$\blacksquare\vee\blacksquare$	|	Logisches Oder
$\blacksquare\oplus\blacksquare$	Panel Boolesch	Exklusives Oder (XOR)
$\blacksquare\le\blacksquare\le\blacksquare$	lele ⇆	Intervall (Plugin *Functions Extensions*)

▮≤▮<▮	lelt ⇆	
▮<▮≤▮	ltle ⇆	
▮<▮<▮	ltle ⇆	
▮≥▮≥▮	gege ⇆	
▮≥▮>▮	gegt ⇆	
▮>▮≥▮	gtge ⇆	
▮>▮>▮	gtgt ⇆	

Integral- und Differenzialrechnung (Analysis)

$\blacksquare'$	dif ⇆	Differentiation
$\frac{d}{d\blacksquare}\blacksquare$	dif ⇆ ;	Differentiation
$\frac{d\blacksquare}{d\blacksquare}\blacksquare$	dif ⇆ ;;	Differentiation
$\int \blacksquare\, d\blacksquare$	int ⇆	unbestimmtes Integral
$\int_{\blacksquare}^{\blacksquare} \blacksquare\, d\blacksquare$	int ⇆ ;;	bestimmtes Integral
$\lim_{\blacksquare\to\blacksquare}\blacksquare$	lim ⇆	Grenzwert (Plugin *Maxima*)

Programmierung

▮ if ▮ ▮ otherwise	cas ⇆ ;;	Fallunterscheidung Intervall (Plugin *Functions Extensions*)
for ▮∈▮ ▮	for ⇆	for-Schleife mit Wertebereich
for ▮;▮;▮ ▮	for ⇆ ;	for-Schleife mit Initialisierung, Fortsetzungsbedingung und Inkrementierung
while ▮ ▮	wh ⇆	while-Schleife
if ▮ ▮ else ▮	if ⇆	Bedingte Anweisung
▮ ▮	lin ⇆	Anweisungsblock

B. Tastenbelegung

Taste	Anweisung	Funktion
"		Rechenblatt: Textblock Formelbereich: Zeichenkette
’ (Hochkomma)		Platzhalter für vordefinierte Einheit, Funktion oder Konstante
\	sqrt()	 Quadratwurzel
[	el()	 Platzhalter für Elementindex (Vektor, Matrix) erzeugen
]	line()	 Anweisungsblock
.		Textindex an Bezeichnern
^		 Platzhalter für Exponent
%		
!		Fakultät
&		 Logisches Und
|		 Logisches Oder
~		 Logisches Gleich
@		2D-Diagramm
Strg - .		Symbolische Auswertung
Strg - ^	nthroot()	 n-te Wurzel
Strg - ↵		Blatt: Trennlinie einfügen Textblock: Neue Zeile
Strg - Leert.		Dynamische Auswahlliste aktivieren
⇆		Gehe zum nächsten Bereich
⇑ - ⇆		Gehe zum vorigen Bereich
Strg - ⇆		Umschalten zum nächsten offenen Rechenblatt
Strg - +		 Logisches Gleich
Strg - 1	transpose()	 Transposition (Matrix)
Strg - 3		 Ungleich
Strg - 8		 Kreuzprodukt (von Vektoren)
Strg - 9		 Kleiner oder gleich
Strg - 0		 Größer oder gleich
Strg - A		Alles markieren
Strg - B		Textblock Fettschrift an/aus

Taste	Anweisung	Funktion
Strg - C		Markierung in die Zwischenablage kopieren
Strg - E		Dialog „Funktion einfügen”
Strg - F		Snapshot-Bereich einfügen
Strg - G		vorherigen Buchstaben griechisch machen
Strg - H		Hyperlink(-bereich) einfügen
Strg - I		Textblock Kursivschrift an/aus
Strg - M	`mat()`	Matrix
Strg - N		Neues Rechenblatt erzeugen
Strg - O		Dialog „Blatt Öffnen”
Strg - P		Dialog „Drucken”
Strg - S		Blatt speichern
Strg - T		Zeichenfläche einfügen
Strg - U		Textblock Unterstreichung an/aus
Strg - V		Zwischenablage einfügen
Strg - W		Dialog „Einheit einfügen”
Strg - X		Markierung ausschneiden und in Zwischenablage kopieren
Strg - Y		Wiederholen
Strg - X		Rückgängig
Strg - ⇑ - P		π Pi
Strg - ⇑ - Z		∞ Unendlich
Pos1		Anfang des Ausdrucks oder Textblocks
End		Ende des Ausdrucks oder Textblocks
Strg - F4		Rechenblatt schließen
F8		gleichzeitiges Editieren eines Namens an allen Stellen eines Formelbereichs
F9		Rechenblatt neu berechnen
F12		Gehe zur Definition (des Bezeichners, in dem sich die Einfügemarke befindet)
In der dynamischen Auswahl		
⇆		Auswahl bestätigen
Strg - Leert.		Auswahl bestätigen
Strg - ↵		Auswahl bestätigen
⇑ - ↵		Auswahl bestätigen

C. Vordefinierte Maßeinheiten und Konstanten

Konstanten und Maßeinheiten werden in der Datei *Installationsverzeichnis/Entries/Units.xml* definiert. Man könnte dort eigene Einheiten und Sprachversionen definieren. Das ist allerdings nicht zu empfehlen, da diese Definitionen in anderen SMath-Installationen nicht bekannt sind. Für die Portabilität ist es besser, solche Definitionen im SMath-Rechenblatt selbst vorzunehmen.

Winkel

Der Anschaulichkeit halber werden die Ergebnisse hier in ° (Grad) angezeigt, standardmäßig würde die Anzeige im Bogenmaß erfolgen.

$1\ rad = 57{,}2958\ °$	Standardeinheit: Radiant
$1\ ° = 1\ °$	Grad
$1\ gon = 0{,}9\ °$	Neugrad, Gon
$1\ rev = 360\ °$	Umdrehung

Länge

$1\ m = 1\ m$	Standardeinheit: Meter
$1\ mi = 1609{,}344\ m$	Meile
$1\ km = 1000\ m$	Kilometer
$1\ dm = 0{,}1\ m$	Dezimeter
$1\ cm = 0{,}01\ m$	Zentimeter
$1\ mm = 0{,}001\ m$	Millimeter
$1\ \mu m = 1 \cdot 10^{-6}\ m$	Mikrometer
$1\ nm = 1 \cdot 10^{-9}\ m$	Nanometer
$1\ pm = 1 \cdot 10^{-12}\ m$	Picometer

$1\ Angstrom = 1 \cdot 10^{-10}\ m$	Angström
$1\ bohr = 5{,}2918 \cdot 10^{-11}\ m$	Bohrscher Radius
$1\ ft = 0{,}3048\ m$	Fuß (Zollsystem)
$1\ furlong = 201{,}168\ m$	Furlong ("Furche")
$1\ in = 0{,}0254\ m$	Zoll
$1\ yd = 0{,}9144\ m$	Yard
$1\ Smoot = 1{,}7018\ m$	Smoot (Körpergröße von O. Smoot)

Fläche

$1\ acre = 4046{,}8564\ m^2$	Morgen
$1\ barn = 1 \cdot 10^{-28}\ m^2$	Barn (Atomphysik)
$1\ hectare = 10000\ m^2$	Hektar

Volumen

$1\ L = 0{,}001\ m^3$	Liter
$1\ mL = 1 \cdot 10^{-6}\ m^3$	Milliliter
$1\ gal = 0{,}0038\ m^3$	Gallone

Zeit

$1\ s = 1\ s$	Sekunde
$1\ yr = 3{,}1557 \cdot 10^{7}\ s$	Jahr
$1\ day = 86400\ s$	Tag
$1\ hr = 3600\ s$	Stunde
$1\ min = 60\ s$	Minute
$1\ ks = 1000\ s$	Kilosekunde
$1\ ms = 0{,}001\ s$	Millisekunde
$1\ \mu s = 1 \cdot 10^{-6}\ s$	Mikrosekunde
$1\ ns = 1 \cdot 10^{-9}\ s$	Nanosekunde
$1\ ps = 1 \cdot 10^{-12}\ s$	Picosekunde

Winkelgeschwindigkeit, Kreisfrequenz

Die Standardeinheit für Drehzahl, Winkelgeschwindigkeit und Kreisfrequenz ist Radiant pro Sekunde. Eine Umdrehung entspricht 2π Radiant.

$1\ rpm = 0,1047 \cdot \frac{1}{s}$	Umdrehung pro Minute
$1\ rph = 0,0017 \cdot \frac{1}{s}$	Umdrehung pro Stunde
$1\ radpm = 0,0167 \cdot \frac{1}{s}$	Radiant pro Minute

Frequenz

$1\ Hz = 1 \cdot \frac{1}{s}$	Hertz
$1\ GHz = 1 \cdot 10^{9} \cdot \frac{1}{s}$	Gigahertz
$1\ MHz = 1 \cdot 10^{6} \cdot \frac{1}{s}$	Megahertz
$1\ kHz = 1000 \cdot \frac{1}{s}$	Kilohertz

Masse

$1\ kg = 1\ kg$	Standardeinheit: Kilogramm
$1\ Mg = 1000\ kg$	Megagramm (Tonne)
$1\ t = 1000\ kg$	Tonne
$1\ ton = 907,1847\ kg$	Tonne (Zollsystem)
$1\ gram = 0,001\ kg$	Gramm (Symbol g durch Erdbeschleunigung belegt)
$1\ mg = 1 \cdot 10^{-6}\ kg$	Milligramm
$1\ \mu g = 1 \cdot 10^{-9}\ kg$	Mikrogramm
$1\ oz = 0,0283\ kg$	Unze (Zollsystem)
$1\ lb = 0,4536\ kg$	Pfund (Zollsystem)
$1\ slug = 14,5939\ kg$	Slug (kohärente Masseneinheit im Zollsystem)
	$1\ lbf = 1\ \frac{slug\ ft}{s^{2}}$

Kraft

$1\ N = 1\ N$	Standardeinheit: Newton
$1\ TN = 1 \cdot 10^{12}\ N$	Teranewton
$1\ GN = 1 \cdot 10^{9}\ N$	Giganewton
$1\ MN = 1 \cdot 10^{6}\ N$	Meganewton
$1\ kN = 1000\ N$	Kilonewton
$1\ mN = 0,001\ N$	Millinewton
$1\ \mu N = 1 \cdot 10^{-6}\ N$	Mikronewton

$1\ dyn = 1 \cdot 10^{-5}\ N$	Dyn
$1\ lbf = 4,4482\ N$	Pfund Kraft (Zollsystem)
$1\ kgf = 9,8066\ N$	Kilogramm Kraft (Kilopond)
$1\ tonnef = 9806,65\ N$	Tonne Kraft (Megapond)
$1\ tonf = 8896,4432\ N$	Tonne Kraft (Zollsystem)
$1\ kip = 4448,2216\ N$	Tausend Pfund Kraft (Zollsystem)

Druck

$1\ Pa = 1\ Pa$	Standardeinheit: Pascal
$1\ GPa = 1 \cdot 10^{9}\ Pa$	Gigapascal
$1\ MPa = 1 \cdot 10^{6}\ Pa$	Megapascal
$1\ kPa = 1000\ Pa$	Kilopascal
$1\ atm = 1,0133 \cdot 10^{5}\ Pa$	Atmosphäre
$1\ torr = 133,3224\ Pa$	Torr (mm Quecksilbersäule)
$1\ psi = 6894,7573\ Pa$	Pfund Kraft pro Quadratzoll (Zollsystem)
$1\ psf = 47,8803\ Pa$	Pfund Kraft pro Quadratfuß (Zollsystem)
$1\ ksi = 6,8948 \cdot 10^{6}\ Pa$	Tausend Pfund Kraft pro Quadratzoll
$1\ ksf = 47880,259\ Pa$	Tausend Pfund Kraft pro Quadratfuß

Energie

$1\ J = 1\ J$	Standardeinheit: Joule
$1\ TJ = 1 \cdot 10^{12}\ J$	Terajoule
$1\ GJ = 1 \cdot 10^{9}\ J$	Gigajoule
$1\ MJ = 1 \cdot 10^{6}\ J$	Megajoule
$1\ kJ = 1000\ J$	Kilojoule
$1\ mJ = 0,001\ J$	Millijoule
$1\ cal = 4,1868\ J$	Kalorie
$1\ kcal = 4186,8\ J$	Kilokalorie
$1\ BTU = 1055,0559\ J$	British Thermal Unit
$1\ erg = 1 \cdot 10^{-7}\ J$	Erg

Leistung

$1\,W = 1\,W$	Standardeinheit: Watt
$1\,GW = 1 \cdot 10^{9}\,W$	Gigawatt
$1\,MW = 1 \cdot 10^{6}\,W$	Megawatt
$1\,kW = 1000\,W$	Kilowatt
$1\,mW = 0{,}001\,W$	Milliwatt
$1\,\mu W = 1 \cdot 10^{-6}\,W$	Mikrowatt
$1\,nW = 1 \cdot 10^{-9}\,W$	Nanowatt
$1\,pW = 1 \cdot 10^{-12}\,W$	Picowatt
$1\,hp = 745{,}6999\,W$	Britische Pferdestärke (Zollsystem)
$1\,hhp = 746{,}043\,W$	Britische Pferdestärke (hydraulisch, inkorrekt)
	Metrische Pferdestärke:
	$PS := 75\,kgf \cdot 1\,\frac{m}{s}$ $PS = 735{,}4988\,W$

Stromstärke

$1\,A = 1\,A$	Standardeinheit: Ampere
$1\,kA = 1000\,A$	Kiloampere
$1\,mA = 0{,}001\,A$	Milliampere
$1\,\mu A = 1 \cdot 10^{-6}\,A$	Mikroampere
$1\,nA = 1 \cdot 10^{-9}\,A$	Nanoampere
$1\,pA = 1 \cdot 10^{-12}\,A$	Picoampere

Spannung

$1\,V = 1\,V$	Standardeinheit: Volt
$1\,kV = 1000\,V$	Kilovolt
$1\,mV = 0{,}001\,V$	Millivolt
$1\,\mu V = 1 \cdot 10^{-6}\,V$	Mikrovolt
$1\,nV = 1 \cdot 10^{-9}\,V$	Nanovolt
$1\,pV = 1 \cdot 10^{-12}\,V$	Picovolt

Kapazität

$1\,F = 1\,F$	Standardeinheit: Farad
$1\,mF = 0{,}001\,F$	Millifarad
$1\,\mu F = 1 \cdot 10^{-6}\,F$	Mikrofarad
$1\,nF = 1 \cdot 10^{-9}\,F$	Nanofarad
$1\,pF = 1 \cdot 10^{-12}\,F$	Picofarad

Ladung

$1\,C = 1\,C$	Standardeinheit: Coulomb
$1\,mC = 0{,}001\,C$	Millicoulomb
$1\,\mu C = 1 \cdot 10^{-6}\,C$	Mikrocoulomb
$1\,nC = 1 \cdot 10^{-9}\,C$	Nanocoulomb
$1\,pC = 1 \cdot 10^{-12}\,C$	Picocoulomb

Induktivität

$1\,H = 1\,H$	Standardeinheit: Henry
$1\,mH = 0{,}001\,H$	Millihenry
$1\,\mu H = 1 \cdot 10^{-6}\,H$	Mikrohenry

Magnetische Flussdichte

$1\,T = 1\,T$	Standardeinheit: Tesla
$1\,G = 0{,}0001\,T$	Gauss

Widerstand

$1\,\Omega = 1\,\Omega$	Standardeinheit: Ohm
$1\,M\Omega = 1 \cdot 10^{6}\,\Omega$	Megaohm
$1\,k\Omega = 1000\,\Omega$	Kiloohm

Strahlendosis

$1\,Gy = 1\,Gy$	Standardeinheit: Gray
$1\,Sv = 1\,Gy$	Sievert

Lichtstärke, Lichtstrom

$1\ cd = 1\ cd$	Standardeinheit: Candela
$1\ lm = 1\ cd$	Lumen

Beleuchtungsstärke

$1\ lx = 1\ lx$	Standardeinheit: Lux $\quad 1\ lx = 1\ \frac{cd}{m^2}$

Stoffmenge

$1\ mol = 1\ mol$	Standardeinheit: Mol
$1\ kmol = 1000\ mol$	Kilomol
$1\ mmol = 0,001\ mol$	Millimol
$1\ \mu mol = 1 \cdot 10^{-6}\ mol$	Mikromol

Geschwindigkeit

$1\ kn = 0,5144\ \frac{m}{s}$	Knoten
$1\ kph = 0,2778\ \frac{m}{s}$	Kilometer pro Stunde (kmh)
$1\ mph = 0,447\ \frac{m}{s}$	Meilen pro Stunde

Viskosität

Dynamische Viskosität (Maßeinheit Poise) und kinematische Viskosität (Maßeinheit Stokes):

$1\ P = 0,1\ s\ Pa$	Poise $\quad 1\ P = 0,1\ \frac{kg}{m\ s}$
$1\ St = 1\ \frac{cm^2}{s}$	Stokes

Katalytische Aktivität

$1\ katal = 1\ \frac{mol}{s}$	Katal

Auflösung

$1\ dpi = 39,3701 \cdot \frac{1}{m}$	Punkte pro Zoll

Information

$1\ bit = 1\ bit$	Standardeinheit: Bit
$1\ B = 8\ bit$	Byte
$1\ TB = 8 \cdot 10^{12}\ bit$	Terabyte
$1\ GB = 8 \cdot 10^{9}\ bit$	Gigabyte
$1\ MB = 8 \cdot 10^{6}\ bit$	Megabyte
$1\ kB = 8000\ bit$	Kilobyte

$1\ TiB = 8{,}7961 \cdot 10^{12}\ bit$	Tebibyte
$1\ GiB = 8{,}5899 \cdot 10^{9}\ bit$	Gibibyte
$1\ MiB = 8{,}3886 \cdot 10^{6}\ bit$	Mebibyte
$1\ kiB = 8192\ bit$	Kibibyte

Temperatur

Absoluttemperaturskalen, auch für Temperaturdifferenzen verwendbar:

$1\ K = 1\ K$	Kelvin
$1\ °Ra = 0{,}5556\ K$	Grad Rankine

Temperaturskalen mit verschobenem Nullpunkt:

$0\ °C = 273{,}15\ K$	Grad Celsius
$0\ °F = 255{,}3722\ K$	Grad Fahrenheit
$0\ °Re = 273{,}15\ K$	Grad Reamur

Temperaturdifferenzen von einem Grad auf verschiedenen Skalen:

$1\ °C - 0\ °C = 1\ K$	
$1\ °F - 0\ °F = 0{,}5556\ K$	$1\ °F - 0\ °F = 1\ °Ra$
$1\ °Re - 0\ °Re = 1{,}25\ K$	

Hinweis: Temperaturen in Differenzen dürfen in °C, °F und °Re angegeben werden, die Temperaturdifferenz selbst kann aber nur die Einheit K oder °Ra besitzen.

Konstanten

$c = 2{,}9979 \cdot 10^{8}\ \frac{m}{s}$	Lichtgeschwindigkeit im Vakuum
$G_N = 6{,}6743 \cdot 10^{-11}\ \frac{m^2}{kg^2}\ N$	Gravitationskonstante
$h = 6{,}6261 \cdot 10^{-34}\ s\ J$	Plancksches Wirkungsquantum

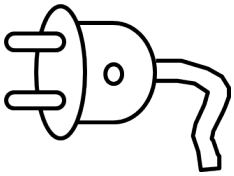
$m_e = 9,1094 \cdot 10^{-31}\ kg$	Elektronenmasse
$m_p = 1,6726 \cdot 10^{-27}\ kg$	Protonenmasse
$m_n = 1,6749 \cdot 10^{-27}\ kg$	Neutronenmasse
$m_u = 1,6605 \cdot 10^{-27}\ kg$	Atomare Masseneinheit

$R = 8,3145\ \frac{m\ N}{K\ mol}$	Gaskonstante
$k = 1,3807 \cdot 10^{-23}\ \frac{m\ N}{K}$	Boltzmann-Konstante
$g = 9,8066\ \frac{m}{s^2}$	Erdbeschleunigung

D. Funktionen

Einige Funktionsnamen (inverse Winkelfunktionen) sind von der Benutzereinstellung im Menü **Extras> Einstellungen> Interface> Funktionsstil** abhängig. Diese Anleitung geht von der Einstellung „Europa" aus.

In den Beschreibungstexten sind solche Funktionen, die ein externes Plugin benötigen, mit einem Stecker (plug) gekennzeichnet. Im Beschreibungstext wird das erforderliche Plugin benannt und verlinkt. Wenn Sie die inoffizielle portable Distribution des Autors benutzen, dann sind die meisten Plugins vorinstalliert.



D.1. Übersicht

Hier sind die Einträge thematisch geordnet. Die Seitenzahlen verweisen auf eventuelle Erläuterungen im alphabetisch sortierten Beschreibungsteil. Kursive Namen verweisen auf Funktionen in externen Plugins (siehe Abschnitt 9.4).

Reelle und komplexe Zahlen

+ (Addition)
- (Subtraktion)
* (Multiplikation)
/ (Division)
! (Fakultät)
^ (Potenzierung)
± (Plusminus)
∞ (unendlich)
π (Kreiszahl)
i (imaginäre Einheit)
= (Anzeige)
:= (Zuweisung)
∑ (Summe)
∏ (Produkt)
abs() 249
arccos()
arccosec()
arcctg()
arch()
arcsec()
arcsin()
arctg()
arcth()
arg()
arsh()
arth()
at() 253
Beta() 254
BetaRegularized() 255
cases() 256
Ceil() 266
ch()
Clear() 38
cos()
cosec()
cot()
cth()
diff() 270
description() 268
Dirac() 270
erf() 273
erfc() 274
erfinv() 274
eval() 275
exp()
Floor() 280
Gamma() 281
GammaRegularized.P() 282
GammaRegularized.Q() 282
GetType() 285
Gradient() 288
Heaviside() 289
Heaviside.D() 290
Im()

int() 300
log()
numden()
polyroots()
Re()
sech()
sinh()
sum() 338
tanh()
ln()
mod()
perc() 323
product()
roots()
sign()
solve() 334
sys() 339
trunc()
lg()
nthroot() 314
pol2xy()
random()
round()
sin()
sqrt() 335
tan()
xy2pol()

Spezialfunktionen

al_airy() 249
BetaRegularized() 255
erfinv() 274
GammaRegularized.Q() 282
al_beta() 249
erf() 273
Gamma() 281
Beta() 254
erfc() 274
GammaRegularized.P() 282

Matrizen und Vektoren

+ (Addition)
/ (Division)
alg() 252
Clear() 38
csort() 268
Diag() 270
GetType() 285
Hessian.CD() 291
invert() 131
Jacobian.CD() 302
mat() 305
max() 306
min() 309
norm1() 313
Ones() 314
range() 196
row()331
sort() 335
sys2mat() 339
sum() 338
Zeros() 344
- (Subtraktion)
× (Kreuzprodukt)
augment() 254
col() 267
det() 269
el() 272
Gradient() 288
identity() 121
Jacob()
length() 303
mat2sys() 305
Max() 306
Min() 309
norme() 313
Random() 323
rank() 197
rows() 331
stack() 335
tr() 339
sys() 339
* (Multiplikation)
ainterp() 249
cinterp()304
cols() 267
diag() 270
findrows() 279
Hessian() 288
InterpBilinear() 301
Jacobian() 302
linterp() 304
matrix() 305
Mean() 309
minor() 310
normi() 314
Random.N() 323
reverse() 329
rsort() 331
submatrix() 338
transpose() 340
vminor() 341

Ebene Polygone

gpc_add_contour() 286
gpc_polygon() 287
gpc_tristrip_clip() 288
gpc_clip() 287
gpc_polygon_to_tristrip() 287
gpc_write_polygon() 288
gpc_get_contour() 287
gpc_read_polygon() 288

Zeichenketten

Clear() 38
findstr() 279
num2str()174
strlen() 228
strtolower() 231
ucfirst() 241
concat() 268
GetType() 285
str2num() 226
strrep() 229
strtoupper() 232
ucwords() 242
description() 268
IsString() 132
strjoin() 227
strsplit() 230
substr() 234

Programmierung

at() 253
Clear()38
for() 281
IsDefined()
time() 339
break 256
continue 268
GetType() 285
line() 304
trace()
cases() 256
error() 275
if() 297
range() 324
while() 343

Dateien

CurrentDirectory() 268
exportCell()275
exportData.ODS() 277
GETWAVINFO() 285
importData.ODS() 298
rfile() 330
READBMP() 327
SettingsDirectory() 332
WRITEBIN() 343
dfile()269
exportData.CSV() 275
exportData.XLSX() 278
importCell() 297
importData.XLSX() 299
READ_IMAGE() 326
READRGB() 328
wfile() 342
WRITEWAV() 343
DocumentDirectory() 271
exportData.ODF() 276
GetFolderPath() 284
importData() 298
rfile() 329
READBIN() 327
READWAV() 328
wfile() 342

Numerische Methoden

al_convr1d() 250
al_fftc1dinv() 251
Bisection() 255
FindRoot() 278
GaussNewton.CDGSS() 283
GoldenSection...min() 286
GradientDescent() 289
HRE.NR() 291
LevenbergMarquardt.CD() 303
NCGM.CD() 311
NewtonMethod.CDGSS() 312
NewtonRaphson.CD() 313
rkfixed() 331
al_convr1dinv() 250
al_nleqsolve() 252
Brent() 256
GaussNewton() 283
GaussNewton.GSS() 284
GradientAscent() 288
GradientDescent.GSS() 289
HRE.RK() 291
mapUnknowns() 305
NewtonMethod() 311
NewtonMethod.GSS() 312
Ridder() 330
Secant() 331
al_fftc1d() 251
BDQRF() 255
Broyden() 256
GaussNewton.CD() 283
GoldenSection...max() 285
GradientAscent.GSS() 288
HRE.B() 291
LevenbergMarquardt() 303
NCGM() 311
NewtonMethod.CD() 311
NewtonRaphson() 312
Rkadapt() 330

Wahrscheinlichkeitsrechnung und Statistik

Grafik

Maxima-Schnittstelle

Sonstiges

D.2. Beschreibung

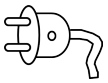
```
abs(Zahl)
```

Absoluter Betrag der Zahl.

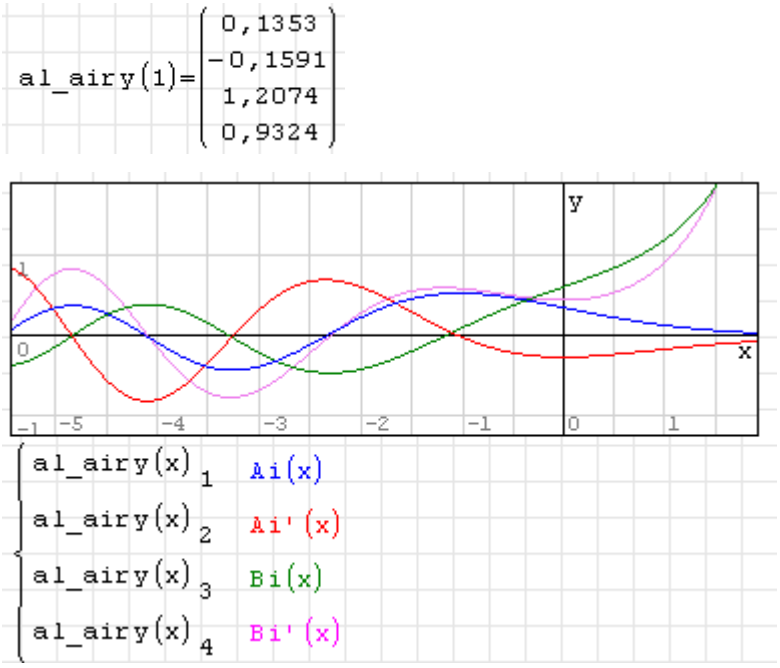
```
ainterp(X-Daten; Y-Daten; x)
```

kubische Spline-Interpolation der Vektoren `X-Daten` und `Y-Daten` am Punkt x nach Akima Erläuterung des Verfahrens (TU Wien). Das Verfahren berechnet vor der Interpolation an den Stützstellen die Neigung aus einer 5-Punkte-Umgebung.

```
al_airy(x)
```

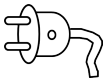


AlgLib-Implementation der Airy-Funktionen und ihrer Ableitungen (Alglib-Dokumentation) für reelle Zahlen. Ergebnis ist ein Vektor mit den vier Elementen $Ai(x)$, $Ai'(x)$, $Bi(x)$ und $Bi'(x)$.

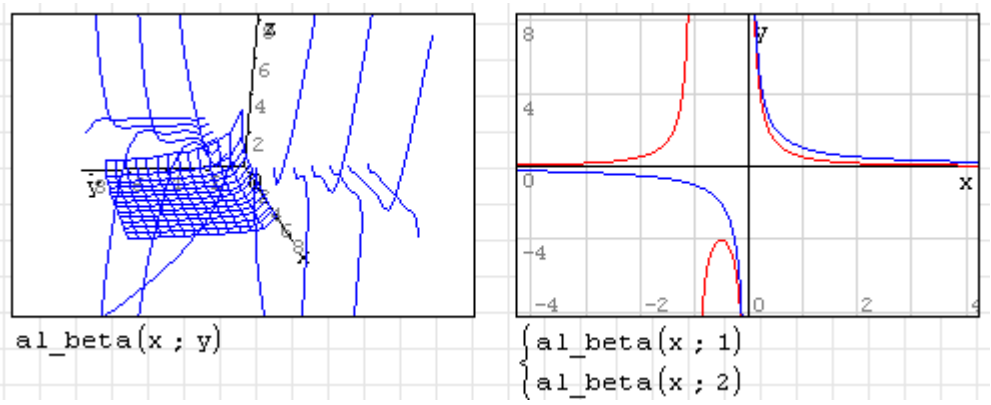


Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

```
al_beta(p,q)
```



AlgLib-Implementation der Beta-Funktion (Alglib-Dokumentation) für $p, q > 0$. Entgegen der Spezifikation liefert die Funktion auch Ergebnisse, wenn eines der beiden Argumente kleiner als Null ist.

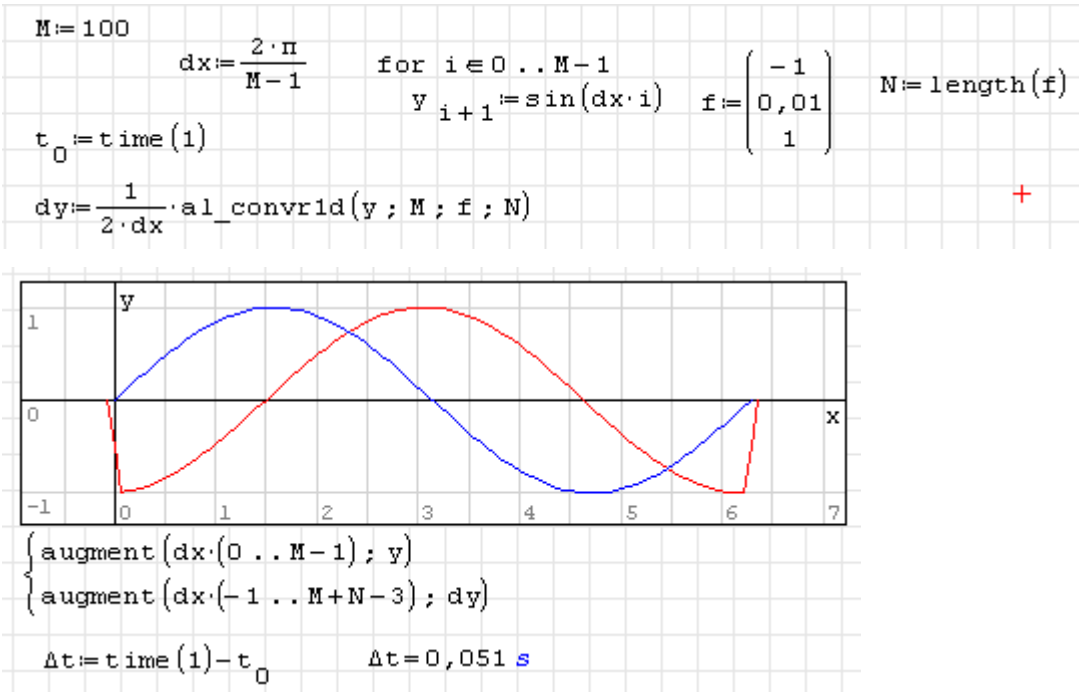


Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

```
al_convr1d(A; B)
al_convr1d(A; LängeA; B; LängeB)
```

Reelle eindimensionale Faltung (Alglib-Dokumentation) der beiden Vektoren `A` und `B`. Ergebnis ist ein Vektor der Länge `LängeA+LängeB-1`. Die Version mit vier Argumenten bietet keine Vorteile gegenüber derjenigen mit zwei Argumenten, außer dass sie den internen Aufruf 1:1 abbildet.

Das Beispiel zeigt die numerische Ableitung mit dem zentralen Differenzenquotienten.



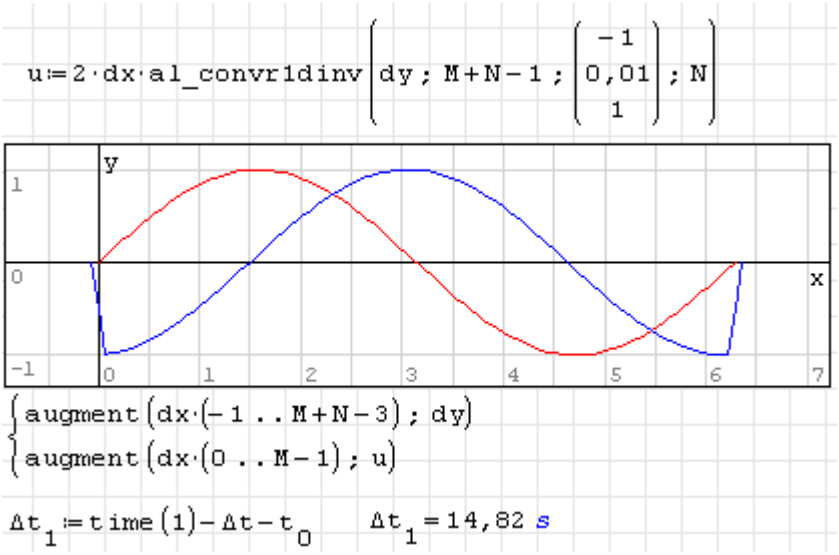
Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

```
al_convr1dinv(A; LängeA; B; LängeB)
al_convr1dinv(A; LängeA; B; LängeB)
```

Inverse reelle eindimensionale Faltung (Alglib-Dokumentation) der beiden Vektoren `A` und `B`., Umkehrfunktion von `al_convr1d()`. Ergebnis ist ein Vektor der Länge

LängeA-LängeB+1. Die Version mit vier Argumenten bietet keine Vorteile gegenüber derjenigen mit zwei Argumenten, außer dass sie den internen Aufruf 1:1 abbildet.

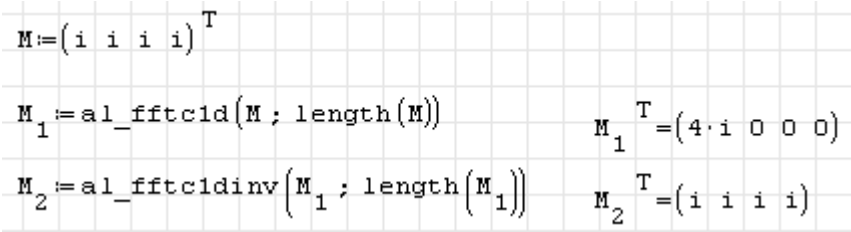
Das Beispiel ist eine Fortsetzung des Beispiels für die Funktion al_convr1d() (siehe Seite 5).



Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

al_fftc1d(A; LängeA)

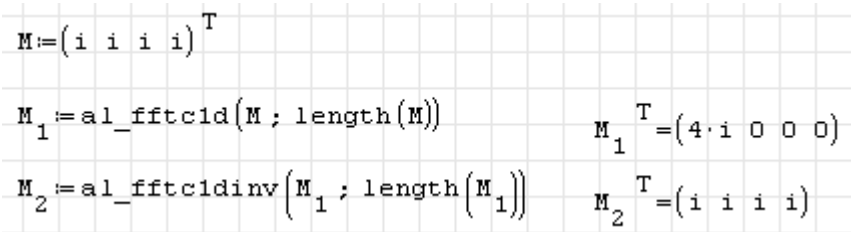
Schnelle Fourier-Transformation (Alglib-Dokumentation) des komplexen Vektors A der Länge LängeA. Ergebnis ist ein Vektor gleicher Länge. In der gegenwärtigen Version (28.4.2013) muss die Länge des Vektors angegeben werden, obwohl diese innerhalb von SMath bekannt ist.



Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

al_fftc1dinv(A; LängeA)

Schnelle inverse Fourier-Transformation (Alglib-Dokumentation) des komplexen Vektors A der Länge LängeA. Ergebnis ist ein Vektor gleicher Länge. In der gegenwärtigen Version (28.4.2013) muss die Länge des Vektors angegeben werden, obwohl diese innerhalb von SMath bekannt ist.



Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2).

```
al_nleqsolve(x0; StepMax; epsF; f(v); j(v))
```

Nichtlinearer Löser (Alglib-Dokumentation)

- `x0` Anfangswerte der unabhängigen Variablen
- `StepMax` Maximale Schrittlänge, Null entspricht unbegrenztem Schritt.
- `epsF` Abbruchtoleranz für die Quadratsumme der Funktionswerte.
- `f(v)` zu lösende Funktion des Unbekanntenvektors `v`
- `j(v)` Jacobimatrix der Funktion

$$F(\beta ; x ; y) := \begin{pmatrix} y\cdot tg(a) - w + x \\ tg(\beta)\cdot(r-x) - y - z \\ r^2 - (r-x)^2 - (y+z)^2 \end{pmatrix} \qquad f(v) := F(v_1 ; v_2 ; v_3)$$

$$Jac(\beta ; x ; y) := Jacob\left(\begin{pmatrix} y\cdot tg(a) - w + x \\ tg(\beta)\cdot(r-x) - y - z \\ r^2 - (r-x)^2 - (y+z)^2 \end{pmatrix} ; \begin{pmatrix} \beta \\ x \\ y \end{pmatrix}\right) \qquad j(v) := Jac(v_1 ; v_2 ; v_3)$$

$$a := 0{,}025 \qquad w := 0{,}005 \qquad r := 0{,}125 \qquad z := 0{,}000135$$

$$\beta_0 := 0{,}1 \qquad x_0 := 0{,}01 \qquad y_0 := 0{,}001 \qquad X_0 := stack(\beta_0 ; x_0 ; y_0)$$

$$StepMax := 0 \qquad Eps := 10^{-7}$$

$$res := al\_nleqsolve(X_0 ; StepMax ; Eps ; f(v) ; j(v))$$

$$res = \begin{pmatrix} 0{,}26 \\ 0{,}0042 \\ 0{,}032 \end{pmatrix} \qquad F(res_1 ; res_2 ; res_3) = \begin{pmatrix} -9{,}8227\cdot 10^{-10} \\ -1{,}8648\cdot 10^{-10} \\ 3{,}7394\cdot 10^{-9} \end{pmatrix}$$

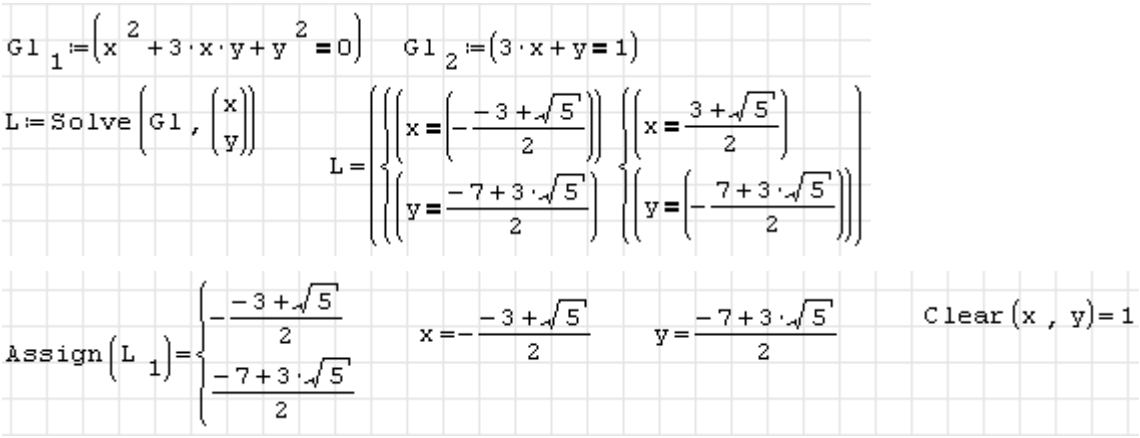
Die Funktion ist Bestandteil des Plugins *AlgLib* (siehe Abschnitt 9.4.2). Das Beispiel ist der mitgelieferten Dokumentation entnommen.

```
alg(Matrix; i; j)
```

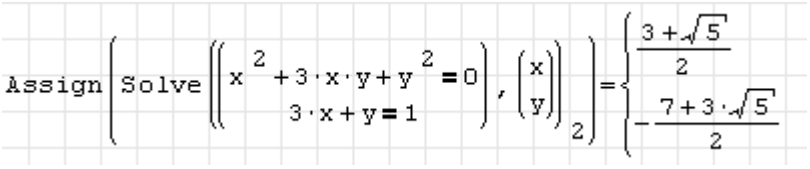
Algebraisches Komplement der Matrix zum Element i,j.

```
Assign(L)
```

Interpretiert die im Ausdruck `L` gegebenen boolschen Gleichungen der Form `var=expr` als Zuweisungen. Die Variablen dürfen vorher keine Werte haben, sind also gegebenenfalls mit `Clear()` zu löschen. Die Funktion dient der Zuweisung von Ergebnissen der Lösungsfunktionen.

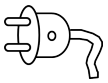


Die Funktion kann auch direkt angewendet werden (das setzt natürlich voraus, dass man bei mehreren Lösungen die richtige auswählt).



Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

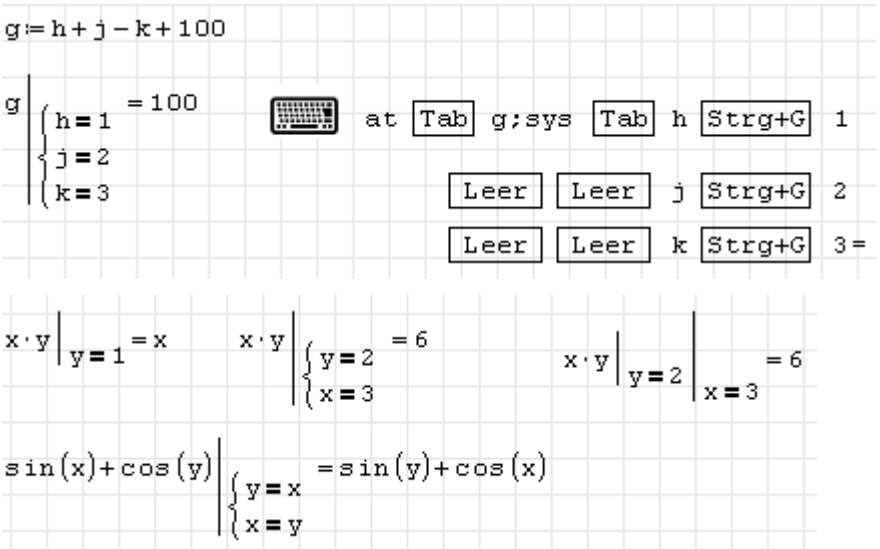
```
at(expr; subst)
at(expr; subst_unten; subst_oben)
```



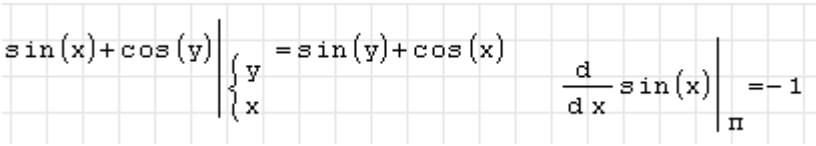
Substitutionsfunktion, ersetzt Variablen im Ausdruck `expr` durch andere Ausdrücke oder Zahlenwerte.

`expr` Ausdruck, in den Werte eingesetzt werden sollen

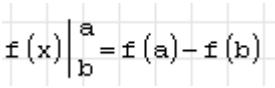
`subst` Werte, die für die Variablen in expr einzusetzen sind. Sie werden als Boolsche Gleichung `Variable=Wert` angegeben, mehrere Ersetzungen werden als Liste zusammengefasst.



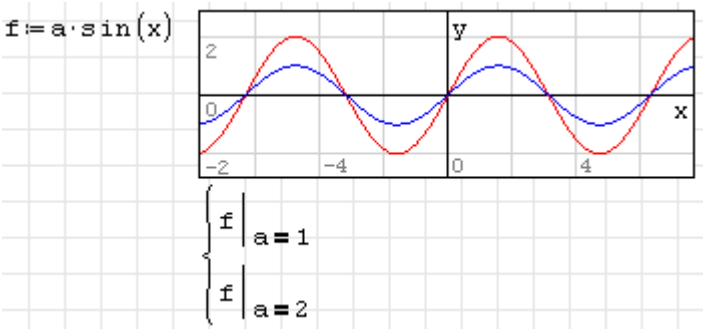
Es können auch einfach Werte angegeben werden, deren Anzahl muss dann mit der Variablenanzahl in expr übereinstimmen. Die Werte werden in alphabetischer Reihenfolge eingesetzt.



Wird die Funktion mit drei Argumenten benutzt, dann wird die Differenz (oberer Wert minus unterer Wert) gebildet:



Die Substitution kann häufig da angewendet werden, wo man ansonsten einen Ausdruck als Funktion mit Argumentliste definieren müsste.

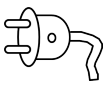


Die Funktion ist Bestandteil des Plugins *Functions Extensions* (siehe Abschnitt 9.4.6).

```
augment(Matrix1; Matrix2)
```

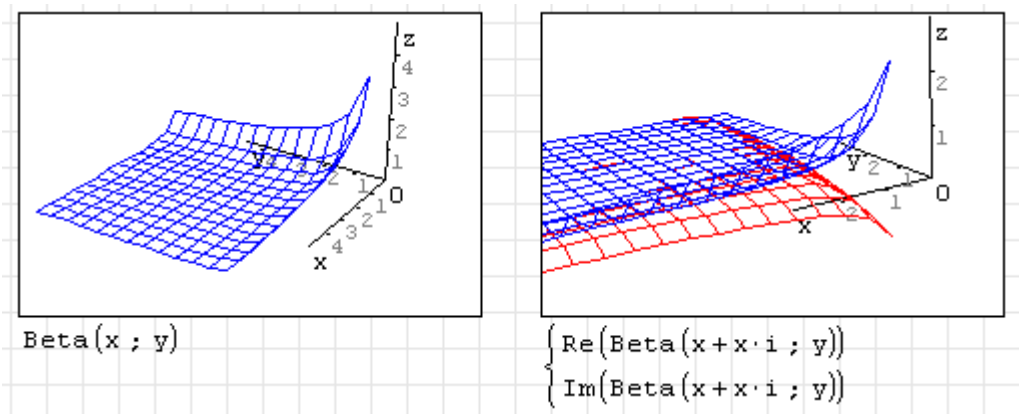
Nebeneinanderstellen von `Matrix1` und `Matrix2`. Beide Matrizen müssen gleiche Zahl von Zeilen haben.

```
Beta(p; q)
```



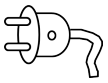
Betafunktion zweier komplexer Zahlen mit positivem Realteil, Eulersches Integral 1. Art:

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} = \int_0^1 t^{p-1}(1-t)^{q-1}\mathrm{d}t$$



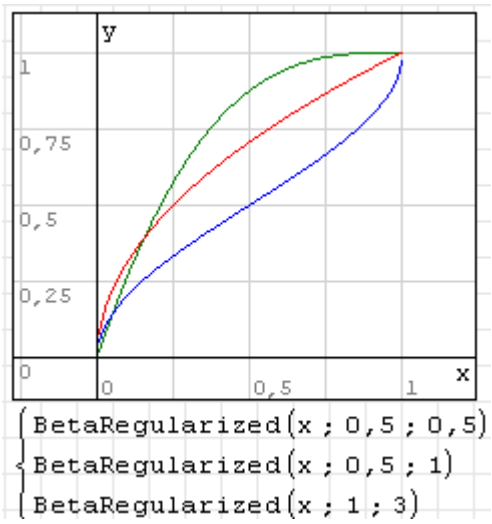
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
BetaRegularized(z; p; q)
```



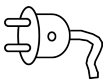
Regularisierte unvollständige Betafunktion $I(z,p,q)$ dreier komplexer Zahlen. Der Betrag der Integrationsobergrenze z muss kleiner als 1 sein, die Realteile der Parameter p und q müssen wie bei der Betafunktion selbst positiv sein. Sie ist definiert als Quotient aus der unvollständigen Betafunktion $B_z(p,q)$ und der Betafunktion $B(p,q)$

$$I(z,p,q) = \frac{B_z(p,q)}{B(p,q)} \quad \text{mit} \quad B_z(p,q) = \int_0^z t^{p-1}(1-t)^{q-1}\,\mathrm{d}t$$



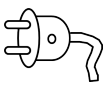
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
BDQRF(Funktion; Untergrenze; Obergrenze; Genauigkeit; maxIter)
BDQRF(Funktion; Untergrenze; Obergrenze; Genauigkeit; maxIter; Schalter)
```

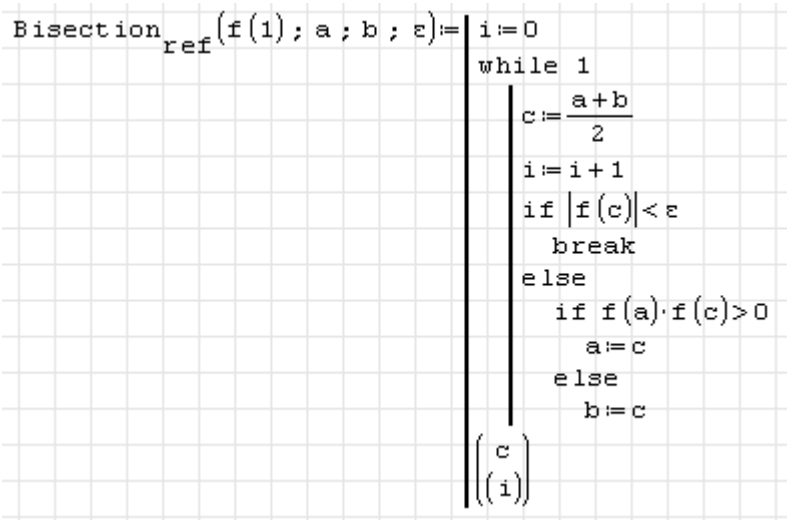


Nichtlinearer Löser nach dem Bisected Direct Quadratic Regula Falsi Verfahren. Erfordert Plugin *Non Linear Solvers* (Seite 218). Wenn „Schalter" angegeben und ungleich Null ist, wird auch die Zahl der Iterationen ausgegeben

```
Bisection(Funktion; Untergrenze; Obergrenze; Genauigkeit)
Bisection(Funktion; Untergrenze; Obergrenze; Genauigkeit; Schalter)
```



Nichtlinearer Löser nach der Halbteilungsmethode. Wenn „Schalter" angegeben und ungleich Null ist, wird auch die Zahl der Iterationen ausgegeben. Hier eine Referenzimplementierung des Verfahrens:

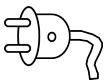


Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (Seite 218).

```
break
```

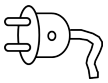
Bricht eine unmittelbar übergeordnete Schleife ab. Ein Anwendungsbeispiel finden Sie in der Referenzimplementierung des Halbteilungsverfahrens (Funktion `Bisection()`)

```
Brent(Funktion; Untergrenze; Obergrenze; Genauigkeit; MaxIter)
Brent(Funktion; Untergrenze; Obergrenze; Genauigkeit; MaxIter; Schalter)
```



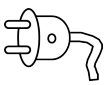
Nichtlinearer Löser nach Brent. Wenn „Schalter” angegeben und ungleich Null ist, wird auch die Zahl der Iterationen ausgegeben. Erfordert Plugin *Non Linear Solvers* (Seite 218).

```
Broyden(Funktion; Untergrenze; Obergrenze; Genauigkeit; MaxIter)
Broyden(Funktion; Untergrenze; Obergrenze; Genauigkeit; MaxIter; Schalter)
```



Nichtlinearer Löser nach Broyden. Wenn „Schalter” angegeben und ungleich Null ist, wird auch die Zahl der Iterationen ausgegeben. Erfordert Plugin *Non Linear Solvers* (Seite 218).

```
cases(expr; cond; expr0)
cases(expr1; cond1; expr2; cond2; expr0)
```



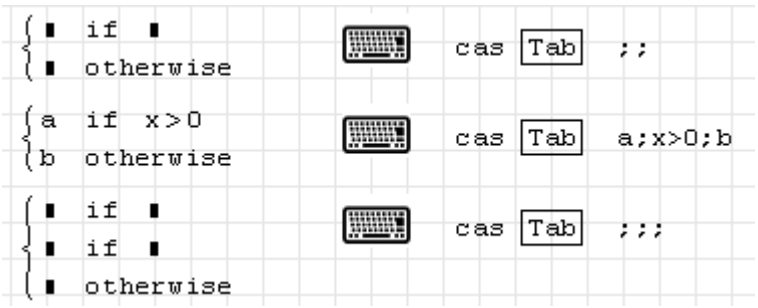
Fallunterscheidung.

`expr` Ergebnis wenn Bedingung cond erfüllt ist.

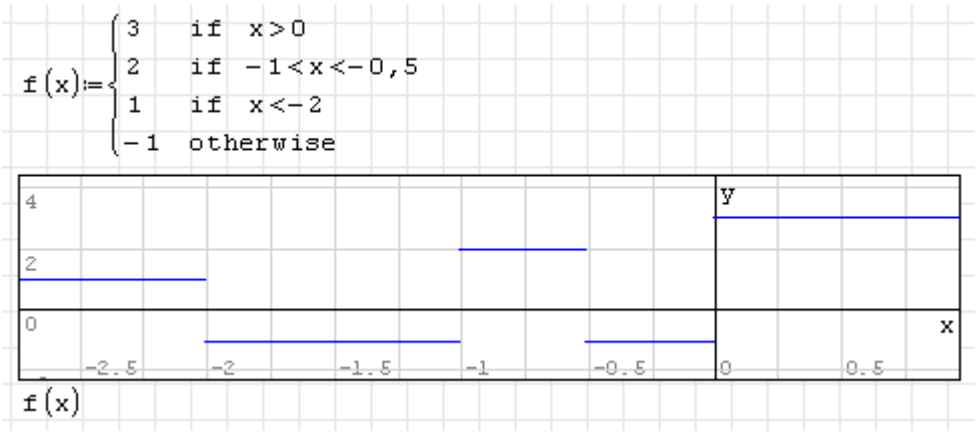
`cond` Bedingung, gilt als wahr, wenn der Wert ungleich Null ist

`expr0` Ergebnis, wenn keine Bedingung erfüllt ist.

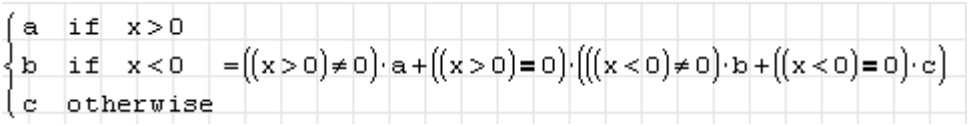
Tastatureingabe: Funktionsname `cases()` und die entsprechende Anzahl Argumenttrenner



Eine naheliegende Anwendung ist die Definition stückweise stetiger Funktionen:

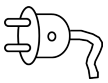


Die Funktion wird intern als Boolscher Ausdruck dargestellt:



Die Funktion ist Bestandteil des Plugins *Functions Extensions* (siehe Abschnitt 9.4.6).

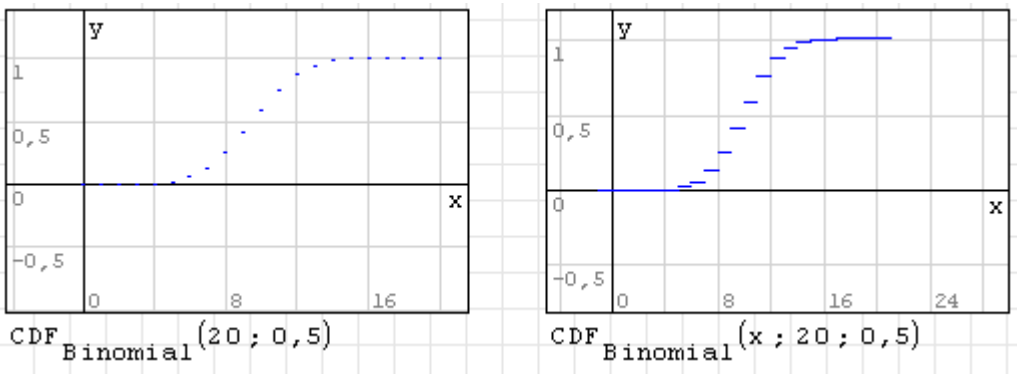
```
CDF.Binomial(n; p)
CDF.Binomial(k; n; p)
```



Verteilungsfunktion der (diskreten) Binomialverteilung. Sie beschreibt die Anzahl der Erfolge in einer Serie von gleichartigen und unabhängigen Versuchen, die jeweils genau zwei mögliche Ergebnisse haben („Erfolg" oder „Misserfolg"). p ist die Erfolgswahrscheinlichkeit bei einem Versuch, n ist die Zahl der Versuche.

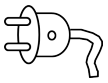
$$F(k,p,n)=\sum_{i=1}^{k}\binom{n}{i}p^i(1-p)^{n-i}=\sum_{i=1}^{k}\frac{n!}{i!\,(n-i)!}p^i(1-p)^{n-i}$$

Wird das Argument k nicht angegeben, dann liefert die Funktion eine plotbare zweispaltige Matrix aus Paaren von k und $F(k,p,n)$ mit $k\in 0\ldots n$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

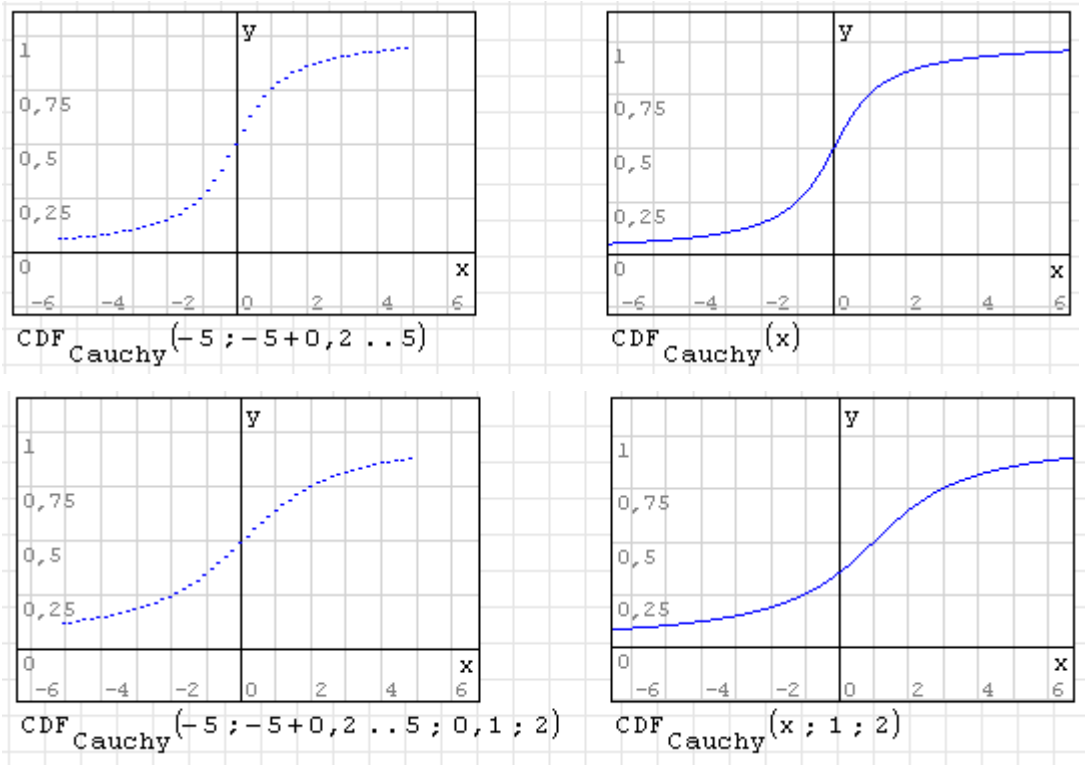
```
CDF.Cauchy(x)
CDF.Cauchy(x, t, s)
```



Verteilungsfunktion der Cauchy-Verteilung mit mit dem Ortsparameter t (Standardwert 0) und dem Breitenparameter s (Standardwert 1):

$$F(x,t,s) = \frac{1}{2} + \frac{1}{\pi}\arctan\frac{x-t}{s}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,t,s)$.



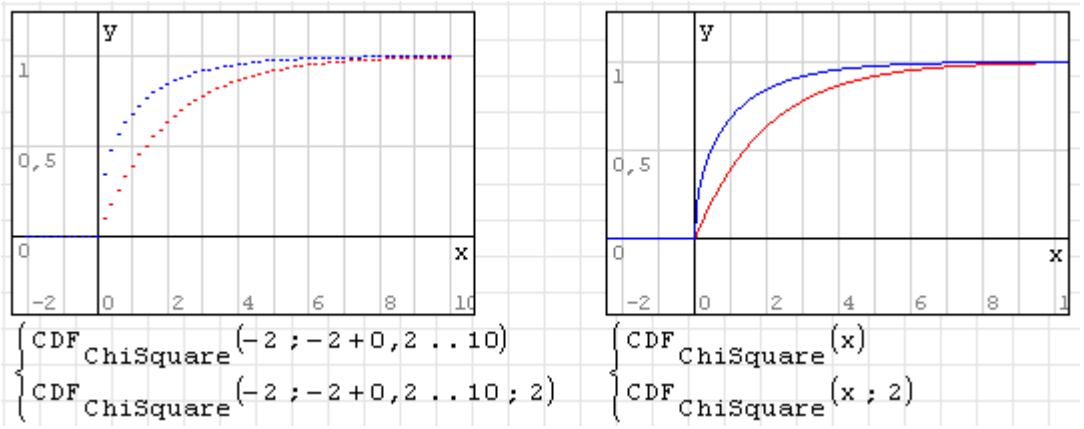
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.ChiSquare(x)
CDF.ChiSquare(x, n)
```

Verteilungsfunktion der χ^2_n- (Chi-Quadrat-)Verteilung mit der Zahl der Freiheitsgrade n (Standardwert 1). Sie beschreibt die Wahrscheinlichkeit, dass die Quadratsumme von n standardnormalverteilten Zahlen kleiner oder gleich x ist. Sie kann mit Hilfe der regularisierten unvollständigen Gammafunktion P dargestellt werden:

$$\chi^2_n(x) = F(x,n) = P\left(\frac{n}{2}, \frac{x}{2}\right)$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,n)$.



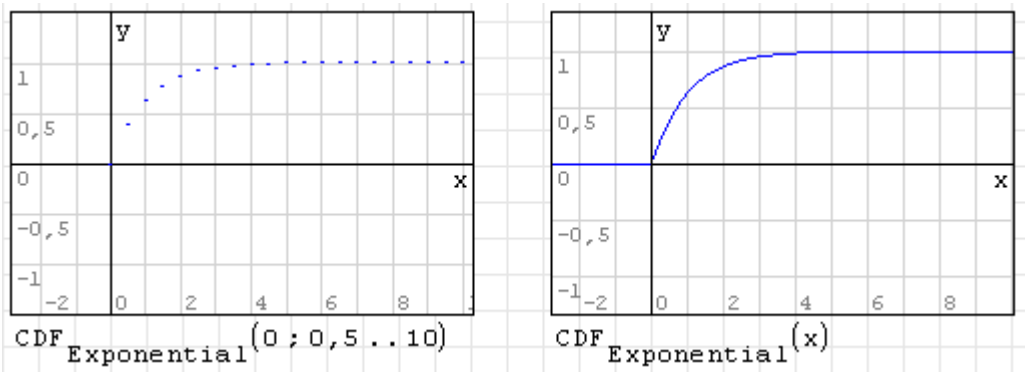
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

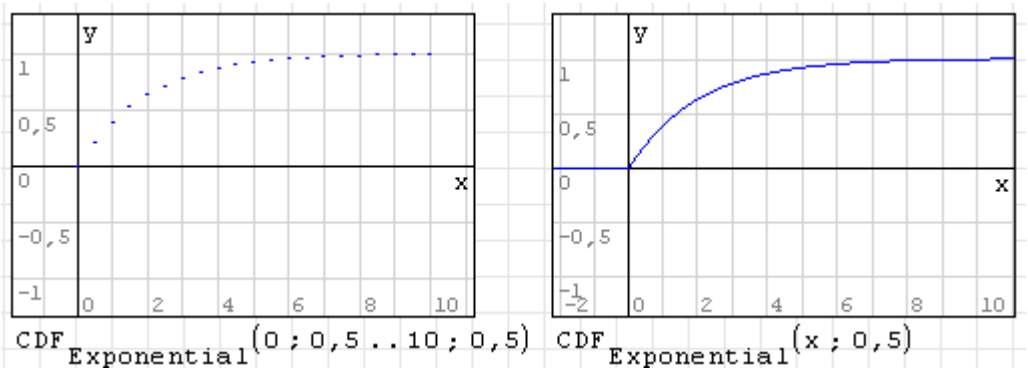
```
CDF.Exponential(x)
CDF.Exponential(x; Ereignisrate)
```

Verteilungsfunktion der Exponentialverteilung mit mittlerer Ereignisrate λ. Gibt die Wahrscheinlichkeit des Auftretens des nächsten Ereignisses innerhalb eines Bereichs von 0 bis x an. Voreinstellung der Ereignisrate ist $\lambda = 1$. Es gilt:

$$F(x,\lambda) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,\lambda)$.





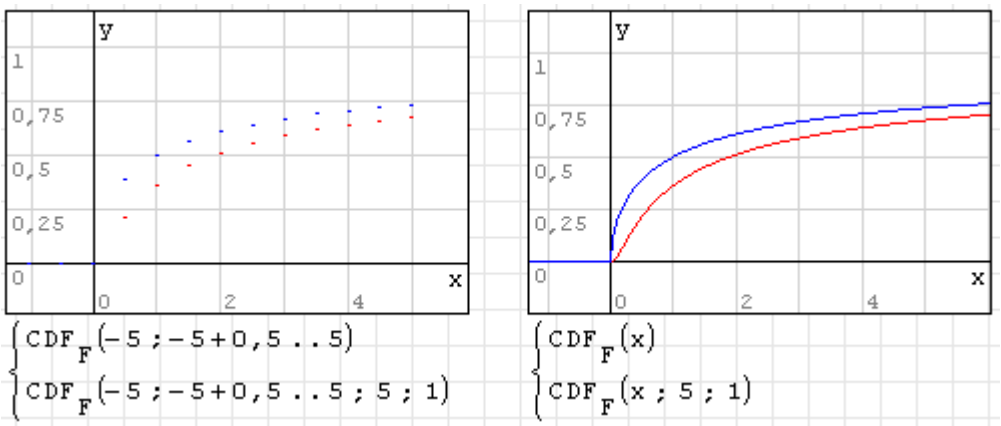
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.F(x)
CDF.F(x,m,n)
```

Verteilungsfunktion der Fisher- oder F-Verteilung mit m Freiheitsgraden im Zähler und n Freiheitsgraden im Nenner. Sie entspricht der Verteilung des Quotienten aus zwei normierten Quadratsummen standardnormalverteilter Zahlen. Die Verteilung wird mit der regularisierten unvollständigen Betafunktion $I(z,p,q)$ (siehe Seite 255) berechnet.

$$F_{m,n}(x) = \frac{\chi^2_m/m}{\chi^2_n/n} = I\left(\frac{xm}{xm+n}, \frac{m}{2}, \frac{n}{2}\right).$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F_{m,n}(x)$.



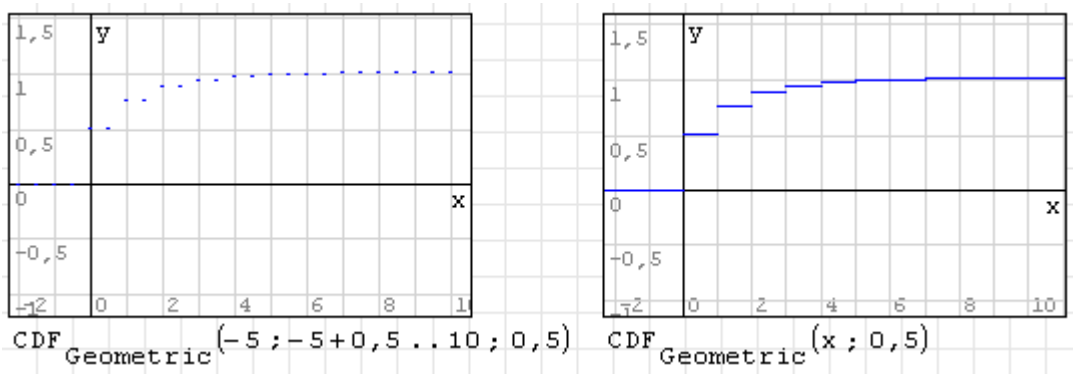
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.Geometric(x; p)
```

Verteilungsfunktion der Geometrischen Verteilung. Sie gibt die Wahrscheinlichkeit an, maximal x Fehlversuche zu benötigen, bevor ein Versuch mit der Erfolgswahrscheinlichkeit p erfolgreich ist. Es gilt

$$F(x,p) = 1-(1-p)^x$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,p)$.



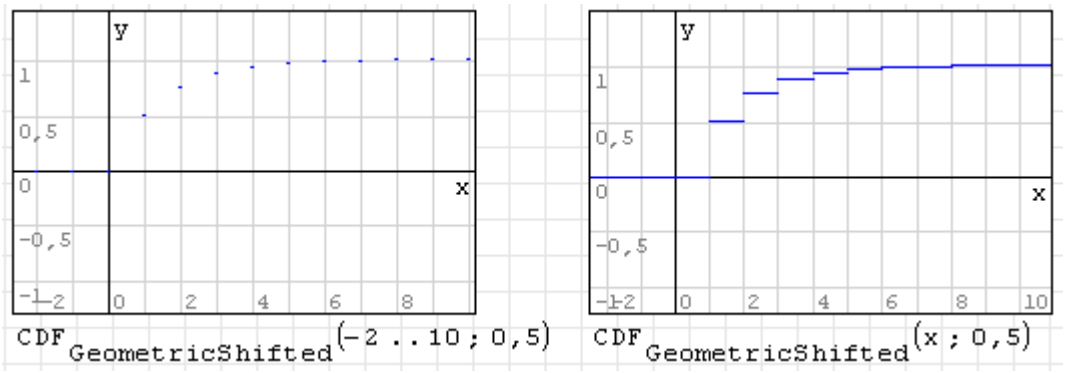
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.GeometricShifted(x; p)
```

Häufigkeitsfunktion der verschobenen Geometrischen Verteilung. Sie gibt die Wahrscheinlichkeit an, maximal x Versuche der Erfolgswahrscheinlichkeit p zu benötigen, damit der Erfolg eintritt. Es gilt

$$F(x,p) = 1-(1-p)^{x-1}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,p)$.



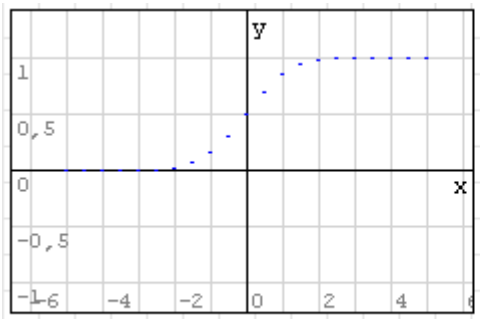
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.Normal(x)
CDF.Normal(x; Mittelwert)
CDF.Normal(x; Mittelwert; Standardabweichung)
```

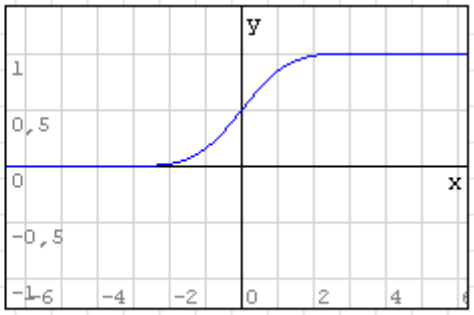
Verteilungsfunktion der Normalverteilung, Voreinstellung ist Mittelwert $\mu = 0$ und Standardabweichung $\sigma = 1$ (Standardnormalverteilung). Es gilt:

$$F(x,\mu,\sigma) = \frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

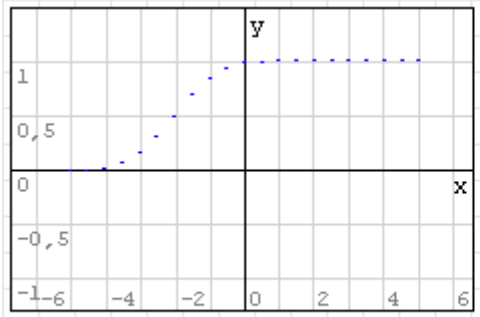
Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $F(x,\mu,\sigma)$ erzeugt.



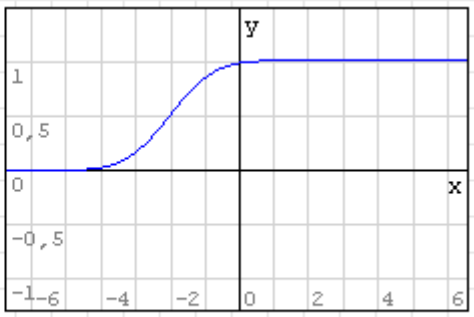
CDF$_{Normal}$(−5 ; −5+0,5 .. 5)



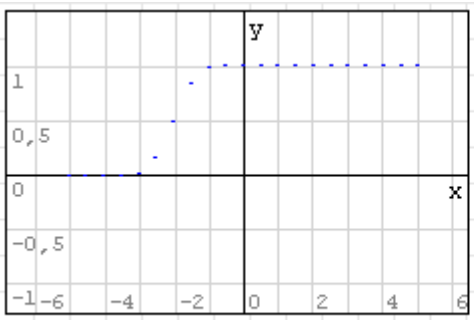
CDF$_{Normal}$(x)



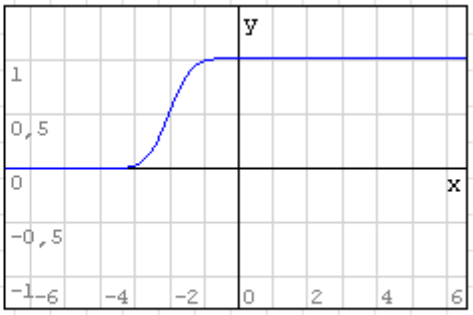
CDF$_{Normal}$(−5 ; −5+0,5 .. 5 ; −2)



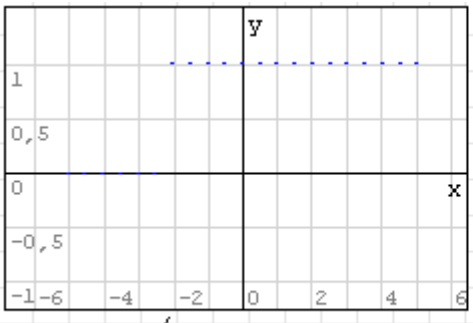
CDF$_{Normal}$(x ; −2)



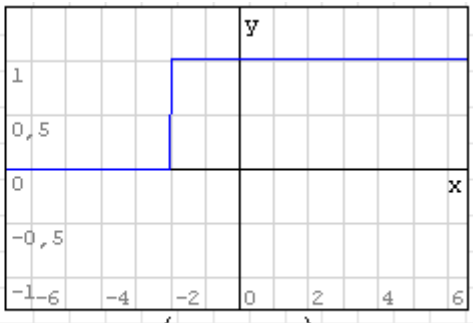
CDF$_{Normal}$(−5 ; −5+0,5 .. 5 ; −2 ; 0,5)



CDF$_{Normal}$(x ; −2 ; 0,5)



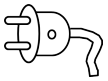
CDF$_{Normal}$(−5 ; −5+0,5 .. 5 ; −2 ; 0)



CDF$_{Normal}$(x ; −2 ; 0)

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.Poisson(x)
CDF.Poisson(x; lambda)
```

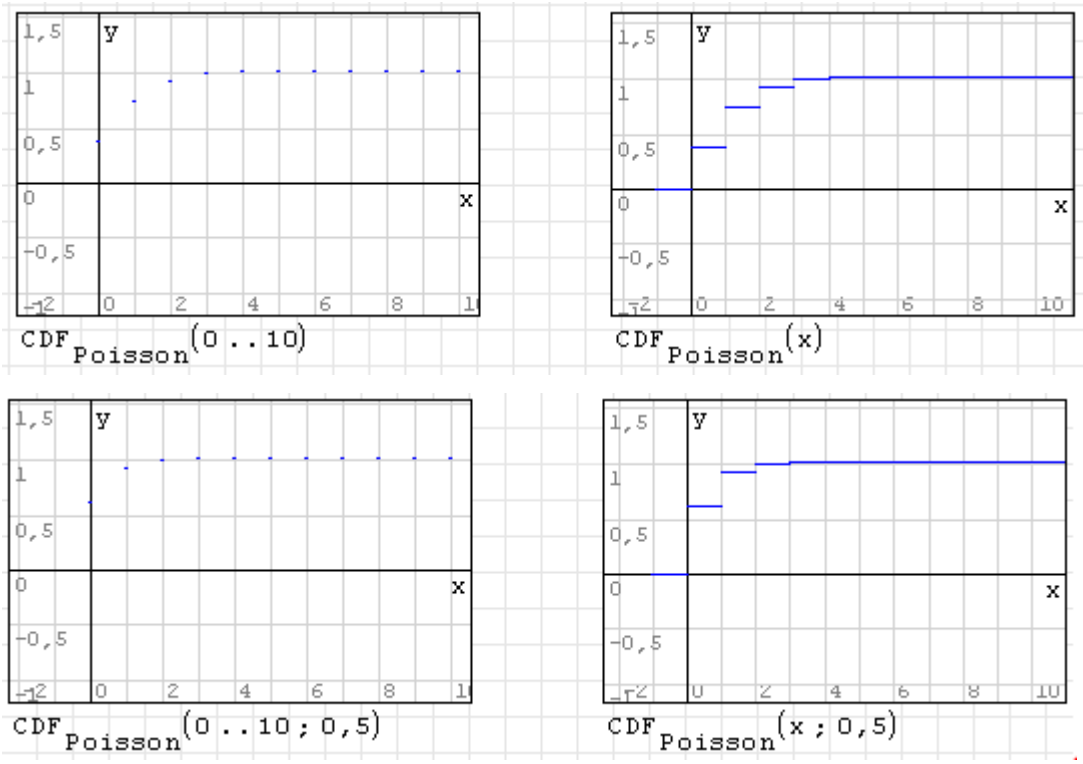


Verteilungsfunktion der (diskreten) Poisson-Verteilung. Gibt die Wahrscheinlichkeit an, innerhalb eines Zeitraums oder eines Gebiets höchstens x seltene Ereignisse anzutreffen,

wenn das Ereignis im Mittel lambda mal in diesem Zeitraum oder Gebiet erwartet wird. Voreinstellung ist $\lambda = 1$. Es gilt:

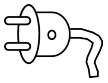
$$F(x,\lambda) = e^{-\lambda} \sum_{k=0}^{x} \frac{\lambda^k}{k!}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $F(x,\lambda)$. x muss ganzzahlig sein.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

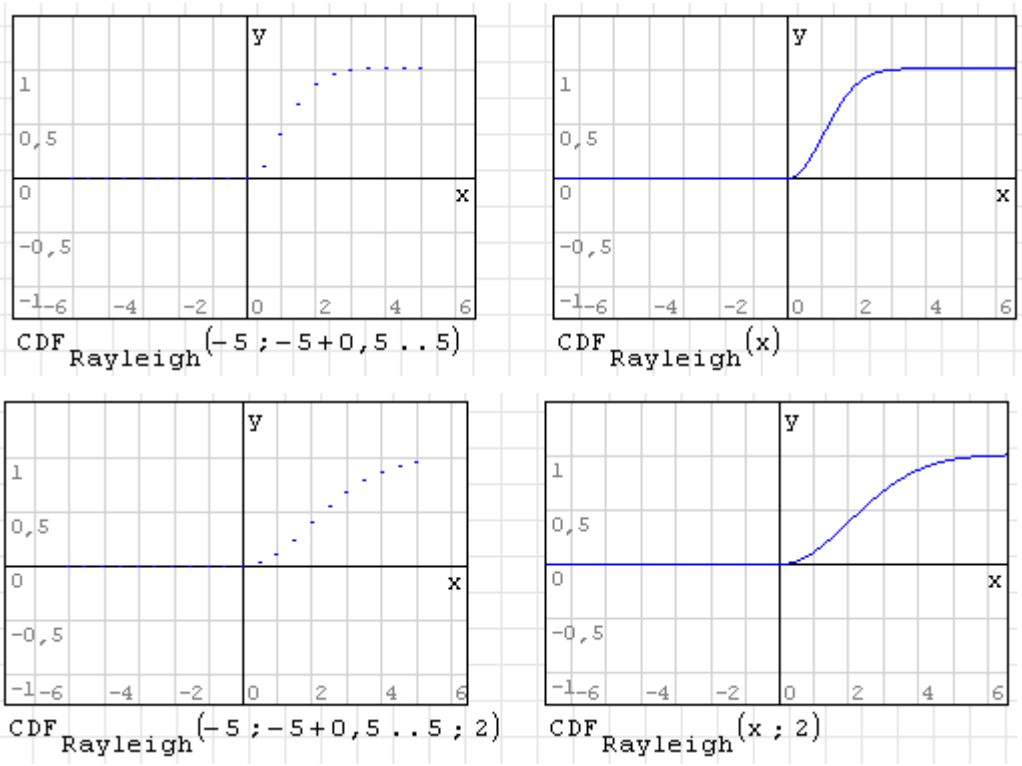
```
CDF.Rayleigh(x)
CDF.Rayleigh(x; σ)
```



Verteilungsfunktion der Rayleigh-Verteilung. Gibt die Wahrscheinlichkeitdichte an, dass ein Vektor aus zwei mit $\mu = 0$ und σ normalverteilten Komponenten höchstens den Betrag x hat. Die Standardabweichung ist auf $\sigma = 1$ voreingestellt. Es gilt:

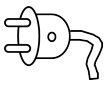
$$F(x,\sigma) = \begin{cases} 1 - e^{-x^2/2\sigma^2} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $F(x,\sigma)$ erzeugt.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

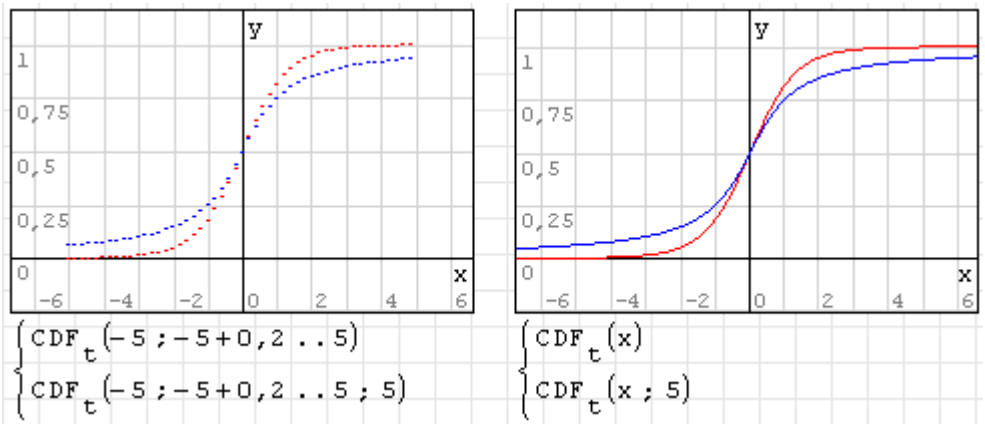
```
CDF.t(x)
CDF.t(x,n)
```



Verteilungsfunktion der Studentschen t-Verteilung mit n Freiheitsgraden (Standardwert 1). Die Verteilung wird mit der regularisierten unvollständigen Betafunktion $I(z,p,q)$ (siehe Seite 255) berechnet.

$$F(x,n) = I\left(\frac{x+\sqrt{x^2+n}}{2\sqrt{x^2+n}}, \frac{n}{2}, \frac{n}{2}\right).$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $F(x,\nu)$ erzeugt.



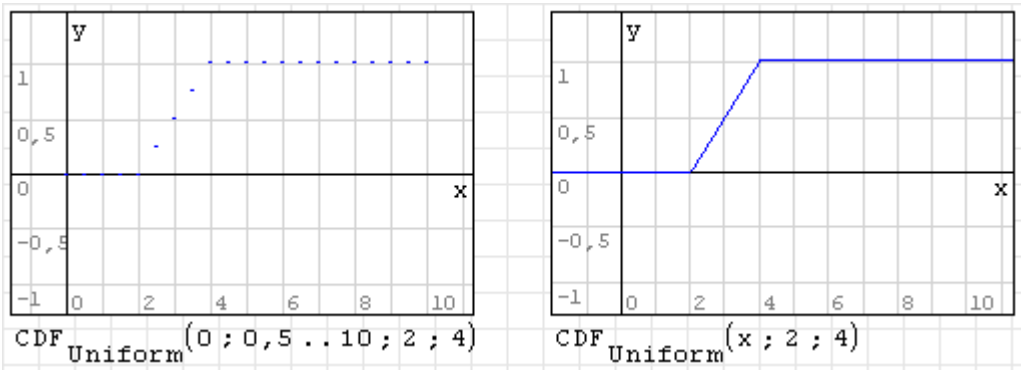
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`CDF.Uniform(x; a; b)`

Verteilungsfunktion der stetigen Gleichverteilung im Intervall $[a, b]$ mit

$$F(x,a,b) = \begin{cases} 0 & x < a \\ 1 & x > b \\ \frac{x-a}{b-a} & \text{sonst} \end{cases}$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $F(x,a,b)$ erzeugt.



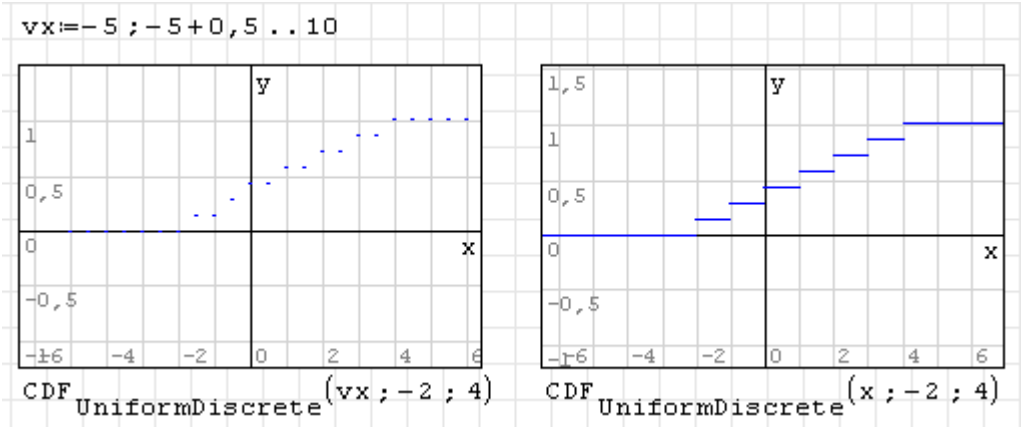
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`CDF.UniformDiscrete(x; a; b)`

Verteilungsfunktion der diskreten Gleichverteilung im Intervall $[a, b]$ mit

$$F(x,a,b) = \begin{cases} \frac{\lfloor x \rfloor - a + 1}{b-a+1} & a \le x \le b \\ 0 & \text{sonst} \end{cases}$$

Darin sind x, a und b ganze Zahlen. Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $F(x,a,b)$ erzeugt.



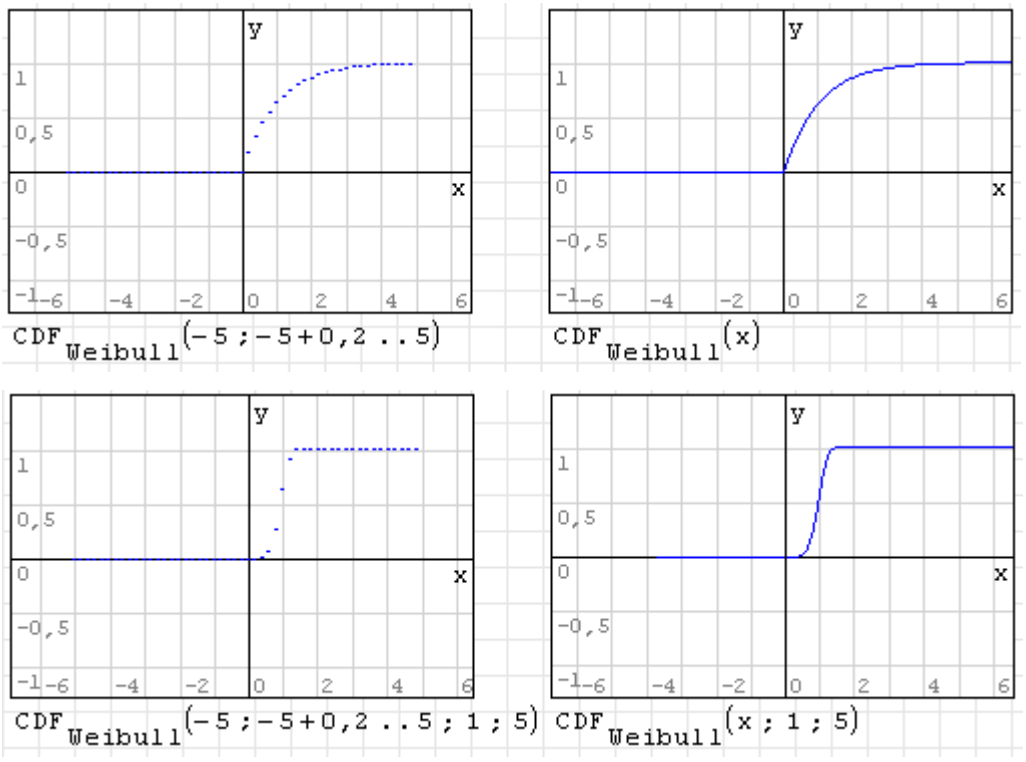
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
CDF.Weibull(t)
CDF.Weibull(t; T; k)
```

Verteilungsfunktion der Weibullverteilung mit der charakteristischen Lebensdauer T (Voreinstellung $T = 1$) und dem Streuungsparameter k (Voreinstellung $k = 1$). Es gilt:

$$F(t, T, k) = 1 - e^{-(t/T)^k} \quad t \geq 0.$$

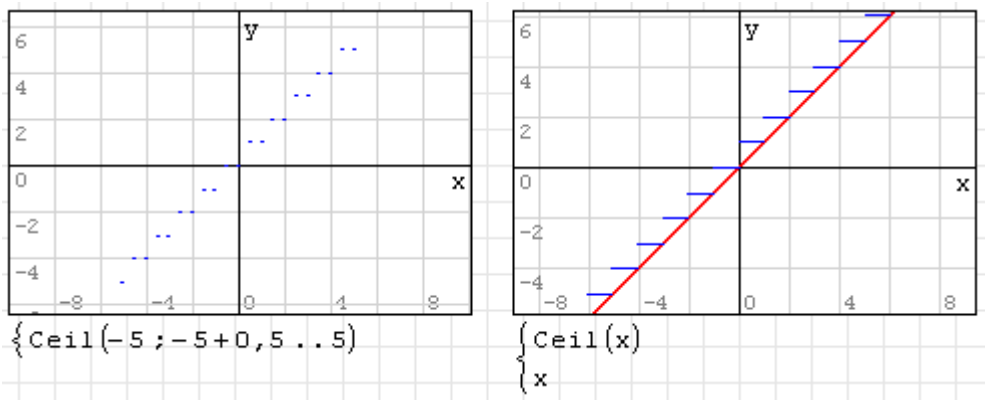
Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren t und $F(t, T, k)$ erzeugt.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Ceil(x)
```

Liefert die nächste ganze Zahl, die größer oder gleich x ist. Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und Ceil(x) erzeugt.

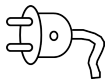


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

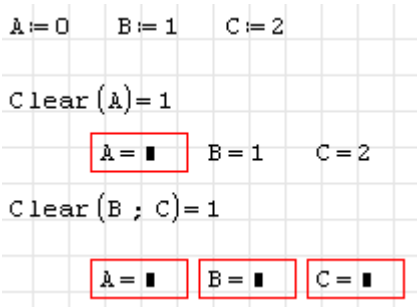
```
cinterp(X-Daten; Y-Daten; x)
```

Kubische Spline Interpolation der Daten am Punkt x

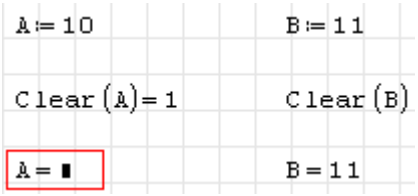
```
Clear(Variable1; Variable2;...)
```



Löscht die Werte der Variablen.



Die Funktion wird nur ausgeführt, wenn das Ergebnis abgefragt wird, daher reicht die Anweisung `Clear()` allein nicht aus, um eine Variable zu löschen.

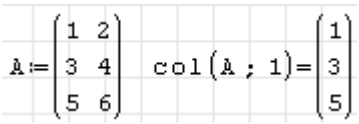


Dabei wird ein Nebeneffekt der line()-Funktion verwendet. Dadurch ist die Variable bis zur Zuweisung anderer Datentypen vom Typ line(), also ein leerer Anweisungsblock.

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

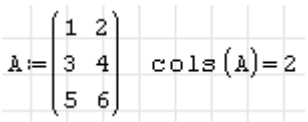
```
col(Matrix; i)
```

liefert die Spalte i der Matrix.




```
cols(Matrix)
```

Liefert die Zahl der Spalten der Matrix.



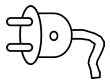
```
concat(Zeichenkette; Zeichenkette [; Zeichenkette...])
```

Verbindet die gegebenen Zeichenketten.

```
continue
```

Bricht den aktuellen Schleifendurchlauf ab, verlässt die Schleife aber nicht.

```
CreateMesh(Funktion; u0; u1; v0; v1; nu; nv)
```



Erzeugt ein plotfähiges Datenraster aus einer Funktion, die x-, y- und die z-Koordinaten einer Fläche als Funktion zweier unabhängiger Variablen. Beispiele siehe Forumbeitrag

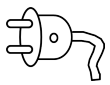
Die Funktion ist Bestandteil des Plugins *3D Plot Region* (siehe Abschnitt 9.4.1).

```
csort(Matrix; i)
```

Sortiert die Zeilen einer Matrix aufsteigend anhand der Spalte i.

$$\text{csort}\left(\begin{pmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{pmatrix};1\right)=\begin{pmatrix}1&2\\1&2\\3&4\\3&4\\5&6\end{pmatrix}$$

```
CurrentDirectory(Pfadname)
```



Ermittelt oder setzt das aktuelle Verzeichnis. Pfadname ist eine Zeichenkette. Ist sie leer (""), wird das aktuelle Verzeichnis als Zeichenkette geliefert, ansonsten wird als Argument ein Pfadname erwartet, auf den das aktuelle Verzeichnis zu setzen ist.

Rückgabewert ist immer das tatsächliche aktuelle Verzeichnis, so dass man prüfen kann, ob der Verzeichniswechsel erfolgreich war.

```
CWD := CurrentDirectory("")

CurrentDirectory("") = "D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"

CWD = "D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"

CurrentDirectory("d:\tmp") = "d:\tmp\"

CurrentDirectory(CWD) = "D:\FHB\Software\SMath\SMath Skript\SMath\MathcadFileAccess\"
```

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

```
description(Bezeichner)
```

Liefert das Beschreibungsattribut eines Bezeichners passend zur aktuellen Spracheinstellung der Benutzeroberfläche. Die Beschreibung kann im Kontextmenü (rechte Maustaste) der Definition angezeigt und geändert werden.

```
Das ist einfach ein Bezeichner
a

Das Attribut "description" des Bezeichners "a" wurde nicht gesetzt
description(a)=0

Hier wird ein Wert zugewiesen.
a:=1

Die Beschreibung an Zuweisungen setzt das Attribut "description"
description(a)="Hier wird ein Wert zugewiesen."

Noch eine Zuweisung
a:=2

Das Attribut "description" kann bei jeder Zuweisung
zum Bezeichner geändert werden.
description(a)="Noch eine Zuweisung"
```

det(Matrix)

Determinante der Matrix

$$\left|\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\right| = -2$$

Die numerische Berechnung von Determinante und Rang ist nicht robust, da die Zwischenergebnisse für die interne Zahlendarstellung zu groß werden. Im Einzelfall kann die symbolische Auswertung noch sinnvolle Werte liefern, stößt aber auch an Grenzen der Zahlendarstellung.

$$A := \begin{pmatrix} 8 & 0 & 4 & 8 & 8 & 2 & 8 & 3 & 3 & 2 \\ 5 & 3 & 0 & 7 & 4 & 7 & 5 & 4 & 6 & 7 \\ 5 & 8 & 4 & 1 & 8 & 3 & 1 & 8 & 6 & 9 \\ 6 & 3 & 5 & 0 & 0 & 3 & 8 & 9 & 8 & 0 \\ 5 & 8 & 3 & 5 & 8 & 3 & 4 & 6 & 6 & 5 \\ 9 & 6 & 7 & 2 & 8 & 9 & 6 & 3 & 9 & 5 \\ 1 & 9 & 6 & 4 & 5 & 1 & 8 & 0 & 6 & 1 \\ 9 & 5 & 5 & 5 & 1 & 7 & 5 & 8 & 2 & 4 \\ 8 & 4 & 2 & 1 & 9 & 9 & 5 & 2 & 1 & 1 \\ 6 & 5 & 9 & 3 & 9 & 5 & 1 & 8 & 0 & 7 \end{pmatrix}$$

numerische Auswertung:

rank(A)=

|A|=

symbolische Auswertung:

rank(A)=10

$$|A| = 16213093592340300000000$$

maple(|A|)=784894425

Maple findet die Determinante problemlos.

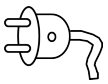
dfile(Dateiname)

Löscht eine mit `wfile()` (siehe Seite 247 geschriebene Datei, falls sie existiert. Liefert im Erfolgsfall 1 zurück, ansonsten 0. Beachten Sie, dass das Argument „Name” buchstäblich genommen wird. Es wird nicht ausgewertet und kann keinen Pfad oder Endung enthalten.

Die Datei wird in einem speziellen Systemverzeichnis vermutet (unter XP: *"C:\Documents and Settings\%USERNAME%\Application Data\SMath\user"*, unter Windows 7: *C:\Users\$Benutzer$\AppData\Roaming\SMath\user)*. Sie können sich dieses Verzeichnis mit der Funktion `SettingsDirectory()` (im Plugin *Custom Functions*, siehe Seite 9.4.3) anzeigen lassen.

Man kann also nicht beliebige Dateien im ganzen Dateisystem löschen. Sowohl die Beschränkung auf das oben genannte Verzeichnis als auch der Verzicht auf berechnete Dateinamen tragen zur Sicherheit gegen Schäden durch versehentliche oder böswillige Löschung wichtiger Dateien bei.

```
Diag(Matrix)
```



Liefert eine Diagonalmatrix. Hat die Argumentmatrix mehr als eine Spalte, ist das Ergebnis die Argumentmatrix, bei der alle Nichtdiagonalelemente auf Null gesetzt werden. Ist die Argumentmatrix ein Spaltenvektor, wird eine Matrix mit den Vektorelementen als Diagonalelementen erzeugt. Das ist dann gleichwertig zur eingebauten Funktion diag().

$$\mathrm{Diag}\left(\begin{pmatrix}1\\2\\3\end{pmatrix}\right)=\begin{pmatrix}1&0&0\\0&2&0\\0&0&3\end{pmatrix}\quad \mathrm{Diag}\left(\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}\right)=\begin{pmatrix}1&0&0\\0&5&0\\0&0&9\end{pmatrix}\quad \mathrm{Diag}\left(\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}\right)=\begin{pmatrix}1&0\\0&4\\0&0\end{pmatrix}$$

Diese Funktion ist Bestandteil des Plugins *Custom Functions* (Abschnitt 9.4.3).

```
diag(Spaltenmatrix)
```

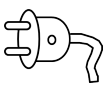
Liefert eine quadratische Matrix, deren Diagonaleinträge aus der Spaltenmatrix genommen werden.

$$\mathrm{diag}\left(\begin{pmatrix}1\\3\\5\end{pmatrix}\right)=\begin{pmatrix}1&0&0\\0&3&0\\0&0&5\end{pmatrix}$$

```
diff(Ausdruck; Variable)
diff(Ausdruck; Variable; n)
```

n-te Ableitung von Ausdruck nach Variable. Die Angabe von n ist optional und voreingestellt auf 1.

```
Dirac(x)
Dirac(x; x0)
```



Dirac-Funktion, hat am Punkt x_0 den Wert ∞, an allen anderen Stellen den Wert 1. Voreinstellung ist $x_0 = 0$.

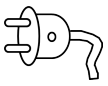
$$\mathrm{Dirac}(0)=\infty \qquad \mathrm{Dirac}(1)=0 \qquad \mathrm{Dirac}(-1)=0$$

$$\mathrm{Dirac}(2\ ;\ 2)=\infty \qquad \mathrm{Dirac}(1\ ;\ 2)=0 \qquad \mathrm{Dirac}(3\ ;\ 2)=0$$

$$\mathrm{Dirac}(-2\ ..\ 2)=\begin{pmatrix}-2 & 0\\ -1 & 0\\ 0 & \infty\\ 1 & 0\\ 2 & 0\end{pmatrix} \qquad \mathrm{Dirac}(-1\ ..\ 3\ ;\ 2)=\begin{pmatrix}-1 & 0\\ 0 & 0\\ 1 & 0\\ 2 & \infty\\ 3 & 0\end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
DocumentDirectory("")
```

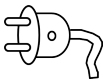


Liefert den Pfadname für das aktuell offene SMath-Dokument. Es ist zu empfehlen, bei Dateioperationen immer das aktuelle Verzeichnis auf diesen Wert zu setzen, damit für die Dateinamen portable relative Pfade benutzt werden können.

```
dir := CurrentDirectory(DocumentDirectory(""))

dir = "D:\FHB\Software\SMath\SMath Skript\Bilder\"
```

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

```
Draw2d(obj)
Draw2d(obj; name)
Draw2d(obj; size)
Draw2d(obj; name; size)
```



Erzeugt ein 2D-Diagramm anhand der Objekte und Optionen in der Liste `obj`.

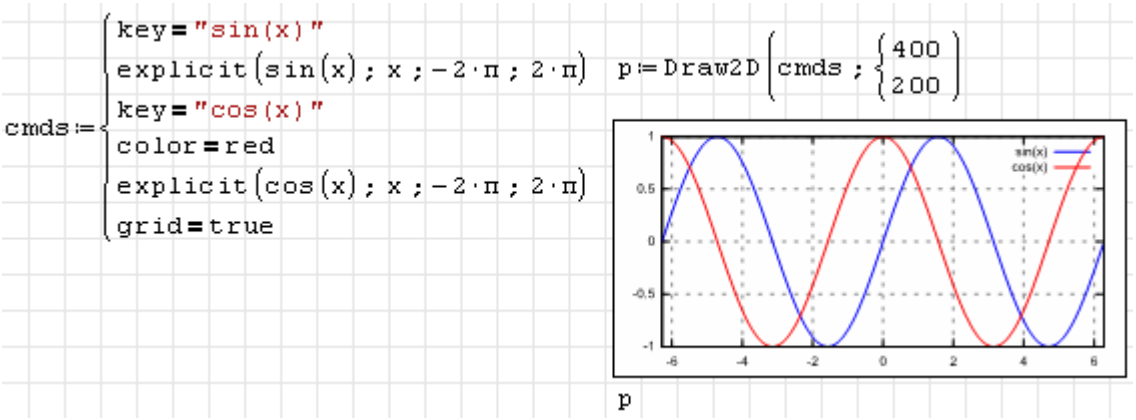
Ist der Dateiname `name` angegeben, gilt er relativ zum aktuellen Verzeichnis. Hat `name` keine Endung, wird ein skalierbares SVG-Bild erzeugt. Hat er die Endung *png*, wird ein (nichtskalierbares) PNG-Bild erzeugt. Ist kein Dateiname angegeben, wird eine Datei mit zufälligem Namen in einem temporären Verzeichnis erzeugt.

Der Rückgabewert ist der relative Dateiname (wenn name angegeben ist) oder der absolute Dateiname, wenn er automatisch erzeugt wurde. Der Rückgabewert kann benutzt werden, um das Bild mit einem Image-Bereich anzuzeigen.

Die Größe ist voreingestellt auf 300x240 Punkte und muss als Liste der Form

$$\begin{cases}\text{Breite}\\ \text{Höhe}\end{cases}'$$

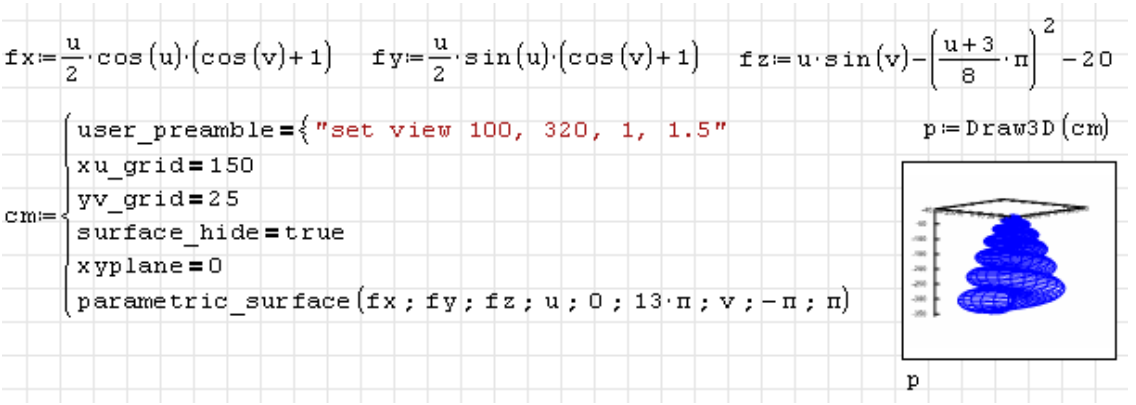
angegeben werden.



Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt 5.1).

```
Draw3d(obj)
Draw3d(obj; name)
Draw3d(obj; size)
Draw3d(obj; name; size)
```

Erzeugt ein 3D-Diagramm anhand der Objekte und Optionen in der Liste `obj` mit der voreingestellten Größe 300x300 Punkte.



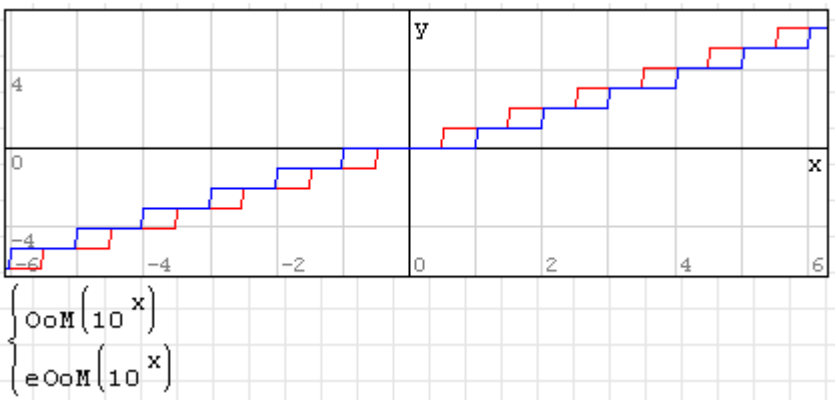
Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt 5.1).

```
el(Matrix; i)
el(Matrix; i; j)
```

Matrixelement i (Zeile), j (Spalte). Wird nur ein Index angegeben, wird zeilenweise durchgezählt. Die Indexfunktion kann auch auf der linken Seite von Zuweisungen verwendet werden. Wenn das adressierte Matrixelement nicht existiert, wird die Matrix erzeugt oder vergrößert.

```
eOoM(Variable)
```

Größenordnung der Variable (gerundeter dekadischer Logarithmus). Bei Variablen mit Einheiten bezieht sich das Ergebnis auf die Basiseinheit.



Die Funktion ist Bestandteil des Plugins *customFunction* (Seite 214).

```
eOoM.10(Variable)
```

10 hoch Größenordnung der Variable. Die Größenordnung wird als korrekt gerundeter dekadischer Logarithmus berechnet. Bei Variablen mit Einheiten bezieht sich das Ergebnis auf die Basiseinheit.

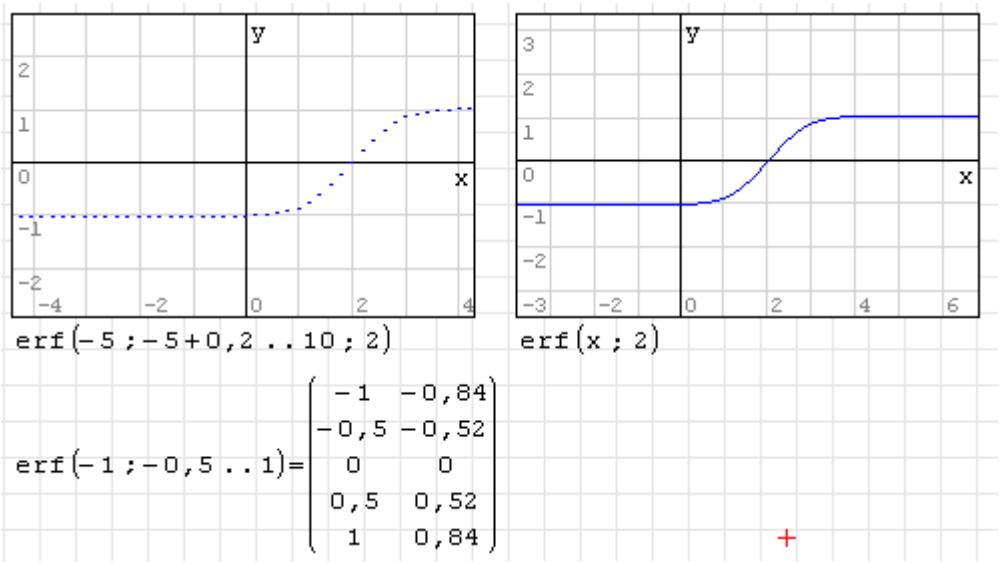
Die Funktion ist Bestandteil des Plugins *Custom Function* (Seite 214).

```
erf(x)
erf(x; x0)
```

Fehlerfunktion, numerisch berechnet.

$$\mathrm{erf}(z, z_0) = \frac{2}{\sqrt{\pi}} \int_{-z_0}^{z-z_0} e^{-t^2} \, \mathrm{d}t$$

Ist das Argument ein Vektor, wird eine plotbare zweispaltige Matrix mit Argument-Wertepaaren zurückgegebben. Ein optionales zweites Argument verschiebt den Symmetriepunkt entlang x.



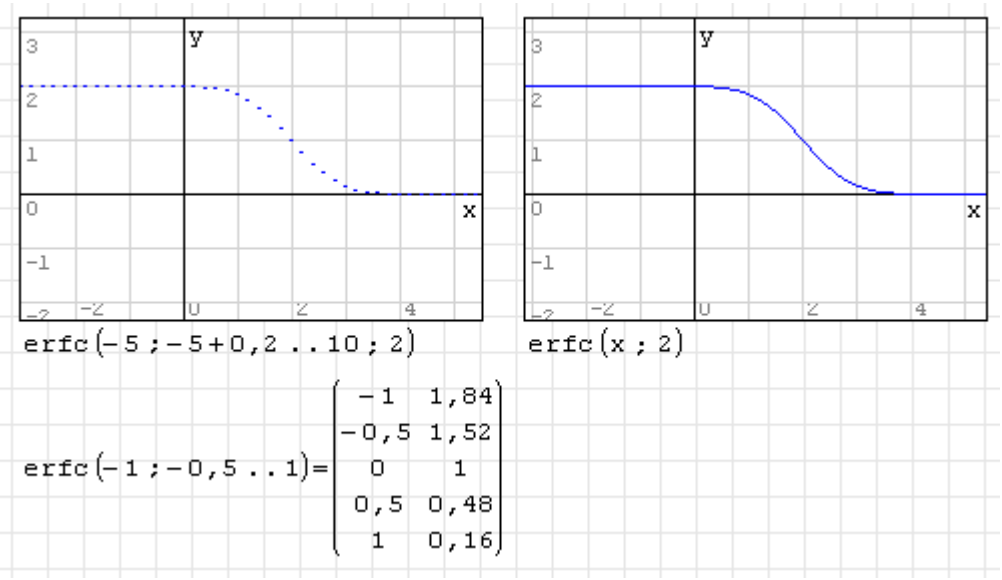
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
erfc(x)
erfc(x; x0)
```

komplementäre Fehlerfunktion, numerisch berechnet.

$$\operatorname{erfc}(z, z_0) = 1 - \operatorname{erf}(z, z_0) = 1 - \frac{2}{\sqrt{\pi}} \int_{-z_0}^{z-z_0} e^{-t^2} \, \mathrm{d}t$$

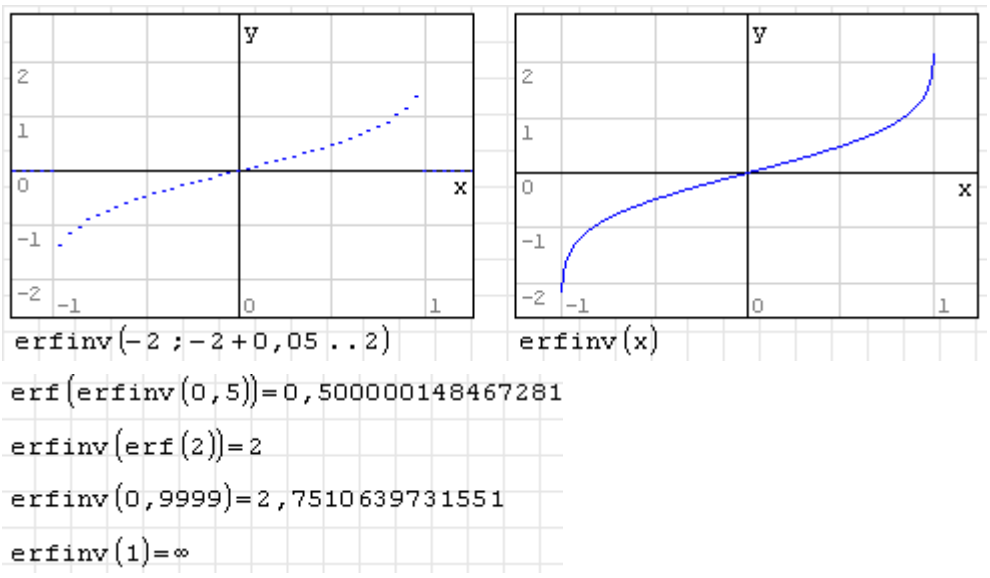
Ist das Argument ein Vektor, wird eine plotbare zweispaltige Matrix mit Argument-Wertepaaren zurückgegebben. Ein optionales zweites Argument verschiebt den Symmetriepunkt entlang x.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
erfinv(x)
```

Inverse Fehlerfunktion, durch Reihenentwicklung berechnet. Ist das Argument ein Vektor, wird eine plotbare zweispaltige Matrix mit Argument-Wertepaaren zurückgegebben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

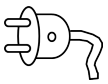
```
error(Text)
```

Erzeugt eine SMath-Fehlermeldung mit Text (Zeichenkette) als Inhalt.

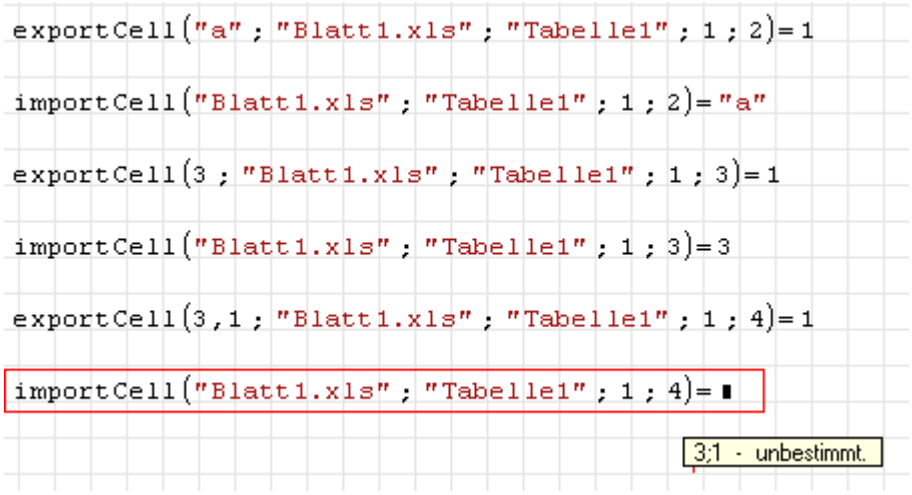
```
eval(Ausdruck)
```

Erzwingt die numerische Auswertung des Ausdrucks. Wenn dies in Funktionsdefinitionen verwendet wird, erfolgt die numerische Auswertung zum Zeitpunkt des Aufrufs. Kann zu erheblicher Rechenzeitverkürzung führen.

```
exportCell(Wert; Dateiname; Blattname; Zeile; Spalte)
```

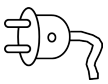


Schreibt den Wert in die angegebenen Zeile und Spalte (beides als Zahlen) des angegebenen Blatts in der angegebenen Datei (.xls). Die Datei und das Blatt (Tabelle) müssen bereits existieren. Die geschriebenen Werte können mit importCell(), siehe Seite 297 wieder eingelesen werden, das funktioniert aber nur für ganzzahlige Werte oder Zeichenketten. Fließkommazahlen können nicht korrekt eingelesen werden. Zudem sind die von SMath modifizierten Dateien nicht mit Excel unter Windows 7 lesbar.



Diese Funktion ist Bestandteil des Plugins *Excel Collaboration* (siehe Abschnitt 9.4.5).

```
exportData.CSV(Variable; Dateiname; Pfadname)
```



Schreibt die Daten aus `Variable` in eine *csv*-Datei (Kommagetrennte Spalten) mit dem Namen `Dateiname` (Endung *.csv* wird automatisch ergänzt) im Verzeichnis `Pfadname`. Liefert im Erfolgsfall den Wert 1, ansonsten den Wert 0.

Wird die Verzeichnisangabe weggelassen, wird die Datei in das gleiche Verzeichnis geschrieben, in der die exportierende .sm-Datei liegt.

$$A := \begin{pmatrix} 1 & 2{,}3\ kN & \frac{4}{5} & -57{,}1 & \left(\sin\left(\sqrt{2}\right)\ \text{"Text mit äöü"}\right) \\ \sin\left(\sqrt{2}\right) & \text{"Text φ"} & e^{50} & 3>2 & \text{"Noch "ein" Text"} \end{pmatrix}$$

$$Dir := \text{"D:\tmp"} \qquad Name := \text{"export"}$$

$$\text{exportData}_{CSV}(A\ ;\ Name) = 1$$

$$\text{exportData}_{CSV}(A\ ;\ Name\ ;\ Dir) = 1$$

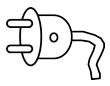
Das Dateiformat wird durch die aktuellen Einstellungen von Argumenttrennzeichen (im vorliegenden Handbuch standardmäßig das Semikolon) und Dezimaltrennzeichen (hier das Komma) bestimmt. In symbolischen Ausdrücken, die nicht als Zahl ausgewertet werden können, wird die einstellungsunabhängige interne Darstellung mit Komma und Punkt als Argument- und Dezimaltrennzeichen verwendet.

Die Datei *export.csv* hat im oben angegebenen Beispiel folgenden Inhalt:

```
1,"2300*{'kg*'m}/{'s^2}",0.8,-57.1,"mat(0.987765945992735,""Text mit äöü"",1,2)"
0.987765945992735,"Text φ",5.18470552858706E+21,1,"Noch ""ein"" Text"
```

Diese Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

`exportData.ODF(Ausdruck; Dateiname; Pfadname)`



Exportiert `Ausdruck` in eine odf-Datei (Openoffice Formeleditor) mit dem Namen `Dateiname` (Endung *.odf* wird automatisch ergänzt) im Verzeichnis `Pfadname`. Liefert im Erfolgsfall den Wert 1, ansonsten den Wert 0.

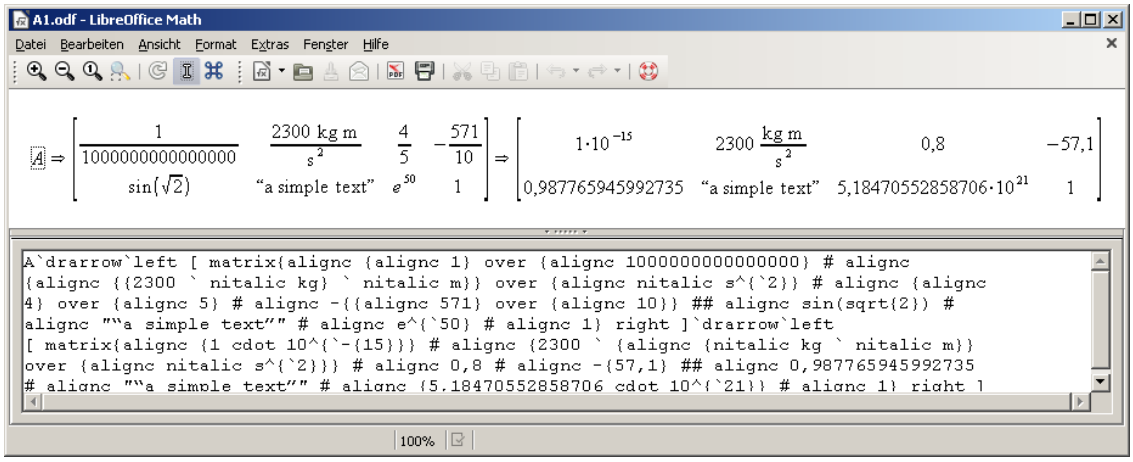
Wird die Verzeichnisangabe weggelassen, wird die Datei in das gleiche Verzeichnis geschrieben, in der die exportierende .sm-Datei liegt.

Die exportierte Formel hat die Struktur

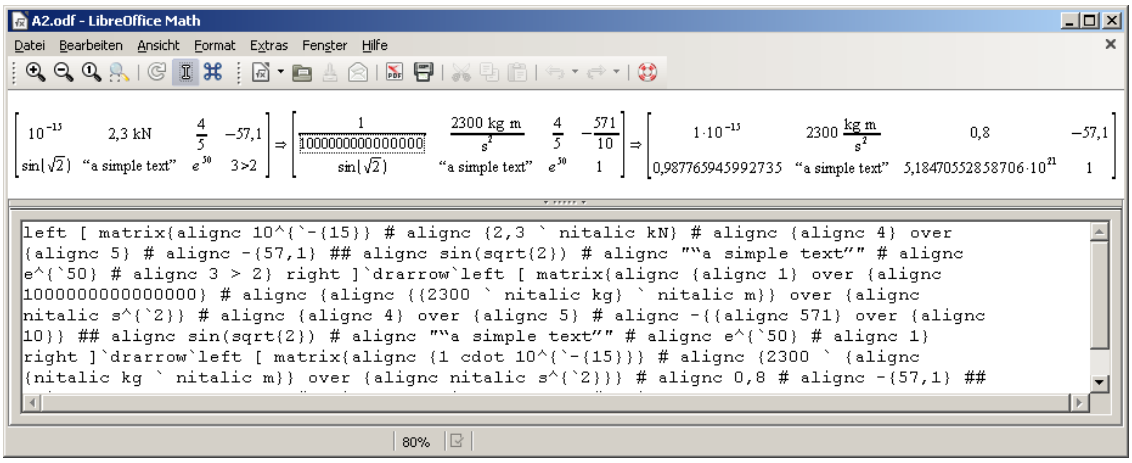
Ausdruck ⇒ symbolisch ausgewertet ⇒ numerisch ausgewertet

$$A := \begin{pmatrix} 10^{-15} & 2{,}3\ kN & \frac{4}{5} & -57{,}1 \\ \sin\left(\sqrt{2}\right) & \text{"a simple text"} & e^{50} & 3>2 \end{pmatrix}$$

$$\text{exportData}_{ODF}(A\ ;\ \text{"A1"}) = 1$$

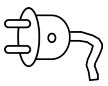


$$\text{exportData}_{\text{ODF}}\left(\begin{pmatrix} 10^{-15} & 2{,}3\ kN & \frac{4}{5} & -57{,}1 \\ \sin\left(\sqrt{2}\right) & \text{"a simple text"} & e^{50} & 3>2 \end{pmatrix}; \text{"A2"}\right)=1$$



Diese Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

```
exportData.ODS(Ausdruck; Dateiname)
exportData.ODS(Ausdruck; Dateiname; Pfadname)
```



Schreibt die Daten aus `Ausdruck` in eine ods-Datei (Openoffice Calc) mit dem Namen `Dateiname` (Endung *.ods* wird automatisch ergänzt) im Verzeichnis `Pfadname`. Liefert im Erfolgsfall den Wert 1, ansonsten den Wert 0.

Wird die Verzeichnisangabe weggelassen, wird die Datei in das gleiche Verzeichnis geschrieben, in der die exportierende .sm-Datei liegt.

$$A:=\begin{pmatrix} 1 & 2{,}3\ kN & \frac{4}{5} & -57{,}1 & \left(\sin\left(\sqrt{2}\right)\ \text{"a nested simple text"}\right) \\ \sin\left(\sqrt{2}\right) & \text{"a simple text"} & e^{50} & 3>2 & \text{"a more "complex" text"} \end{pmatrix}$$

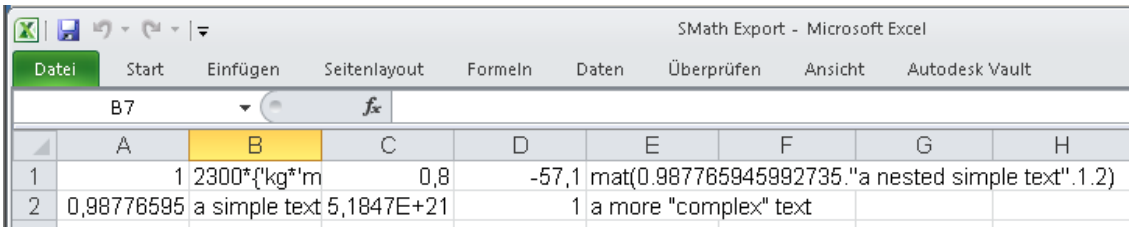
$$Dir:=\text{"D:\tmp"} \qquad Name:=\text{"SMath Export"}$$

$$\text{exportData}_{\text{ODS}}(A\ ;\ Name)=1$$

$$\text{exportData}_{\text{ODS}}(A\ ;\ Name\ ;\ Dir)=1$$

Die Einstellungen für Argumenttrennzeichen und Dezimaltrennzeichen haben keine Auswirkungen auf das Dateiformat. In symbolischen Ausdrücken, die nicht als Zahl ausgewertet werden können, wird die einstellungsunabhängige interne Darstellung mit Komma und Punkt als Argument- und Dezimaltrennzeichen verwendet. Das Ergebnis ist eine entsprechend formatierter Text.

Die so erzeugten ods-Dateien können auch mit Excel geöffnet werden (hier unter Windows 7):

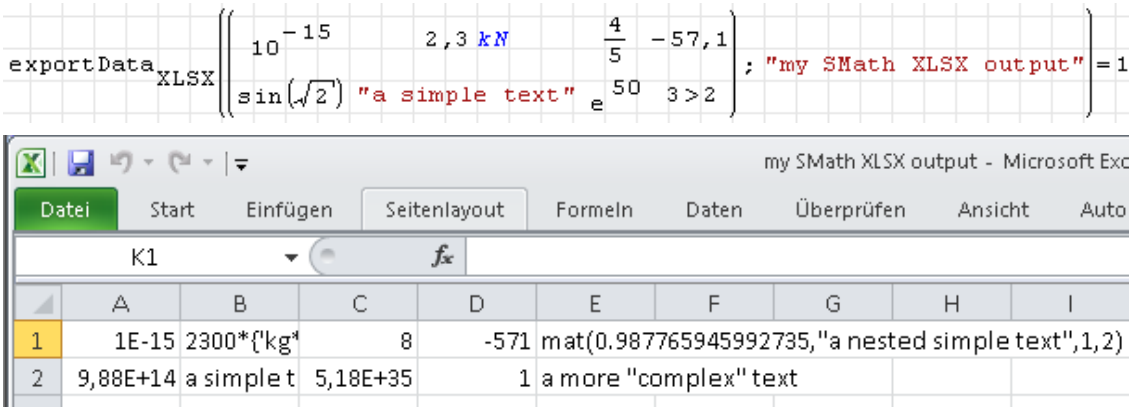


Diese Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

```
exportData.XLSX(Ausdruck; Dateiname)
exportData.XLSX(Ausdruck; Dateiname; Pfadname)
```

Schreibt die Daten aus `Ausdruck` in eine xlsx-Datei (MS Excel) mit dem Namen `Dateiname` (Endung *.xlsx* wird automatisch ergänzt) im Verzeichnis `Pfadname`. Liefert im Erfolgsfall den Wert 1, ansonsten den Wert 0.

Wird die Verzeichnisangabe weggelassen, wird die Datei in das gleiche Verzeichnis geschrieben, in der die exportierende .sm-Datei liegt.



Diese Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

```
FindRoot(f; x0)
FindRoot(f; x0; eps_f)
FindRoot(f; x0; eps_f; eps_x)
```

Numerischer Gleichungslöser

`f` Liste oder Vektor von zu lösenden Ausdrücken (Funktionen der unabhängigen Variablen). Enthalten sie kein Gleichheitszeichen, werden die Nullstellen gesucht.

`x0` Startwert oder Vektor von Startwerten für die Unbekannten. Es können Werte oder Gleichungen der Form `Unbekannte=Wert` (Boolsches Gleichheitszeichen) angegeben werden.

Werden Gleichungen angegeben, dann wird die gefundene Lösung auf die jeweilige Unbekannte zugewiesen und kann in folgenden Rechnungen direkt verwendet werden.

Die Namen der Unbekannten und die Reihenfolge, in der die Startwerte anzugeben sind, können mit der Funktion `Unknowns()` (Seite **??**) ermittelt werden.

`eps_f` Toleranz oder Vektor/Liste von Toleranzwerten für die Funktionen/Gleichungen (Voreinstellung 4)

`eps_x` Toleranz oder Vektor/Liste von Toleranzwerten für die unabhängigen Variablen

Beispiel: Berechnen Sie den Durchmesser d und die Höhe h eines Zylinders mit gegebenem Volumen V und gegebener Oberfläche A.

Zunächst werden die bekannten Werte und das Gleichungssystem definiert:

$$A := 10\,m^2 \qquad V := 1\,m^3$$

$$f_1 := \left(A = 2\cdot\left(\frac{\pi}{4}\cdot d^2\right) + \pi\cdot d\cdot h\right) \qquad f_2 := \left(V = \frac{\pi}{4}\cdot d^2\cdot h\right)$$

Unbekannte zur Kontrolle anzeigen und Lösungsfunktion aufrufen:

$$\mathrm{Unknowns}(f) = \begin{pmatrix} d \\ h \end{pmatrix}$$

$$\mathrm{FindRoot}\left(f ; \begin{pmatrix} d = 1\,m \\ h = 1\,m \end{pmatrix}\right) = \begin{pmatrix} 0{,}4109\,m \\ 7{,}5411\,m \end{pmatrix}$$

Da die Startwerte in Gleichungsform angegeben wurden, wurde die Lösung intern bereits auf die Unbekannten zugewiesen:

$$\boxed{d = 411\,mm} \qquad \boxed{h = 7541\,mm}$$

Die Funktion ist Bestandteil des Plugins *Nonlinear Solvers* (siehe Abschnitt 9.4.10).

```
findrows(Matrix; Ausdruck; i)
```

Liefert eine Matrix mit allen Zeilen aus `Matrix`, deren Wert in Spalte `i` gleich `Ausdruck` ist. Wird kein passender Eintrag gefunden, liefert die Funktion den Wert Null.

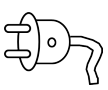
$$\mathrm{findrows}\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} ; 4 ; 2\right) = \begin{pmatrix} 3 & 4 \\ 3 & 4 \end{pmatrix} \qquad \mathrm{findrows}\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} ; 5 ; 1\right) = \begin{pmatrix} 5 & 6 \end{pmatrix}$$

$$\mathrm{findrows}\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} ; 7 ; 1\right) = 0$$

```
findstr(Zeichenkette1; Zeichenkette2)
```

liefert einen Vektor mit den Startpositionen der Zeichenkette2 in Zeichenkette1. Ist Zeichenkette2 nicht enthalten, wird -1 zurückgeliefert.

```
Fit(data; vars; eqn; pars; init)
```



Ausgleichsrechnung (kleinste Fehlerquadrate). Passt die Parameter `pars` in der Gleichung `eqn` mit den Startwerten `init` so an, dass der mittlere quadratische Abstand zwischen den durch `data` gegebenen Daten und der Gleichung minimal wird.

`data` Matrix mit den gegebenen Daten. Jede Zeile ist ein Datensatz, jede Spalte eine Variable.

`vars` Liste der Variablennamen (für jede Spalte in `data` ein Eintrag).

`eqn` Gleichung, verknüpft die in `vars` aufgelisteten Größen und die in `pars` aufgelisteten Ausgleichsparameter.

`pars` Namen der freien Parameter in der Gleichung `eqn`, gegeben als Liste.

`init` Startwerte für die Suche nach den optimalen Parametern.

Rückgabewert ist eine Liste mit Gleichungen der Form Parameter=Wert. Dieser kann genau wie bei `Solve()` mit `Assign()` für die Zuweisung der Werte an die Parameter benutzt werden.

Beispiel 1:

$$Data := \begin{pmatrix} 1 & 1{,}5 & 2{,}25 & 3 & 2 \\ 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 \end{pmatrix}^T$$

$$Fit\left(Data ; \begin{pmatrix} z \\ x \\ y \end{pmatrix} ; (z+D)^2 = (C + A\cdot x + B\cdot y) ; \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} ; \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} A = -3{,}628998943824597 \\ B = -1{,}652625813237137 \\ C = 10{,}45496853509099 \\ D = -3{,}319624288317973 \end{pmatrix}$$

Beispiel 2:

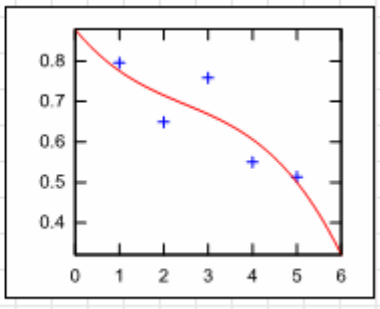
$$D := augment(1..5 ; Random(5))$$

$$f(x) := a\cdot x^3 + b\cdot x^2 + c\cdot x + d$$

$$D = \begin{pmatrix} 1 & 0{,}796 \\ 2 & 0{,}6491 \\ 3 & 0{,}7588 \\ 4 & 0{,}5501 \\ 5 & 0{,}5136 \end{pmatrix}$$

$$Assign\left(Fit\left(D ; \begin{pmatrix} x \\ y \end{pmatrix} ; y = f(x) ; \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} ; \begin{pmatrix} 1, \\ 3, \\ 1, \\ 1, \end{pmatrix}\right)\right) = \begin{pmatrix} -0{,}0048 \\ 0{,}0356 \\ -0{,}1337 \\ 0{,}879 \end{pmatrix}$$

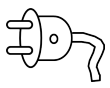
$$p := Draw2D\left(\begin{pmatrix} points(D) \\ color = red \\ explicit(f(x) ; x ; 0 ; 6) \end{pmatrix} ; \begin{pmatrix} 200 \\ 160 \end{pmatrix}\right)$$



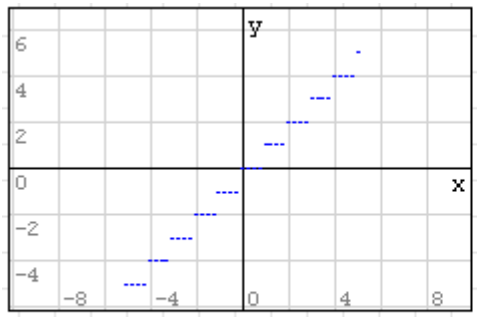
p

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

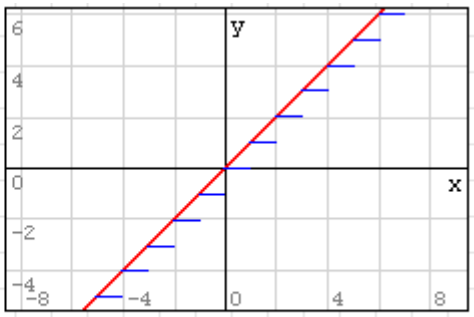
`Floor(x)`



Liefert die nächste ganze Zahl, die kleiner oder gleich x ist. Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und Floor(x) erzeugt.



Floor(−5 ; −5+0,25 .. 5)



Floor(x)
x

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
for(Variable; Wertebereich; Ausdruck)
```

Mengenschleife. Der Ausdruck wird wiederholt ausgeführt, wobei Variable nimmt nacheinander alle Werte aus Wertebereich annimmt. Die Schleife liefert keinen Rückgabewert.

$$s := 0 ; \frac{\pi}{4} .. 2 \cdot \pi \qquad \text{for } i \in 1 .. \text{length}(s)$$
$$y_i := \sin(s_i)$$

$$s = \begin{pmatrix} 0 \\ 0,79 \\ 1,57 \\ 2,36 \\ 3,14 \\ 3,93 \\ 4,71 \\ 5,5 \\ 6,28 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 0,7071 \\ 1 \\ 0,7071 \\ 0 \\ -0,7071 \\ -1 \\ -0,7071 \\ -2,4492 \cdot 10^{-16} \end{pmatrix}$$

```
for(Startanweisung; Bedingung; Fortsetzungsanweisung; Ausdruck)
```

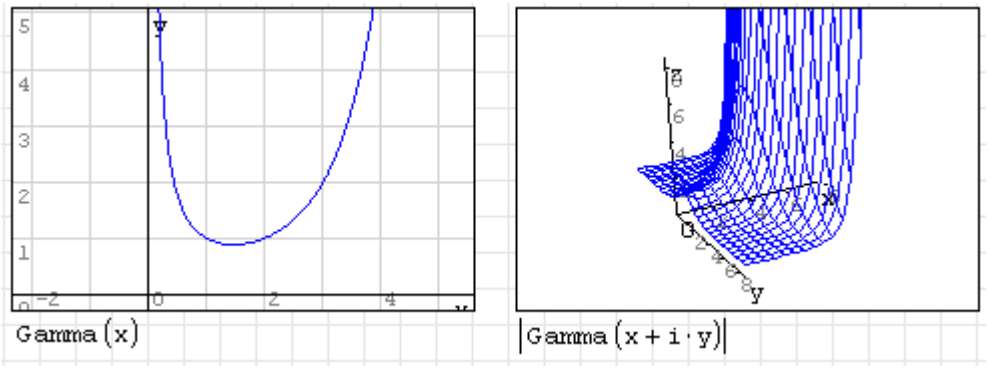
Kopfgesteuerte Schleife. Zunächst wird die Startanweisung ausgeführt. Wenn die Bedingung wahr ist, wird der Ausdruck und dann die Fortsetzungsanweisung ausgeführt. Dies wird wiederholt, solange die Bedingung wahr ist.

```
Gamma(x)
```

Liefert die Gamma-Funktion einer komplexen Zahl, allerdings nur für x mit nichtnegativem Realteil.

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt.$$

Sie interpoliert die für natürliche Zahlen definierte Fakultätsfunktion.



`GammaRegularized.P(a,x)`

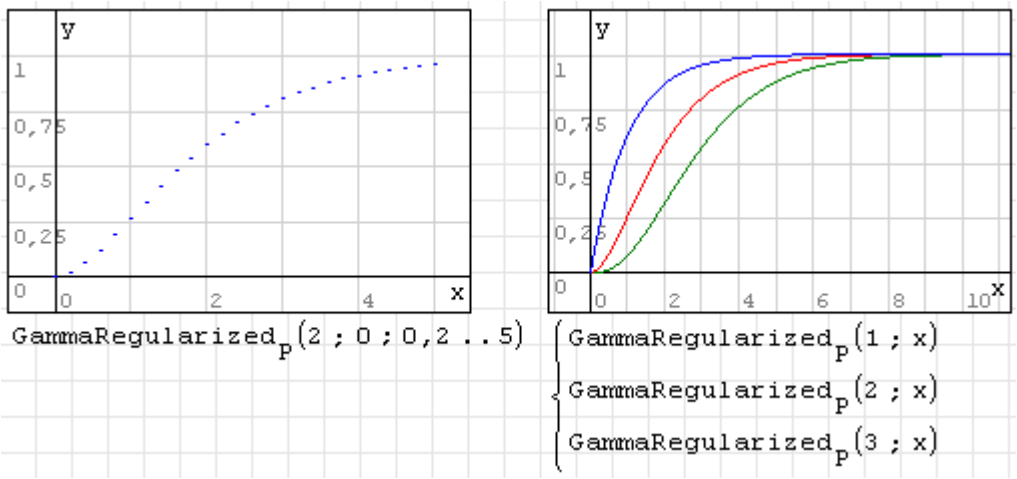
Liefert die regularisierte Gamma-Funktion der oberen Grenze

$$P(a,x) = \frac{\gamma(a,x)}{\Gamma(a)}$$

mit der unvollständigen Gammafunktion der oberen Grenze

$$\gamma(a,x) = \int_0^x t^{a-1} e^{-t} \mathrm{d}t$$

Ist das Argument x ein Vektor, wird eine plotbare zweispaltige Matrix mit Wertepaaren $x, P(a,x)$ zurückgegebben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`GammaRegularized.Q(a,x)`

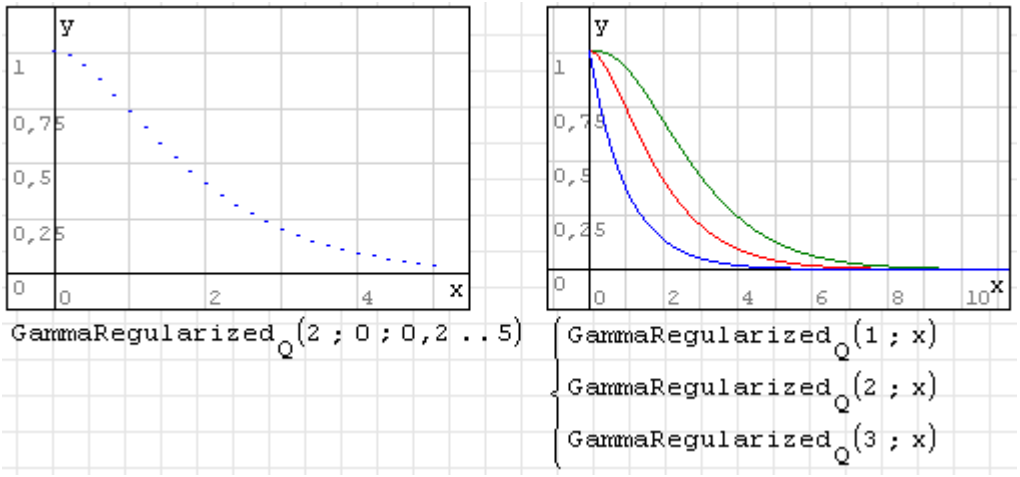
Liefert die regularisierte Gamma-Funktion der unteren Grenze

$$Q(a,x) = \frac{\Gamma(a,x)}{\Gamma(a)}$$

mit der unvollständigen Gammafunktion der unteren Grenze

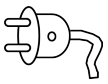
$$\Gamma(a,x) = \int_x^\infty t^{a-1} e^{-t} \mathrm{d}t$$

Ist das Argument x ein Vektor, wird eine plotbare zweispaltige Matrix mit Wertepaaren $x, Q(a,x)$ zurückgegebben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

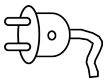
```
GaussNewton(Funktionen; Startwerte; Genauigkeit; maxIter)
GaussNewton(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Gauss-Newton-Optimierer. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

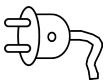
```
GaussNewton.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter)
GaussNewton.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter; Schalter)
```



Gauss-Newton-Optimierer mit numerischer Ableitungsberechnung mittels zentralem Differenzenquotient. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotienten. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
GaussNewton.CDGSS(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter)
GaussNewton.CDGSS(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter; Schalter)
```

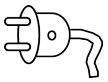


Gauss-Newton-Optimierer mit numerischer Ableitungsberechnung mittels zentralem Differenzenquotient und Line Search nach dem Goldenen Schnitt-Suchverfahren. Funktionen:

Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotienten. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

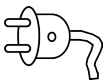
```
GaussNewton.GSS(Funktionen; Startwerte; Genauigkeit; maxIter)
GaussNewton.GSS(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Gauss-Newton-Optimierer mit Line Search nach dem Goldenen Schnitt-Suchverfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
GeometricMean(Daten)
```



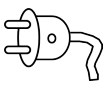
Liefert den geometrischen Mittelwert der Daten (Matrix, Vektor oder Liste):

$$m = \left(\prod_{i=1}^{n} x_i \right)^{\frac{1}{n}}$$

$$\text{GeometricMean}\left(\begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix}\right) = 2{,}5198 \qquad \text{GeometricMean}\left(\begin{pmatrix} 1 & 3 & 5 & 1 & 6 & 2 \\ 3 & 2 & 5 & 2 & 3 & 5 \\ 5 & 2 & 1 & 3 & 4 & 1 \end{pmatrix}\right) = 2{,}5405$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
GetFolderPath(Kennzahl)
```



Liefert den zu Kennzahl gehörenden Systempfad unter Windows zurück. Bei ungültiger Kennzahl erfolgt eine Fehlermeldung. Die hier angegebenen Kennzahlen wurden durch Ausprobieren ermittelt.

```
PROGRAMS := 2    PERSONAL := 5    FAVORITES := 6    STARTUP := 7    RECENT := 8    SENDTO := 9
STARTMENU := 11    MUSIC := 13    DESKTOP := 16    TEMPLATES := 21    APPDATA := 26
LOCALAPPDATA := 28    INTERNETCASH := 32    COOKIES := 33    HISTORY := 34
COMMONAPPDATA := 35    SYSTEM := 37    PROGRAMFILES := 38    PICTURES := 39
COMMONPROGRAMFILES := 43
```

Hier einige Beispiele:

GetFolderPath(PERSONAL)="C:\Users\Kraska\Documents"

GetFolderPath(FAVORITES)="C:\Users\Kraska\Favorites"

GetFolderPath(COOKIES)="C:\Users\Kraska\AppData\Roaming\Microsoft\Windows\Cookies"

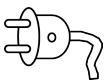
GetFolderPath(HISTORY)="C:\Users\Kraska\AppData\Local\Microsoft\Windows\History"

GetFolderPath(COMMONAPPDATA)="C:\ProgramData"

GetFolderPath(SYSTEM)="C:\Windows\system32"

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.3).

```
GetType(expr)
```



Liefert Typ des Ausdrucks als Zeichenkette.

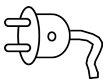
$$\text{GetType}\left(\begin{pmatrix}1\\2\end{pmatrix}\right)=\text{"system"} \qquad \text{GetType}\left(\begin{pmatrix}1&2\\3&5\end{pmatrix}\right)=\text{"matrix"}$$

$$\text{GetType}(a)=\text{"unknown"} \qquad \text{GetType}(1)=\text{"number"}$$

$$\text{GetType}(\text{"a"})=\text{"string"}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

```
GETWAVINFO(Dateiname)
```



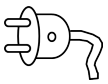
Lesen von .wav-Dateiinformationen. Es wird ein Vektor mit vier Elementen zurückgegeben:

WavInfo:=GETWAVINFO("test.wav")

$$\text{WavInfo}=\begin{pmatrix}1\\8000\\8\\8000\end{pmatrix}\quad\begin{matrix}\text{Kanäle}\\\text{Abtastrate (Werte pro Sekunde)}\\\text{Auflösung (Bit)}\\\text{Bytes pro Sekunde (im Mittel)}\end{matrix}$$

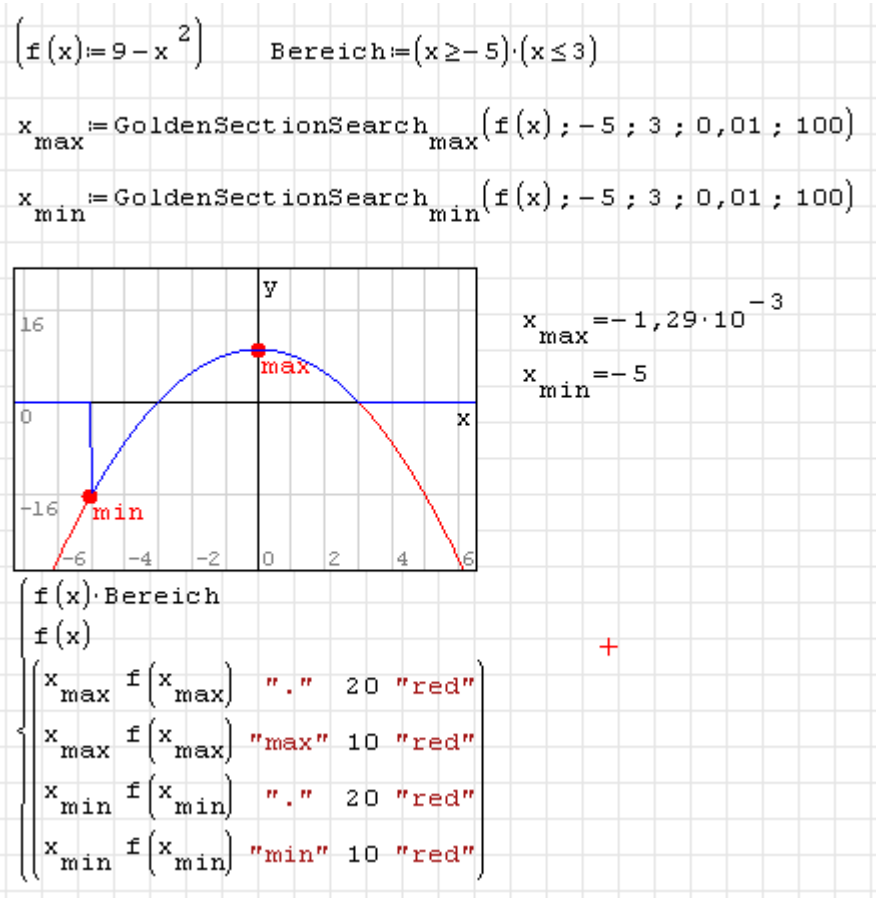
Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

```
GoldenSectionSearch.max(Funktionen; Untergrenzen; Obergrenzen;
Genauigkeiten, maxIter)
GoldenSectionSearch.max(Funktionen; Untergrenzen; Obergrenzen;
Genauigkeiten, maxIter; Schalter)
```

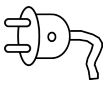


Optimierer nach dem Goldenen-Schnitt-Suchverfahren, maximiert die Zielfunktionen. Funktionen: Liste der Zielfunktionen, Untergrenze/Obergrenze: Liste der unteren/oberen Grenzwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).



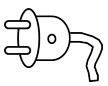

```
GoldenSectionSearch.min(Funktionen; Untergrenzen; Obergrenzen;
Genauigkeiten, maxIter)
GoldenSectionSearch.min(Funktionen; Untergrenzen; Obergrenzen;
Genauigkeiten, maxIter; Schalter)
```



Optimierer nach dem Goldenen-Schnitt-Suchverfahren, minimiert die Zielfunktionen. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
gpc_add_contour(Polygon; Randkurve; Lochschalter)
```



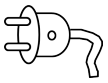
Fügt eine Randkurve zu einem Polygon hinzu. Wenn `Lochschalter=1` ist, dann definiert die Randkurve ein Loch. Die Randkurve ist als zweispaltige Matrix mit den Werten

$$\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$$

gegeben. Sind der erste und der letzte Punkt nicht identisch, wird die Randkurve automatisch geschlossen.

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

`gpc_clip(Nummer; Polygon1; Polygon2)`

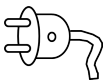


Führt eine Operation mit `Polygon1` und `Polygon2` aus. Folgende Operationen werden unterstützt:

Nummer	Operation
0	Differenz
1	Schnittmenge
2	Exklusives Oder
3	Vereinigung

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

`gpc_get_contour(Gebiet; Randnummer; Format)`

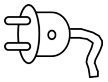


Liefert die Randkurve mit der Nummer `Randnummer` aus dem Gebiet `Gebiet`. `Format` = 0 liefert jeden Randpunkt einmal, `Format` = 1 wiederholt den ersten Punkt am Ende nochmal. Damit kann dann direkt eine geschlossene Randkurve geplottet werden. Das Ergebnis ist eine zweispaltige Matrix:

$$\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$$

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

`gpc_polygon(Randpunkte)`



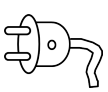
Erzeugt ein Polygon aus einer Matrix von Randpunkten im Format

$$\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}.$$

Die Punkte können, müssen aber keinen geschlossenen Rand bilden. Das Ergebnispolygon ist in jedem Fall geschlossen.

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

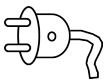
`gpc_polygon_to_tristrip(Gebiet)`



Erzeugt eine Dreieckskettendarstellung.

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

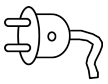
```
gpc_read_polygon(Datei; Lochschalter)
```



Liest ein Polygon aus einer Datei.

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

```
gpc_tristrip_clip(Nummer; Polygon1; Polygon2)
```

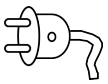


Führt eine Operation mit `Polygon1` und `Polygon2` aus und liefert das Ergebnis als Dreieckskette. Folgende Operationen werden unterstützt:

Nummer	Operation
0	Differenz
1	Schnittmenge
2	Exklusives Oder
3	Vereinigung

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

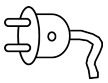
```
gpc_write_polygon(Polygon; Datei; Lochschalter)
```



Schreibt das Polygon `Polygon` in eine Datei namens `Datei` und setzt den Lochschalter auf den Wert `Lochschalter`

Die Funktion ist Bestandteil des Plugins *GPCPlugin* (siehe Abschnitt 9.4.6).

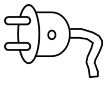
```
Gradient(Funktionen, Variable)
```



Funktionen: Liste von m Funktionen, Variable: Liste von n Variablen. Berechnet den Gradient der Funktionen in Richtung der Variablen. Das Ergebnis ist eine Liste von n Listen mit partiellen Ableitungen der m Funktionen.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

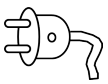
```
GradientAscent(Funktionen; Startwerte; Genauigkeit; maxIter)
GradientAscent(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Optimierer nach dem Verfahren des steilsten Anstiegs. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

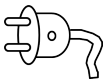
```
GradientAscent.GSS(Funktionen; Startwerte; Genauigkeit; maxIter)
GradientAscent.GSS(Funktionen; Startwerte; Genauigkeit; maxIter;
Schalter)
```



Optimierer nach dem Verfahren des steilsten Anstiegs mit Line Search nach dem Goldenen-Schnitt-Suchverfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

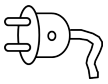
```
GradientDescent(Funktionen; Startwerte; Genauigkeit; maxIter)
GradientDescent(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Optimierer nach dem Verfahren des steilsten Abstiegs. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

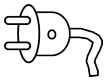
```
GradientDescent.GSS(Funktionen; Startwerte; Genauigkeit; maxIter)
GradientDescent.GSS(Funktionen; Startwerte; Genauigkeit; maxIter;
Schalter)
```



Optimierer nach dem Verfahren des steilsten Abstiegs mit Line Search nach dem Goldenen-Schnitt-Suchverfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
HarmonicMean(Daten)
```



Liefert den harmonischen Mittelwert der Daten (Matrix, Vektor oder Liste). Der harmonische Mittelwert ist definiert als

$$H = n\left(\sum_{i=1}^{n}\frac{1}{x_i}\right)^{-1}$$

$$\text{HarmonicMean}\begin{pmatrix}8\\2\\1\end{pmatrix}=1{,}8462 \qquad \text{HarmonicMean}\begin{pmatrix}1&3&5&1&6&2\\3&2&5&2&3&5\\5&2&1&3&4&1\end{pmatrix}=2{,}1053$$

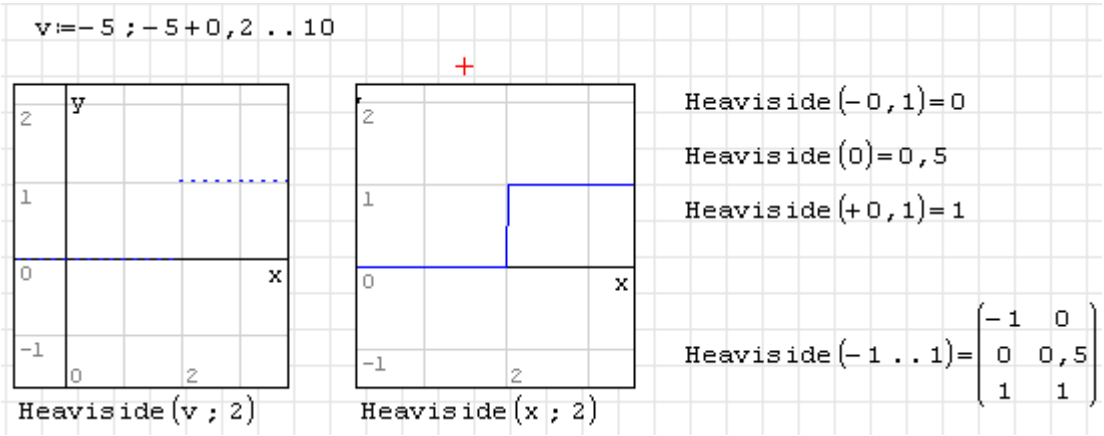
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Heaviside(x)
Heaviside(x; x0)
```

Sprungfunktion, wechselt bei $x_0$von 0 auf 1. Voreinstellung ist $x_0 = 0$. Es gilt

$$H(x - x_0) = \frac{1}{2}\left(1 + \text{sign}(x - x_0)\right)$$

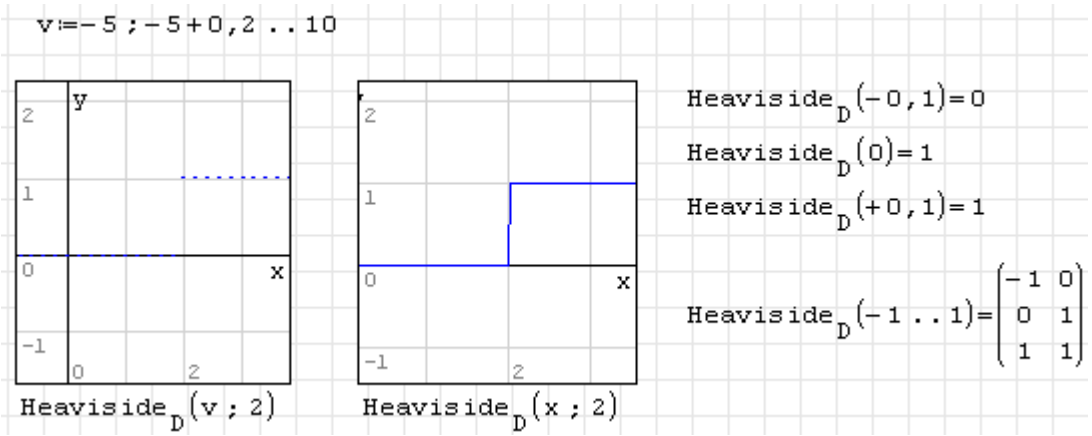
Bei dieser Version gilt $H(x = x_0) = 1/2$. Wird für xBilder/ktor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $x, H(x)$ zurückgegeben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Heaviside.D(x)
Heaviside.D(x; x0)
```

Sprungfunktion, wechselt bei $x_0$von 0 auf 1. Voreinstellung ist $x_0 = 1$. Es gilt $H(x = x_0) = 0$. Wird für x ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $x, H(x)$ zurückgegeben.



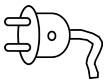
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Hessian(Funktionen; Variable)
```

Hesse-Matrix der zweiten Ableitungen der als Liste gegebenen Funktionen nach den als Liste gegebenen Variablen.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

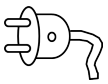
```
Hessian.CD(Funktionen; Variable; Schrittweite)
```



Hesse-Matrix der zweiten Ableitungen der als Liste gegebenen Funktionen nach den als Liste gegebenen Variablen. Die Berechnung erfolgt mittels zentralem Differenzenquotienten mit der gegebenen Schrittweite.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

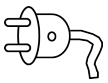
```
HRE.B(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter)
HRE.B(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter;
Schalter)
```



Nichtlinearer Löser nach dem Homotopy Root Estimation Verfahren. Benutzt intern das Broyden-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen in den gegebenen Grenzen. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Homotopie-Transformationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Transformationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

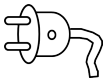
```
HRE.NR(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter)
HRE.NR(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter;
Schalter)
```



Nichtlinearer Löser nach dem Homotopy Root Estimation Verfahren. Benutzt intern das Newton-Raphson-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen in den gegebenen Grenzen. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Homotopie-Transformationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Transformationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

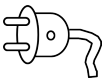
```
HRE.RK(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter)
HRE.RK(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten, maxIter;
Schalter)
```



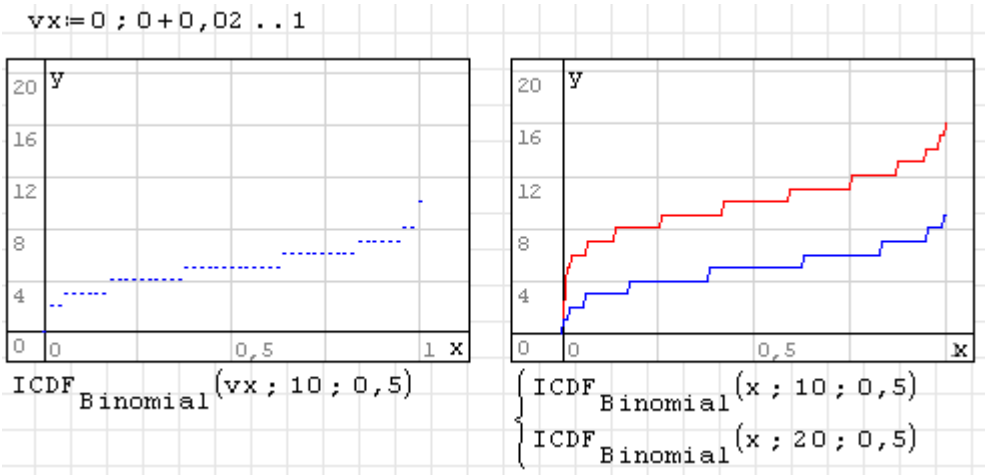
Nichtlinearer Löser nach dem Homotopy Root Estimation Verfahren. Benutzt intern das Runge-Kutta-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen in den gegebenen Grenzen. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Homotopie-Transformationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Transformationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
ICDF.Binomial(p; n; p0)
```

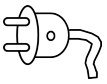


Quantilfunktion $Q(p, n, p_0)$ der diskreten Binomialverteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, n ist die Zahl der Versuche, $0 \le p_0 \le 1$ ist die Erfolgswahrscheinlichkeit eines Versuchs. Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

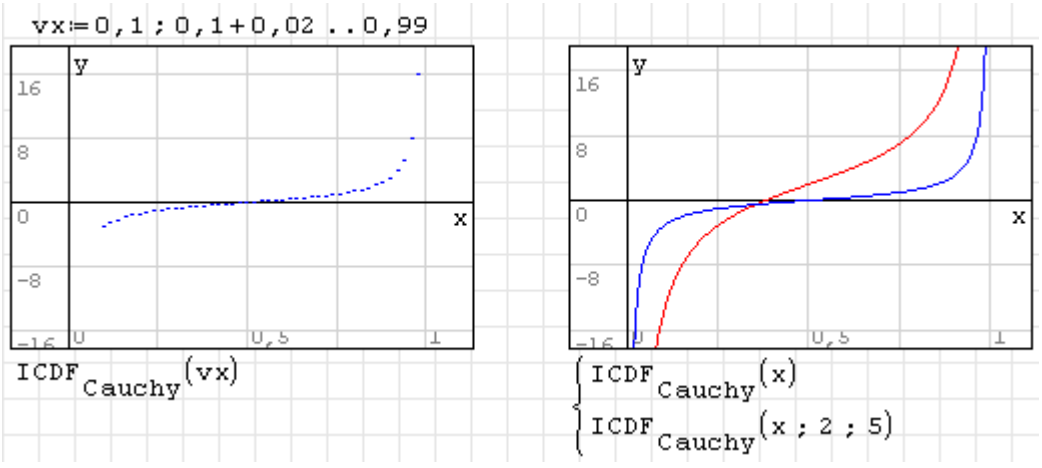


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Cauchy(p)
ICDF.Cauchy(p; t; s)
```

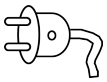


Quantilfunktion $Q(p, t, s)$ der Cauchy-Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, t ist der Ortsparameter (Standardwert 0), s ist der Breitenparameter der Verteilung (Standardwert 1). Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

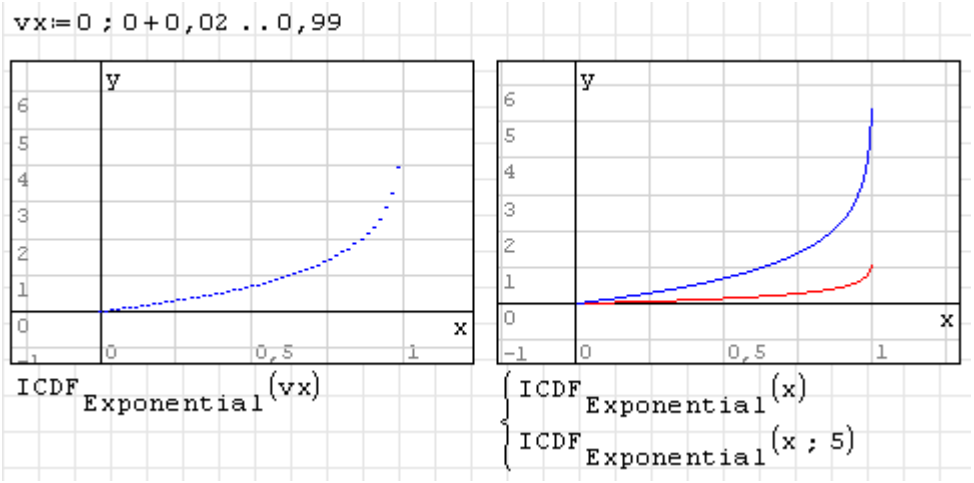


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Exponential(p)
ICDF.Exponential(p; Ereignisrate)
```

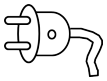


Quantilfunktion $Q(p,\lambda)$ der Exponential-Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, λ ist die Ereignisrate (Standardwert 1). Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

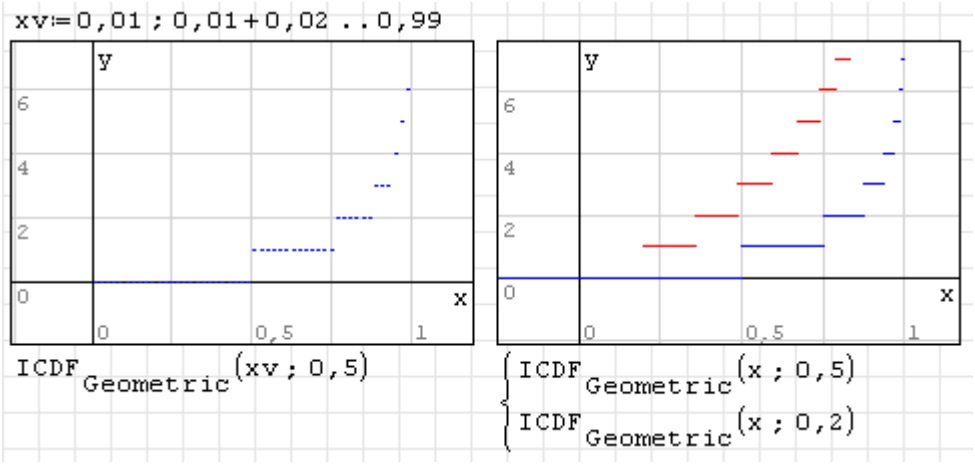


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Geometric(p; p0)
```

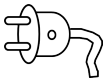


Quantilfunktion $Q(p,p_0)$ der Geometrischen Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, p_0 ist die Erfolgswahrscheinlichkeit eines Einzelversuchs. Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

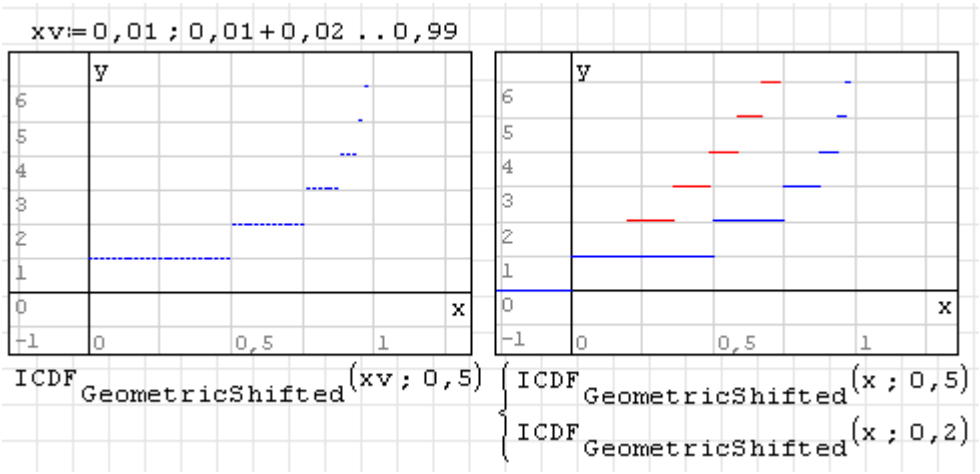


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.GeometricShifted(p; p0)
```



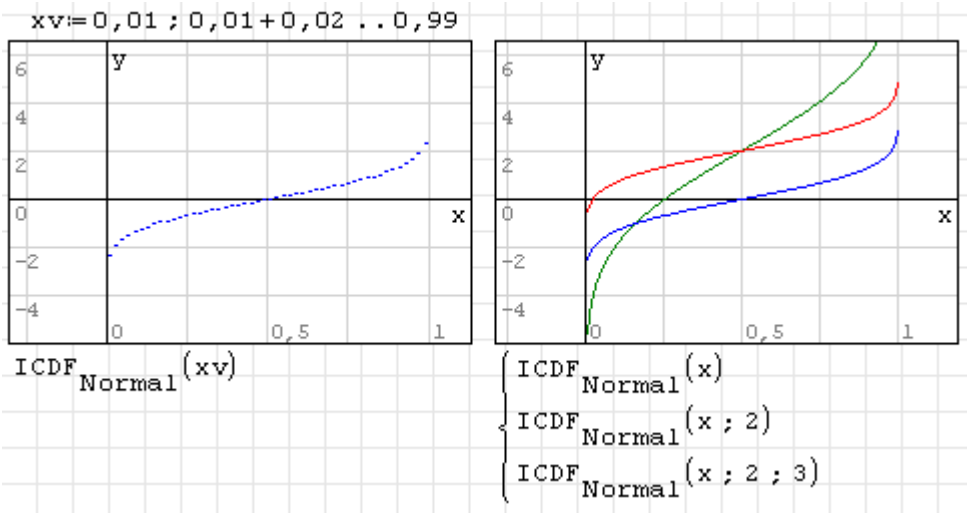
Quantilfunktion $Q(p,p_0)$ der verschobenen Geometrischen Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, p_0 ist die Erfolgswahrscheinlichkeit eines Einzelversuchs. Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Normal(p)
ICDF.Normal(p; Mittelwert)
ICDF.Normal(p; Mittelwert; Standardabweichung)
```

Quantilfunktion $Q(p,\mu,\sigma)$ der Normalverteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, μ ist der Mittelwert (Voreinstellung 0), σ ist die Standardabweichung (Voreinstellung 1). Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

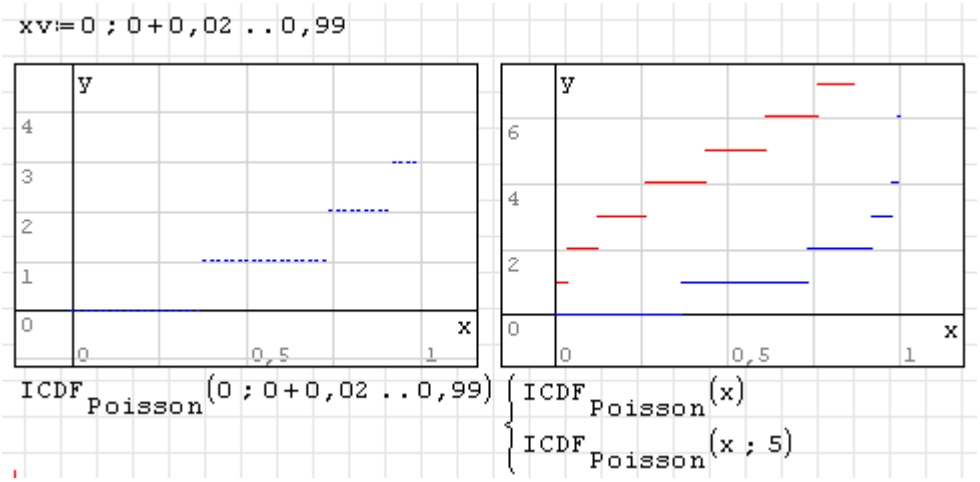


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Poisson(p)
ICDF.Poisson(p; lambda)
```

Quantilfunktion $Q(p,\lambda)$ der (diskreten) Poisson-Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, λ ist die mittlere Ereignisrate (Voreinstellung 1).

Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

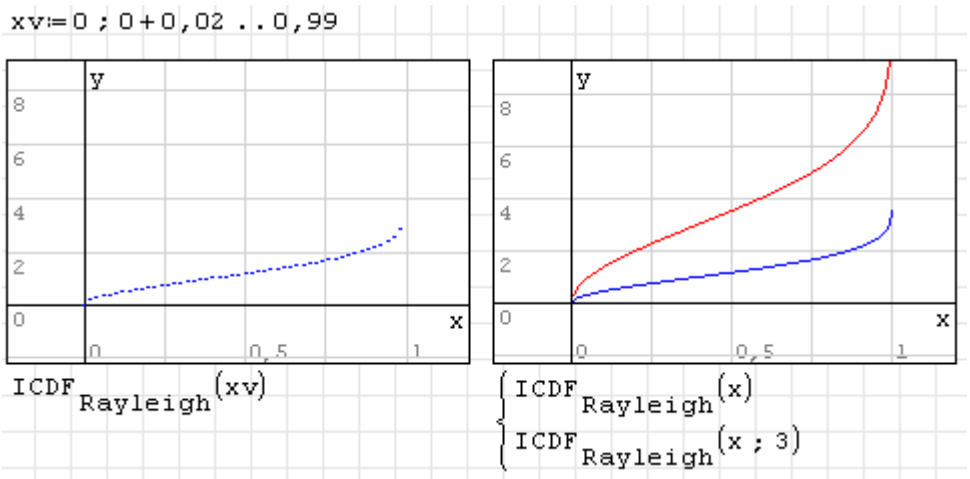


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Rayleigh(p)
ICDF.Rayleigh(p; sigma)
```

Quantilfunktion $Q(p,\sigma)$ der Rayleigh-Verteilung. $0 \le p \le 1$ ist der Unterschreitungsanteil, σ ist die Standardabweichung (Voreinstellung 1).

Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

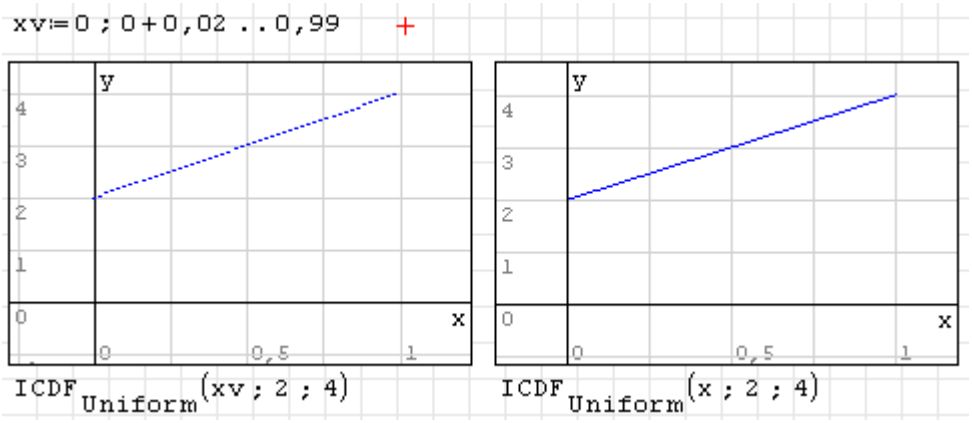


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Uniform(p; a; b)
```

Quantilfunktion $Q(p,a,b)$ der Gleichverteilung im Intervall $[a,b]$. $0 \le p \le 1$ ist der Unterschreitungsanteil.

Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

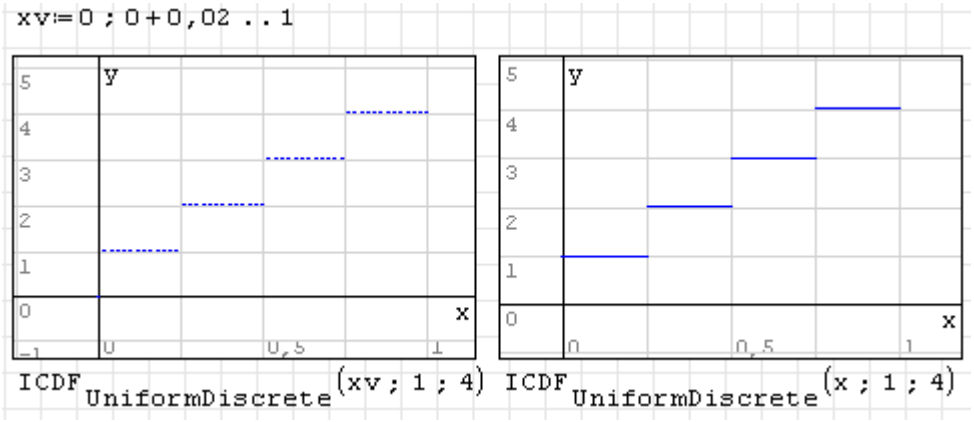


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.UniformDiscrete(p; a; b)
```

Quantilfunktion $Q(p,a,b)$ der diskreten Gleichverteilung im Intervall $[a,b]$. $0 \le p \le 1$ ist der Unterschreitungsanteil.

Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.

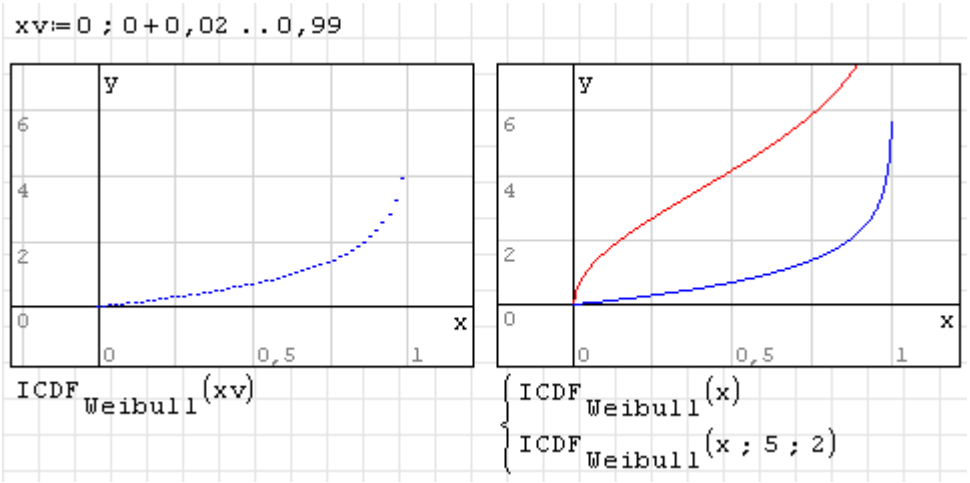


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
ICDF.Weibull(p)
ICDF.Weibull(p; T; k)
```

Quantilfunktion $Q(p,T,k)$ der Weibullverteilung mit der charakteristischen Lebensdauer T (Voreinstellung $T = 1$) und dem Streuungsparameter k (Voreinstellung $k = 1$). $0 \le p \le 1$ ist der Unterschreitungsanteil.

Wird für p ein Vektor angegeben, so wird eine plotbare zweispaltige Matrix mit Wertepaaren $p, Q(p)$ zurückgegeben.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`identity(dim)`

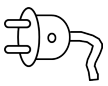
Liefert eine Einheitsmatrix der Dimension `dim`

$$\text{identity}(2)=\begin{pmatrix}1&0\\0&1\end{pmatrix} \qquad \text{identity}(3)=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$$

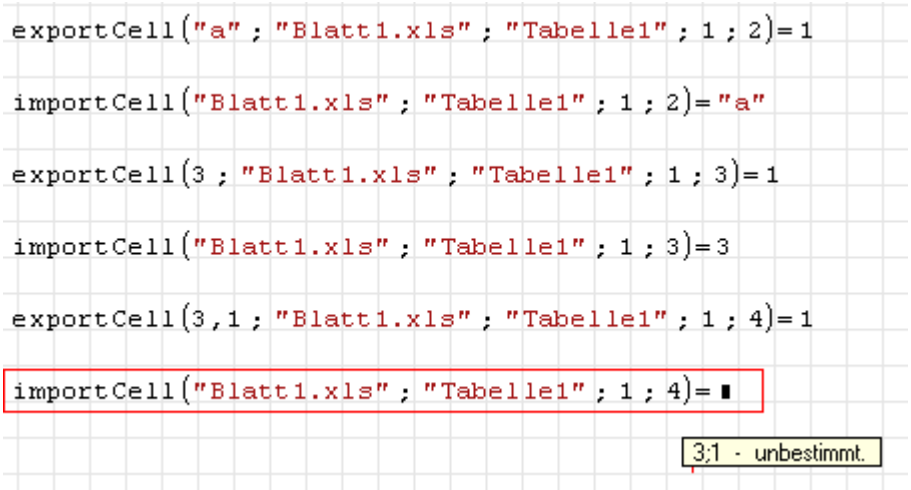
`if(Bedingung; Wahr-Ausdruck; Falsch-Ausdruck)`

Bedingte Anweisung. Ist Bedingung wahr, dann wird Wahr-Ausdruck ausgewertet, ansonsten Falsch-Ausdruck. Der Falsch-Ausdruck kann nicht weggelassen werden.

`importCell(Datei,Blatt,Zeile,Spalte)`



Liest den Wert der Zelle mit der Adresse `Zeile`, `Spalte` aus der Tabelle `Blatt` in der Excel-(.xls) Datei `Datei`. Das funktioniert leider nur für Integerwerte oder Zeichenketten. Fließkommazahlen werden nicht richtig gelesen.



Die Funktion ist Bestandteil des Plugins *Excel Collaboration* (siehe Abschnitt 9.4.5).

```
importData(Name)
importData(Name; Dezimaltrenner; Argumenttrenner; Spaltentrenner;
Startzeile; Endzeile; Startspalte; Endspalte; Schalter)
```

Liest Daten aus Textdateien ein. Siehe Hinweise zum aktuellen Verzeichnis in Abschnitt 3.17. Argumente, die nicht von der Voreinstellung abweichen sollen, können Null gesetzt werden.

Name: Dateiname, relativ zum aktuellen Verzeichnis oder mit vollständigem Pfad.

Dezimaltrenner: Je nach Dateiformat "." oder ","

Argumenttrenner: Je nach Dateiformat "," oder ";" nur bei symbolischer Auswertung relevant.

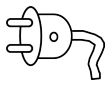
Spaltentrenner: Je nach Dateiformat. Mehrere Leerzeichen " " in der Datei hintereinander werden als ein Spaltentrenner aufgefasst.

Start-und End-Indizes: 0 bedeutet Standardwert (von der ersten Zeile/Spalte bis zur letzten Zeile/Spalte)

Schalter: 0: numerische Auswertung, 1: symbolische Auswertung

Hinweis: Die Funktion kann im numerischen Modus (Schalter=1) nur Zahlen einlesen, keine Zeichenketten. Im symbolischen Modus werden die Ausdrücke symbolisch ausgewertet.

```
importData.ODS(Datei; Blatt)
importData.ODS(Datei; Blatt; Zelle)
importData.ODS(Datei; Blatt; ZelleLO; ZelleRU)
```

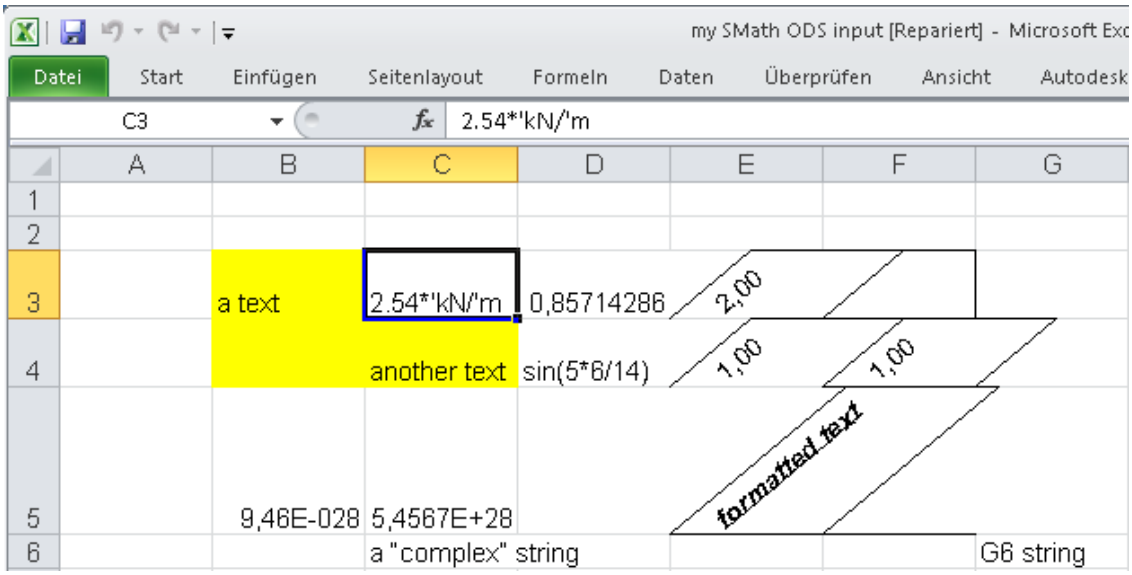


Liest Daten aus ods-Dateien. Anzugeben sind der Dateiname (`Datei`), der Blattname (`Blatt`), die linke obere und die rechte untere Zelladresse des auszulesenden Bereichs (`ZelleLO` und `ZelleRU`). Wenn nur eine Zelle auszulesen ist, reicht die Angabe einer Adresse (`Zelle`). In jedem Fall wird das Ergebnis als Matrix erzeugt, gegebenenfalls mit der Größe 1×1. Wird keine Zelle angegeben, dann wird das ganze Blatt gelesen, die Größe der Ergebnismatrix ist dann von der Datenmenge abhängig.

Die Adressen sind als Zeichenkette aus Spalten-Kennbuchstabe und Zeilennummer anzugeben.

Der Dateiname kann einen absoluten oder relativen Pfad enthalten, letzterer bezieht sich auf das aktuelle Verzeichnis (siehe dazu Abschnitt 3.17).

Die Beispieldatei *my SMath ODS input.ods* des Plugins *Data Exchange* wird von Excel nur unter Protest eingelesen:



Lesen einer einzelnen Zelle

```
A := importData_ODS("my SMath ODS input.ods", "Sheet2", "G6")
A = ("G6 string")
```

Lesen eines Zellbereichs

```
B := importData_ODS("my SMath ODS input.ods", "Sheet2", "B2", "G7")
```

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \text{"a text"} & 2.54 \cdot 10^{3}\ \frac{kg}{s^2} & 0.8571 & 2 & 0 & 0 \\ 0 & \text{"another text"} & 0.8408 & 1 & 1 & 0 \\ 9.4567 \cdot 10^{-28} & 5.4567 \cdot 10^{28} & 0 & \text{"formatted text"} & 0 & 0 \\ 0 & \text{"a ``complex'' string"} & 0 & 0 & 0 & \text{"G6 string"} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

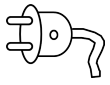
Lesen aller Zellen

```
C := importData_ODS("my SMath ODS input.ods"; "Sheet2")
```

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \text{"a text"} & 2,54 \cdot 10^{3}\ \frac{kg}{s^2} & 0,86 & 2 & 0 & 0 \\ 0 & 0 & \text{"another text"} & 0,84 & 1 & 1 & 0 \\ 0 & 9,46 \cdot 10^{-28} & 5,46 \cdot 10^{28} & 0 & \text{"formatted text"} & 0 & 0 \\ 0 & 0 & \text{"a ``complex'' string"} & 0 & 0 & 0 & \text{"G6 string"} \end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

```
importData.XLSX(Datei; Blatt)
importData.XLSX(Datei; Blatt; Zelle)
importData.XLSX(Datei; Blatt; ZelleLO; ZelleRU)
```



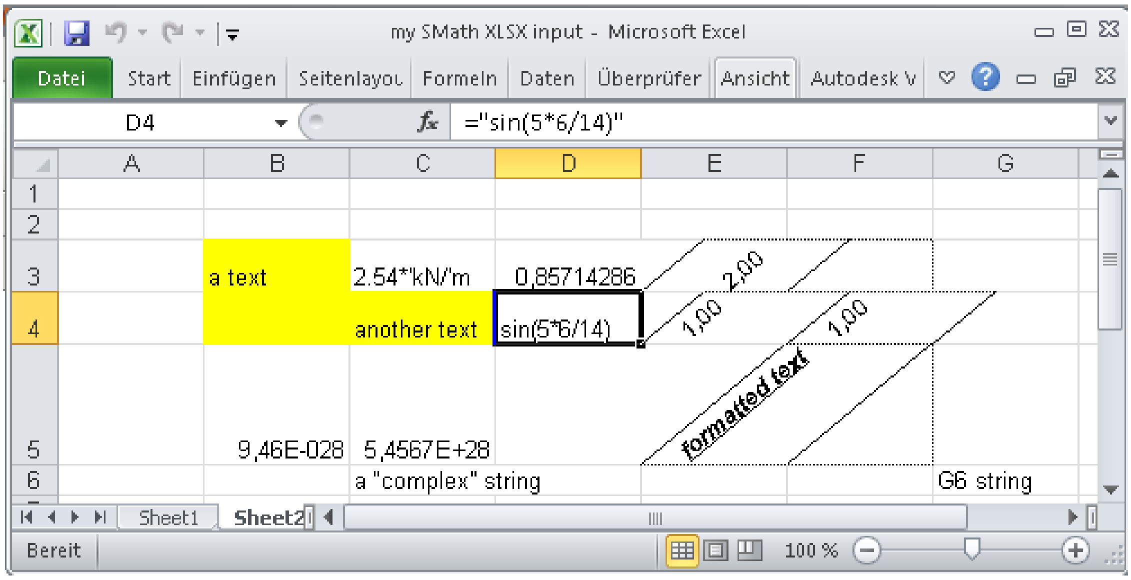
Liest Daten aus *xlsx*-Dateien. Anzugeben sind der Dateiname (`Datei`), der Blattname (`Blatt`), die linke obere und die rechte untere Zelladresse des auszulesenden Bereichs

(`ZelleLO` und `ZelleRU`). Wenn nur eine Zelle auszulesen ist, reicht die Angabe einer Adresse (`Zelle`). In jedem Fall wird das Ergebnis als Matrix erzeugt, gegebenenfalls mit der Größe 1×1.

Die Adressen sind als Zeichenkette aus Spalten-Kennbuchstabe und Zeilennummer anzugeben.

Der Dateiname kann einen absoluten oder relativen Pfad enthalten, letzterer bezieht sich auf das aktuelle Verzeichnis (siehe dazu Abschnitt 3.17).

Dies ist die Beispieldatei *my SMath XLSX input.ods* des Plugins *Data Exchange*:



Einlesen mit SMath:

$$A := \text{importData}_{XLSX}(\text{"my SMath XLSX input.xlsx"}; \text{"Sheet2"}; \text{"G6"})$$

$$A = (\text{"G6 string"})$$

$$B := \text{importData}_{XLSX}(\text{"my SMath XLSX input.xlsx"}; \text{"Sheet2"}; \text{"B2"}; \text{"G7"})$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \text{"a text"} & 2{,}54 \cdot 10^{3} \frac{kg}{s^{2}} & 0{,}8571 & 2 & 0 & 0 \\ 0 & \text{"another text"} & 0{,}8408 & 1 & 1 & 0 \\ 9{,}4567 \cdot 10^{-28} & 5{,}4567 \cdot 10^{28} & 0 & \text{"formatted text"} & 0 & 0 \\ 0 & \text{"a "complex" string"} & 0 & 0 & 0 & \text{"G6 string"} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Data Exchange* (siehe Abschnitt 9.4.4).

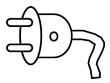
```
int(Integrand; Integrationsvariable)
```

Unbestimmtes Integral. Für die Berechnung ist das Maxima-Plugin erforderlich (siehe Abschnitt 4.12).

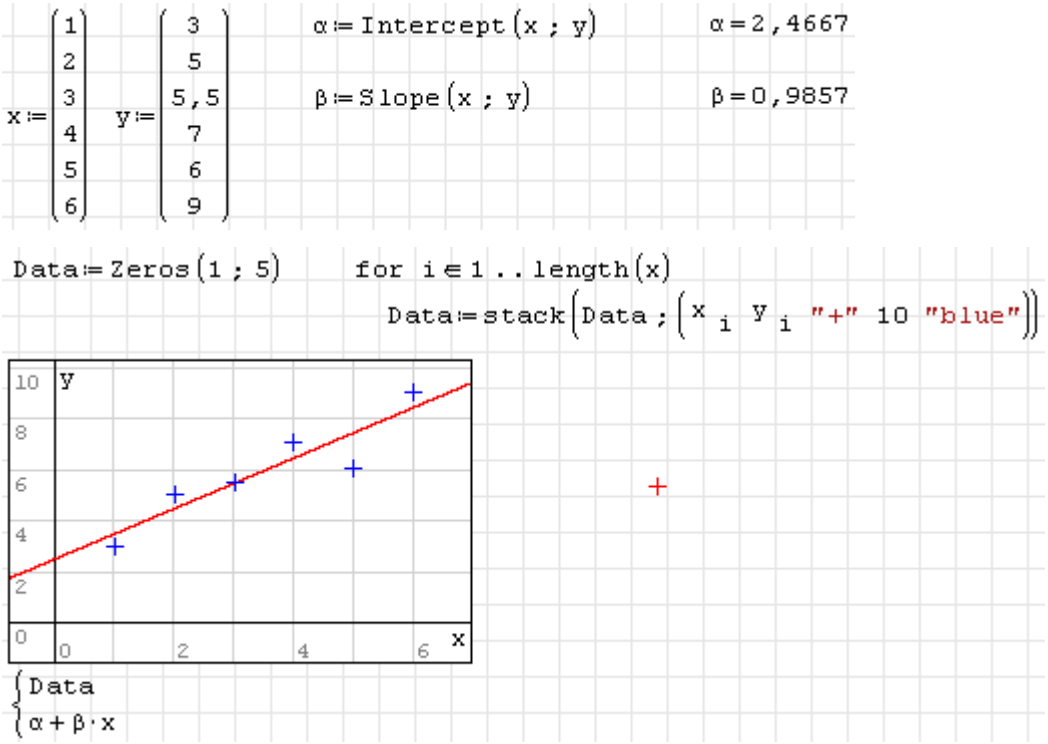
```
int(Integrand; Integrationsvariable; Untergrenze; Obergrenze)
```

bestimmtes Integral, SMath kann nur bestimmte Integrale mit zahlenmäßig auswertbaren Grenzen berechnen. Die Integrationsvariable und deren Grenzen müssen dimensionslos sein. Symbolische Integration bestimmter und unbestimmter Integrale erfordert das Maxima-Plugin (siehe Abschnitt 4.12).

```
Intercept(x; y)
```

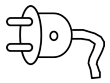


Achsenabschnitt der Regressionsgerade durch die mit den Vektoren $\underline{x}$ und $\underline{y}$ gegebenen Datenpunkte.

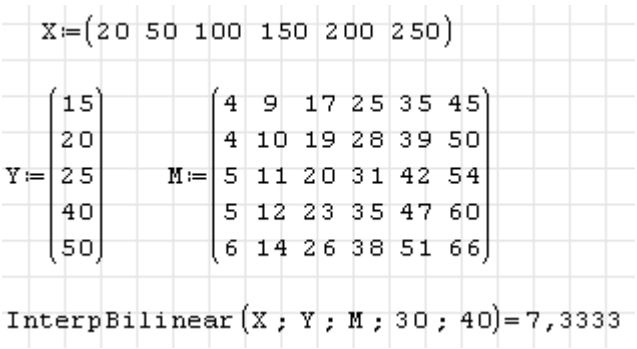


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
InterpBilinear(X; Y; M; x; y)
```



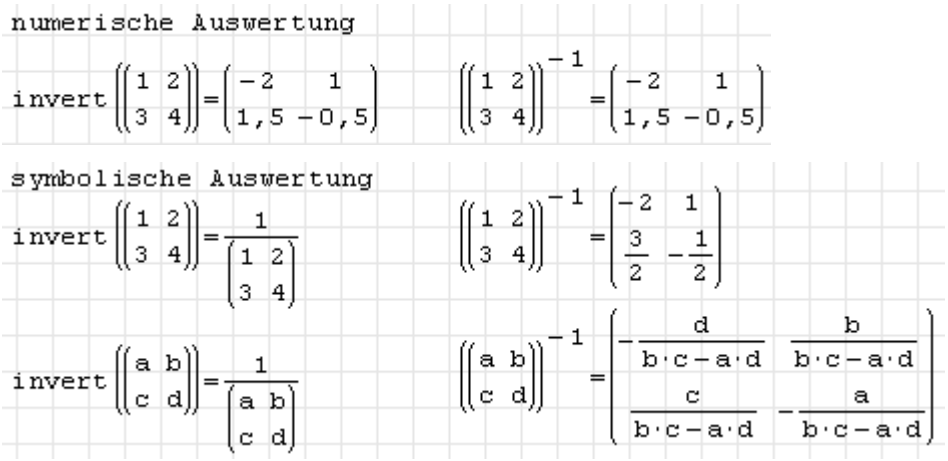
Lineare Interpolation der Matrix `M` mit den Spaltenwerten `X` und den Zeilenwerten `Y` am Punkt `x`, `y`.



Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.12).

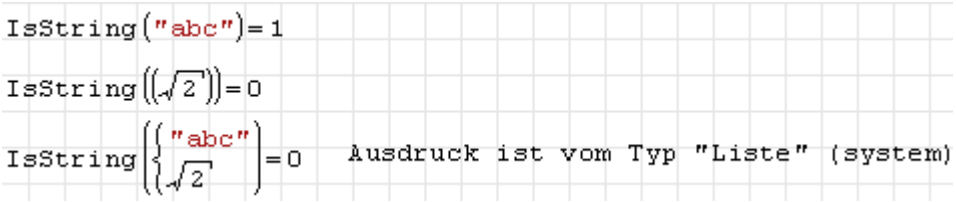
```
invert(M)
```

Liefert die Inverse der Matrix `M`. Die Funktion verhält sich bei symbolischer Auswertung leider nicht identisch zu der Darstellung mit dem Exponenten -1.

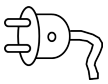



```
IsString(Ausdruck)
```

Liefert 1, wenn der Ausdruck eine Zeichenkette (*string*) ist, anderenfalls 0.



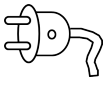

```
Jacobian(Funktionen; Argumente)
```



Berechnet die Jacobimatrix aus *m* als Liste gegebener Funktionen abgleitet nach *n* als Liste gegebenen Variablen.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

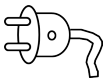
```
Jacobian.CD(Funktionen; Argumente; Schrittweite)
```



Berechnet die Jacobimatrix aus *m* als Liste gegebener Funktionen abgleitet nach *n* als Liste gegebenen Variablen. Die Berechnung erfolgt mittels zentralem Differenzenquotienten mit der gegebenen Schrittweite.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
Kurtosis(y)
```



Wölbung (Kurtosis) β_2 einer durch den Vektor $\underline{y}$ gegebenen Datenreihe.

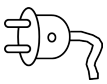
$$\beta_2 = \frac{\mu_4(y)}{\mu_2(y)^2}$$

mit den k-ten zentralen Momenten μ_k.

$y:=(3\ 5\ 5{,}5\ 7\ 6\ 9)^T$ Kurtosis(y)=2,4222

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
KurtosisExcess(y)
```



Exzess (Überkurtosis), die um die Wölbung der Normalverteilung ($\beta_2 = 3$) reduzierte Wölbung einer durch den Vektor $\underline{y}$ gegebenen Datenreihe.

$$\text{Exzess(y)} = \beta_2(y) - 3$$

$y:=(3\ 5\ 5{,}5\ 7\ 6\ 9)^T$ KurtosisExcess(y)=-0,5778

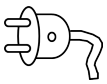
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
length(Matrix)
```

Liefert die Elementanzahl einer Matrix:

$$\text{length}\begin{pmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix} = 6$$

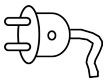
```
LevenbergMarquardt(Funktionen; Startwerte; Genauigkeit; maxIter)
LevenbergMarquardt(Funktionen; Startwerte; Genauigkeit; maxIter;
Schalter)
```



Optimierer nach dem Levenberg-Marquardt-Verfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
LevenbergMarquardt.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter)
LevenbergMarquardt.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter; Schalter)
```



Optimierer nach dem Levenberg-Marquardt-Verfahren mit numerischer Ableitungsberechnung mittels zentralem Differenzenquotient. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotienten. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

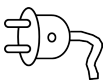
```
line()
```

Anweisungsblock. Er dient der Gruppierung von Ausdrücken. Rückgabewert ist der Wert des letzten Ausdrucks im Block.

```
linterp(X-Daten; Y-Daten; x)
```

Lineare Interpolation der Vektoren `X-Daten` und `Y-Daten` am Punkt `x`.

```
maple(ausdruck)
```



+

Die Funktion benutzt eine DOS-Version von MapleV, um den Ausdruck symbolisch auszuwerten.

Ablauf beim Funktionsaufruf

1. Der Ausdruck wird in Zeilenform gebracht und auf Maple-Syntax transformiert.
2. Die so erhaltene Zeile wird in eine temporäre Textdatei geschrieben.
3. Maple wird mit der Textdatei als Parameter aufgerufen.
4. Maple schreibt das Ergebnis in eine Textdatei.
5. Die Ergebnisdatei wird gelesen, auf SMath Syntax transformiert und an SMath zurückgegeben.

Beispiele

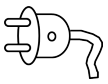
$$\text{maple}(\text{asin}(1)) = \frac{\pi}{2}$$

$$\text{maple}\left(\int x^2 \, dx\right) = \frac{x^3}{3}$$

$$\text{maple}\left(\text{solve}\left(\sqrt{\ln(x)} = 2 \; ; \; x\right)\right) = \exp(4)$$

Die Funktion wird im Plugin *Maple Wrapper* bereitgestellt (siehe Abschnitt 9.4.8).

```
mapUnknowns(Funktion)
mapUnknowns(Funktion; Zustand)
mapUnknowns(Funktion; Zustand; Namen)
```



Liefert einen Vektor der undefinierten Variablen in der Definition der Funktion. Achtung: Symbolische Auswertung einschalten!

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
mat()
```

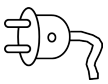
Matrixobjekt. Diese Funktion dient der internen Darstellung von Matrizen.

```
matrix(Zeilen; Spalten)
```

Erzeugt eine mit Nullen besetzte Matrix der angegebenen Größe.

$$\mathrm{matrix}(2\,;\,4)=\begin{pmatrix}0&0&0&0\\0&0&0&0\end{pmatrix}$$

```
mat2sys(Matrix)
```

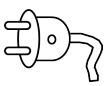


Wandelt die gegebene Matrix in eine Liste („System") um. Dabei werden rekursiv alle eventuellen Verschachtelungsebenen aufgelöst.

$$\mathrm{mat2sys}\left(\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}\right)=\begin{Bmatrix}1\\2\\3\\4\\5\\6\end{Bmatrix}\qquad \mathrm{mat2sys}\left(\begin{pmatrix}\begin{pmatrix}1&2\\3&4\end{pmatrix}\\5\\6\end{pmatrix}\right)=\begin{Bmatrix}1\\2\\3\\4\\5\\6\end{Bmatrix}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

```
mat2sys.1(Matrix)
```



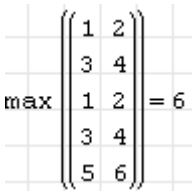
Wandelt die gegebene Matrix in eine Liste („System") um. Dabei wird nur die äußere Matrix (erste Verschachtelungsebene) aufgelöst.

$$\mathrm{mat2sys}_1\left(\begin{pmatrix}\begin{pmatrix}1&2\\3&4\end{pmatrix}\\5\\6\end{pmatrix}\right)=\begin{Bmatrix}\begin{pmatrix}1&2\\3&4\end{pmatrix}\\5\\6\end{Bmatrix}$$

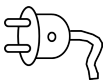
Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`max(Matrix)`

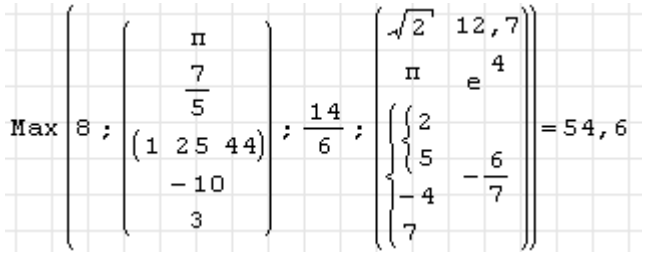
Liefert das maximale Element einer Matrix:



`Max()`

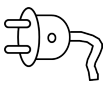


Liefert den Maximalwert aus beliebig vielen beliebig verschachtelten Zahlen, Listen oder Matrizen.



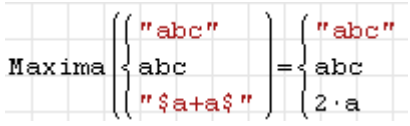
Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`Maxima(expr)`
`Maxima(expr;''debug'')`



Ausführung des Ausdrucks `expr` in einer laufenden oder neu zu startenden Maxima-Sitzung. Für jede SMath-Instanz (Programmfenster) gibt es einen separaten Maxima-Prozess.

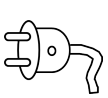
Zeichenketten werden als Zeichenketten übersetzt. Zeichenketten, die mit $ beginnen und enden, werden direkt als Maxima-Input durchgereicht (ohne Übersetzung).



Ist das Argument `"debug"` (kann auch `debug` geschrieben werden) vorhanden, dann wird ein Debug-Fenster geöffnet, in dem die Übersetzung des Ausdrucks `expr` Schritt für Schritt nachvollzogen werden kann.

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Kapitel **??**).

`MaximaControl(text)`



Steuerung des Maxima-Plugins. Die folgenden Werte für `text` werden akzeptiert:

`"settings"` Aufruf des „Settings"-Fensters (Maxima-Einstellungen, auch zugänglich über das Menü **Einfügen> Maxima control window> Settings**)

`"restart"` Kaltstart, Neuaufbau der Kommunikation mit Maxima.

`"cleanup"` Warmstart, Zurücksetzen der Variablen in Maxima, etwas schneller als der Kaltstart.

`""` oder leeres Argument: Anzeige der Maxima-Version.

Die Wirkung von `"restart"` und `"cleanup"` erläutert folgende Tabelle:

Wirkung	`"restart"`	`"cleanup"`
Log löschen	x	x
Variablen löschen	x	x
Neue Sitzung	x	
Übernahmestatus der Funktionen zurücksetzen	x	x

$t_0 := time(0)$

MaximaControl("restart")="Restart complete."

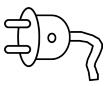
$t_1 := time(0)$ $t_1 - t_0 = 2{,}709\ s$

MaximaControl("cleanup")="Restart complete."

$t_2 := time(0)$ $t_2 - t_1 = 2{,}299\ s$

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

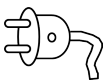
```
MaximaDefine(name)
MaximaDefine(name;value)
```



Definiert Variablen in Maxima.

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Kapitel **??**).

```
MaximaLog(text)
```



Funktion zum Zugriff auf den Datenaustausch zwischen SMath und Maxima. Die folgenden Werte für `text` werden akzeptiert:

`"clear"` Log zurücksetzen. Dies erfolgt auch bei den Aufrufen `MaximaControl("restart")` und `MaximaControl("cleanup")`.

`"all"` gesamtes Log anzeigen.

`"big"` Log-Fenster öffnen, alternativ zu **Einfügen> Maxima control window> Log**.

`""` oder leeres Argument: Detailinformation zum letzten ausgeführten Maxima-Befehl.

Die Log-Information wird im Arbeitsblatt als mehrzeilige Zeichenkette angezeigt. Anführungszeichen im Log werden dabei durch $ dargestellt.

MaximaControl("restart")="Restart complete."

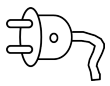
$Maxima\left(\begin{pmatrix}"a+b"\\ a+a\end{pmatrix}\right)=\begin{pmatrix}"a+b"\\ 2\cdot a\end{pmatrix}$

```
MaximaLog(■)= "Request: [$a+b$,a+a];
              Answer:
              (%o16) [$a+b$,2*a]
              (%i17)
              Received bytes: 27
              SMath get: "
```

```
MaximaLog("all")= "pid=13836
                  Maxima 5.33.0 http://maxima.sourceforge.net
                  using Lisp GNU Common Lisp (GCL) GCL 2.6.10 (a.k.a.
                  Distributed under the GNU Public License. See the fi
                  Dedicated to the memory of William Schelter.
                  The function bug_report() provides bug reporting inf
                  (%i1) display2d:false$
                  (%i2) linel:700$
                  (%i3) simpsum:true$
                  (%i4) logsimp:true$
                  (%i5) keepfloat:true$
                  (%i6) ratprint:false$
                  (%i7) load(vect)$
                  (%i8) load(abs_integrate)$
                  (%i9) setup_autoload(mnewton, mnewton)$
                  (%i10) setup_autoload(draw, draw2d, draw3d)$
                  (%i11) setup_autoload(lsquares, lsquares_mse, lsquar
                  (%i12) logb(x,y):=log(x)/log(y)$el(x,y):=x[y]$
                  (%i14) el(x,[y]):=if equal(length(y),1) then x[y[1]]
                  (%i15) check;
                  (%o15) check
                  (%i16) [$a+b$,a+a];
                  (%o16) [$a+b$,2*a]
                  (%i17) "
```

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

MaximaTakeover(text;...)



Feinsteuerung der Funktionsübernahme durch Maxima. Die Funktionen `int()`, `diff()`, `det()`, `lim()` und `sum()` können durch Maxima ausgeführt werden, um ihnen symbolischen Fähigkeiten zu verleihen oder diese zu erweitern. Die folgenden Werte für `text` werden akzeptiert:

`"all"` Übernahme aller Funktionen durch Maxima,

`"none"` alle Funktionen original SMath.

`"diff()"` bis zu fünf beliebig kombinierte Argumente zur Übernahme der Funktionen. Alle nicht angegebenen Funktionen sind original SMath.

`"int()"`

`"det()"`

`"sum()"`

`"lim()"`

Keine oder ungültige Argumente setzen den Zustand auf `"none"` zurück. Dies geschieht auch bei Zurücksetzen der Sitzung mit `MaximaControl("restart")` und `MaximaControl("cleanup")`.

Rückgabewert ist eine Bestätigungsmeldung, welche Funktionen übernommen werden.

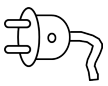
MaximaTakeover("all")="int(), diff(), det(), lim() and sum() now use Maxima"

$$\frac{d}{dx}\operatorname{sech}(x)=-\operatorname{sech}(x)\cdot\tanh(x) \qquad \int \sin(x)\,dx=-\cos(x)$$

$$\left|a\cdot\begin{pmatrix}1 & 2\\ b & 3\end{pmatrix}\right|=a^2\cdot(3-2\cdot b) \qquad \sum_{j=1}^{n} j=\frac{n\cdot(1+n)}{2} \qquad \lim_{x\to\infty}\frac{x^2}{x}=\infty$$

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

Mean(Daten)



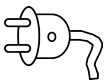
Liefert den arithmetischen Mittelwert der Werte in Daten Matrix, Vektor oder Liste):

$$\mu=\frac{1}{n}\sum_{i=1}^{n}x_i$$

$$\text{Mean}\left(\begin{pmatrix}8\\2\\1\end{pmatrix}\right)=3,6667 \qquad \text{Mean}\left(\begin{pmatrix}1 & 3 & 5 & 1 & 6 & 2\\ 3 & 2 & 5 & 2 & 3 & 5\\ 5 & 2 & 1 & 3 & 4 & 1\end{pmatrix}\right)=3 \qquad \text{Mean}\left(\begin{pmatrix}1\\2\\8\end{pmatrix}\right)=3,6667$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

Median(Daten)



Liefert den Median der Daten (Matrix, Vektor, Liste).

$$\text{Median}\left(\begin{pmatrix}8\\2\\1\end{pmatrix}\right)=2 \qquad \text{Median}\left(\text{Dice}_{M}\right)=4$$

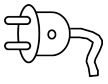
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

min(Matrix)

Liefert das kleinste Element einer Matrix:

$$\min\left(\begin{pmatrix}1 & 2\\ 3 & 4\\ 1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix}\right)=1$$

Min()



Liefert den Minimalwert aus beliebig vielen beliebig verschachtelten Zahlen, Listen oder Matrizen.

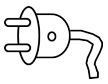
$$\mathrm{Min}\left(8;\begin{pmatrix}\pi\\ \frac{7}{5}\\ (1\ 25\ 44)\\ -10\\ 3\end{pmatrix};\frac{14}{6};\begin{pmatrix}\sqrt{2} & 12{,}7\\ \pi & e^{4}\\ \begin{cases}2\\5\\-4\\7\end{cases} & -\frac{6}{7}\end{pmatrix}\right)=-10$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

```
minor(Matrix; Index1; Index2)
```

Minor des Matrixelements (Index1, Index2)

```
Mode(Daten)
Mode(Daten; 1)
```



Liefert den Modalwert der Daten (den oder die am häufigsten vorkommenden Werte). Ist das optionale Argument vorhanden und ungleich Null, so wird auch die Häufigkeit des Wertes oder der Werte angegeben.

$$\mathrm{Mode}(3)=3 \qquad \mathrm{Mode}(3;1)=\begin{pmatrix}3\\(1)\end{pmatrix}$$

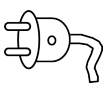
$$\mathrm{Mode}\left(\begin{pmatrix}3\\2\\1\end{pmatrix}\right)=\begin{pmatrix}3\\2\\1\end{pmatrix} \qquad \mathrm{Mode}\left(\begin{pmatrix}3\\2\\1\end{pmatrix};1\right)=\begin{pmatrix}3\\2\\1\\(1)\end{pmatrix}$$

$$\mathrm{Mode}\left(\begin{pmatrix}3\\3\\2\\1\end{pmatrix}\right)=(3) \qquad \mathrm{Mode}\left(\begin{pmatrix}3\\3\\2\\1\end{pmatrix};1\right)=\begin{pmatrix}(3)\\(2)\end{pmatrix}$$

$$\mathrm{Mode}\left(\begin{pmatrix}1&3&5&1&6&2\\3&2&5&2&3&5\\5&2&1&3&4&1\end{pmatrix}\right)=\begin{pmatrix}1\\2\\3\\5\end{pmatrix} \qquad \mathrm{Mode}\left(\begin{pmatrix}1&3&5&1&6&2\\3&2&5&2&3&5\\5&2&1&3&4&1\end{pmatrix};1\right)=\begin{pmatrix}1\\2\\3\\5\\(4)\end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

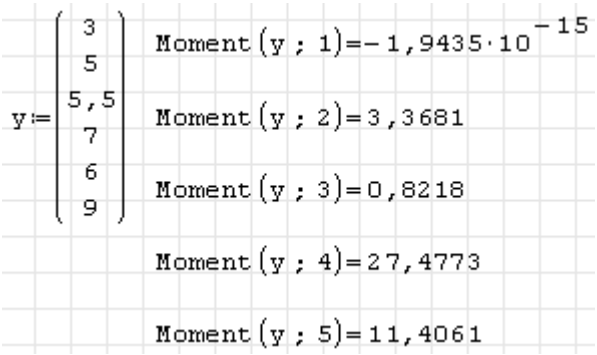
```
Moment(Daten; k)
```



k-tes zentrales Moment einer Datenreihe x.

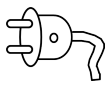
$$\mu_k(x,k)=\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^k$$

Darin ist μ der Mittelwert der Datenreihe.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

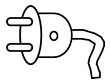
```
NCGM(Funktionen; Startwerte; Genauigkeit; maxIter)
NCGM(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Optimierer (konjugierte Gradienten). Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

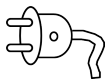
```
NCGM.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit; maxIter)
NCGM.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit; maxIter;
Schalter)
```



Optimierer (konjugierte Gradienten) mit numerischer Ableitungsberechnung mittels zentralem Differenzenquotient. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotienten. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

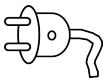
```
NewtonMethod(Funktionen; Startwerte; Genauigkeit; maxIter)
NewtonMethod(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Optimierer nach dem Newton-Verfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

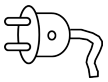
```
NewtonMethod.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter)
NewtonMethod.CD(Funktionen; Startwerte; Schrittweite; Genauigkeit;
maxIter; Schalter)
```



Optimierer nach dem Newton-Verfahren mit numerischer Ableitungsberechnung (zentraler Differenzenquotient). Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotient. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

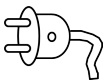
```
NewtonMethod.CDGSS(Funktionen; Startwerte; Genauigkeit; maxIter)
NewtonMethod.CDGSS(Funktionen; Startwerte; Genauigkeit; maxIter;
Schalter)
```



Optimierer nach dem Newton-Verfahren Line Search nach dem Goldenen-Schnitt-Suchverfahren und numerischer Ableitungsberechnung (zentraler Differenzenquotient). Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Schrittweite: Schrittweite im zentralen Differenzenquotient. Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

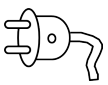
```
NewtonMethod.GSS(Funktionen; Startwerte; Genauigkeit; maxIter)
NewtonMethod.GSS(Funktionen; Startwerte; Genauigkeit; maxIter; Schalter)
```



Optimierer nach dem Newton-Verfahren mit Line Search nach dem Goldenen-Schnitt-Suchverfahren. Funktionen: Liste der Zielfunktionen, Startwerte: Liste der Startwerte für die unbekannten Variablen (in alphabetischer Reihenfolge), Genauigkeit: geforderte Vektornorm für den Gradienten, maxIter: maximale Zahl der Iterationen, Schalter: falls vorhanden und ungleich 0 wird die Zahl der benötigten Iterationen ausgegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

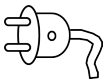
```
NewtonRaphson(Funktionen; Startwerte; Genauigkeiten; maxIter)
NewtonRaphson(Funktionen; Startwerte; Genauigkeiten; maxIter; Schalter)
```



Nichtlinearer Löser nach dem Newton-Raphson-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen mit den gegebenen Startwerten. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Iterationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Iterationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
NewtonRaphson.CD(Funktionen; Startwerte; Schrittweite; Genauigkeiten;
maxIter)
NewtonRaphson.CD(Funktionen; Startwerte; Schrittweite; Genauigkeiten;
maxIter; Schalter)
```



Nichtlinearer Löser nach dem Newton-Raphson-Verfahren mit Berechnung der Ableitung durch den zentralen Differenzenquotienten. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen mit den gegebenen Startwerten. Der zentrale Differenzenquotient wird mit „Schrittweite" berechnet. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Iterationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Iterationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
norm1(Matrix)
```

Spaltensummennorm der Matrix (Summennorm bei einem Vektor). Das Beispiel vergleicht die eingebaute Funktion norm1() mit einem Nachbau

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Spaltensummennorm

$$\text{norm}_1(M\#) := \left| \begin{array}{l} \text{for } j \in 1..\text{cols}(M\#) \\ \quad a\#_j := \sum_{i=1}^{\text{rows}(M\#)} \left| M\#_{i\,j} \right| \\ \max(a\#) \end{array} \right.$$

$$\text{norm}_1(A) = 12 \qquad \text{norm}_1\left(A^T\right) = 11 \qquad \text{norm}_1(b) = 6$$

$$\text{norm1}(A) = 12 \qquad \text{norm1}\left(A^T\right) = 11 \qquad \text{norm1}(b) = 6$$

```
norme(Matrix)
```

Euklidische Norm der Matrix (Wurzel aus der Quadratsumme aller Elemente).

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Euklidische Norm

$$\text{norm}_e(M\#) := \sqrt{\text{tr}\left(M\#^T \cdot M\#\right)}$$

$$\text{norm}_e(A) = 9{,}54 \qquad \text{norm}_e\left(A^T\right) = 9{,}54 \qquad \text{norm}_e(b) = 3{,}74$$

$$\text{norme}(A) = 9{,}54 \qquad \text{norme}\left(A^T\right) = 9{,}54 \qquad \text{norme}(b) = 3{,}74$$

`normi(Matrix)`

Zeilensummennorm der Matrix (Maximumnorm bei einem Vektor). Das Beispiel vergleicht die eingebaute Funktion normi() mit einem Nachbau

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Zeilensummennorm

$$\text{norm}_\infty(M\#) := \left| \begin{array}{l} \text{for } i \in 1..\text{rows}(M\#) \\ \quad a\#_i := \sum_{j=1}^{\text{cols}(M\#)} \left| M\#_{i,j} \right| \\ \max(a\#) \end{array} \right.$$

$$\text{norm}_\infty(A) = 11 \qquad \text{normi}(A) = 11$$

$$\text{norm}_\infty(A^T) = 12 \qquad \text{normi}(A^T) = 12$$

$$\text{norm}_\infty(b) = 3 \qquad \text{normi}(b) = 3$$

`nthroot(Zahl1; Zahl2)`

n-te Wurzel aus Zahl2, Zahl1 = n

`num2str(Ausdruck)`

liefert die symbolisch vereinfachte interne Textdarstellung des Ausdrucks als Zeichenkette.

$$\text{num2str}(2,3) = \text{"23/10"}$$

$$\text{num2str}(a+b) = \text{"a+b"}$$

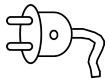
$$\text{num2str}\left(\begin{cases} \text{"a"} \\ \pi \end{cases}\right) = \text{"sys("a",π,2,1)"}$$

$$\text{num2str}\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\right) = \text{"mat(1,2,3,4,2,2)"}$$

$$\text{num2str}\left(\int x^2 \, dx\right) = \text{"int(x^2,x)"}$$

Bug: Wenn der Ausdruck Text mit Leerzeichen enthält, versagt die Funktion (unsinniges Verhalten).

`Ones(Zeilen)`
`Ones(Zeilen; Spalten)`

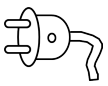


Erzeugt eine mit Einsen belegte Matrix.

$$\text{Ones}(5) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \text{Ones}(2;3) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

```
ODE.2(ODE; f(x); x)
```



Löst gewöhnliche Differenzialgleichungen bis zu zweiter Ordnung. `ODE` ist die Differentialgleichung. Diese wird so an Maxima übergeben, wie sie ist, ohne vorherige Ersetzung von Symbolen. `f(x)` ist die unbekannte Funktion, `x` ist die unabhängige Variable.

Das Ergebnis ist eine boolsche Gleichung, die von `Assign()` für die Definition der gesuchten Funktion verwendet werden kann. Die Lösung enthält abhängig von der Ordnung der DGL eine oder zwei Integrationskonstanten, `c` (erste Ordnung) oder `k1` und `k2` bei zweiter Ordnung.

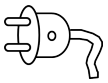
$$\text{Maxima}\left(\text{assume}\left(\omega^2>0\right)\right)=\omega^2>0$$

$$L:=\text{ODE}_2\left(\left(\frac{d^2}{dt^2}w(t)+\omega^2\cdot w(t)=0\right);\ w(t);\ t\right)$$

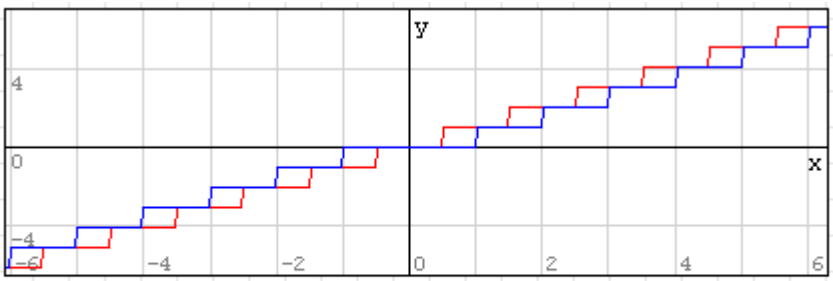
$$L=w(t)=\left(k1\cdot\sin(\omega\cdot t)+k2\cdot\cos(\omega\cdot t)\right)$$

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Kapitel **??**).

```
OoM(Variable)
```



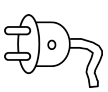
Größenordnung der Variable (von Null weg gerundeter dekadischer Logarithmus). Bei Variablen mit Einheiten bezieht sich das Ergebnis auf die Basiseinheit.



$$\begin{cases}\text{OoM}\left(10^x\right)\\ \text{eOoM}\left(10^x\right)\end{cases}$$

Die Funktion ist Bestandteil des Plugins *customFunction* (Seite 214).

```
pdf.Binomial(n; p)
pdf.Binomial(k; n; p)
```

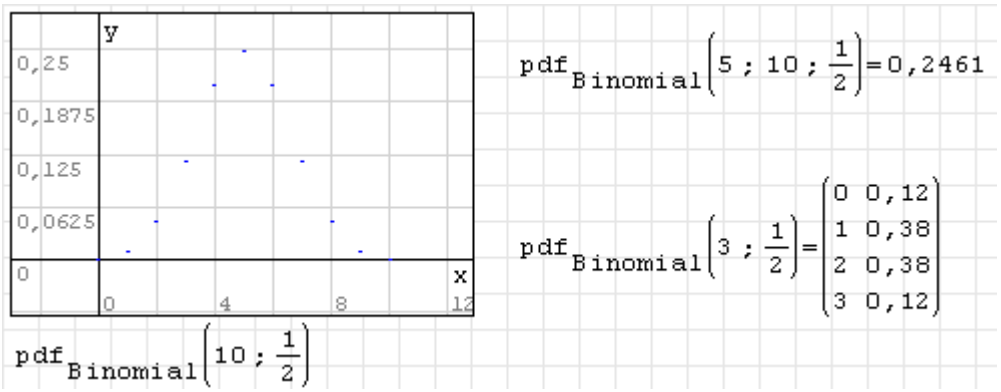


Häufigkeitsfunktion der Binomialverteilung. Sie beschreibt die Anzahl der Erfolge in einer Serie von gleichartigen und unabhängigen Versuchen, die jeweils genau zwei mögliche Ergebnisse haben („Erfolg" oder „Misserfolg"). p ist die Erfolgswahrscheinlichkeit bei einem Versuch, n ist die Zahl der Versuche.

Wird das Argument k angegeben, liefert die Funktion die Wahrscheinlichkeit, mit den gegebenen Parametern p und n genau k Erfolge zu erzielen:

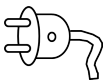
$$f(k,p,n) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} = \frac{n!}{k!\,(n-k)!} p^k (1-p)^{n-k}$$

Wird das Argument k nicht angegeben, dann liefert die Funktion eine plotbare zweispaltige Matrix aus Paaren von k und $f(k,p,n)$ mit $k \in 0 \ldots n$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

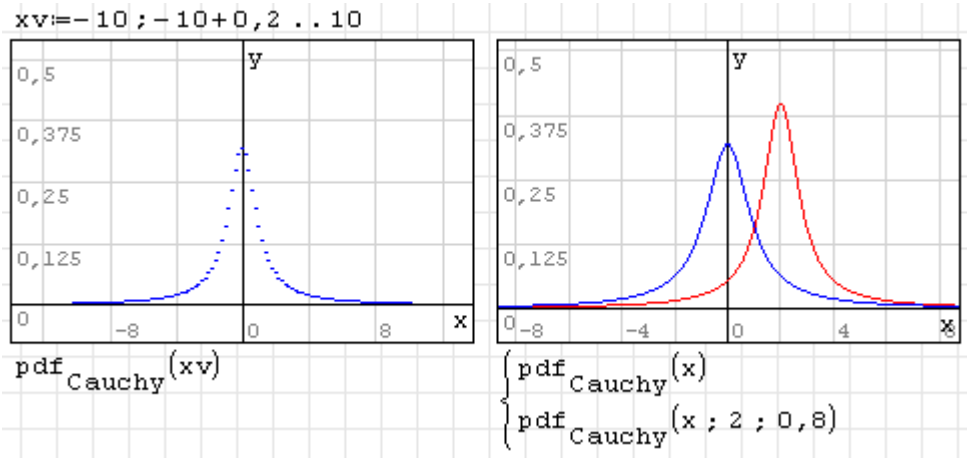
```
pdf.Cauchy(x)
pdf.Cauchy(x; t; s)
```



Häufigkeitsfunktion der Cauchy-Verteilung. mit dem Ortsparameter t (Standardwert 0) und dem Breitenparameter s (Standardwert 1):

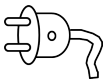
$$f(x,t,s) = \frac{1}{\pi} \frac{s}{s^2 + (x-t)^2}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,t,s)$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

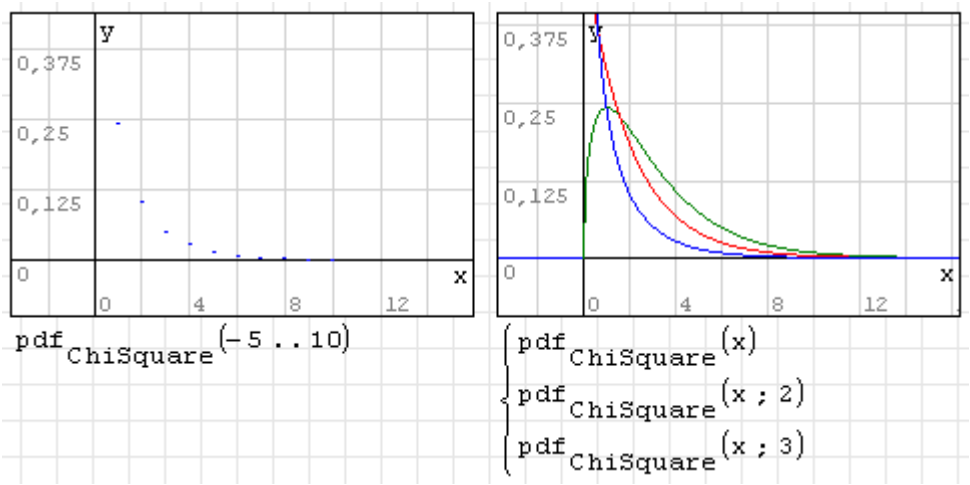
```
pdf.ChiSquare(x)
pdf.ChiSquare(x, n)
```



Häufigkeitsfunktion der χ^2_n- (Chi-Quadrat-)Verteilung mit der Zahl der Freiheitsgrade n (Standardwert 1). Sie beschreibt die Wahrscheinlichkeit, dass die Quadratsumme von n standardnormalverteilten Zahlen kleiner oder gleich x ist.

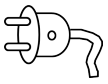
$$f(x,n) = \begin{cases} \frac{x^{\frac{n}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{n}{2}}\Gamma\left(\frac{n}{2}\right)} & x > 0 \\ 0 & x \le 0 \end{cases}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,n)$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

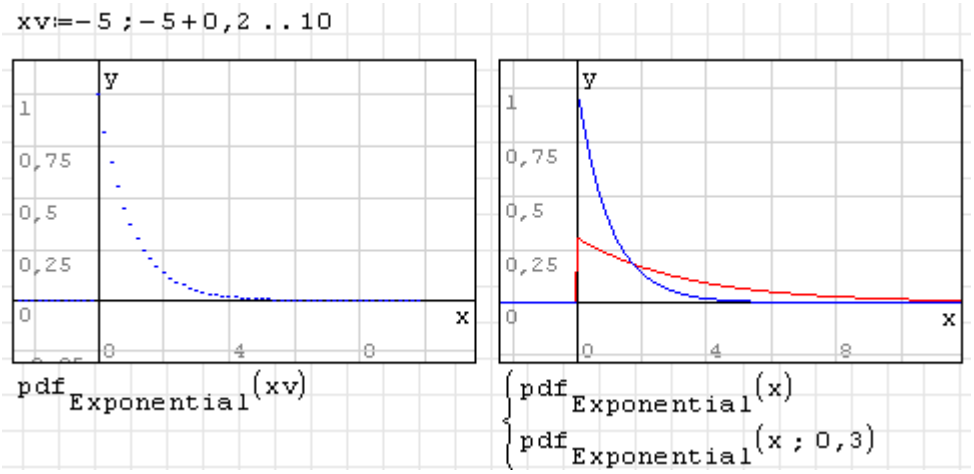
```
pdf.Exponential(x)
pdf.Exponential(x; Ereignisrate)
```



Häufigkeitsfunktion der Exponentialverteilung, kennzeichnet die Wahrscheinlichkeit eines bestimmten Ereignisabstandes x zwischen zufälligen Ereignissen mit mittlerer Ereignisrate λ. Voreinstellung der Ereignisrate ist $\lambda = 1$. Es gilt:

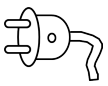
$$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \ge 0 \\ 0 & x < 0 \end{cases}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,\lambda)$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

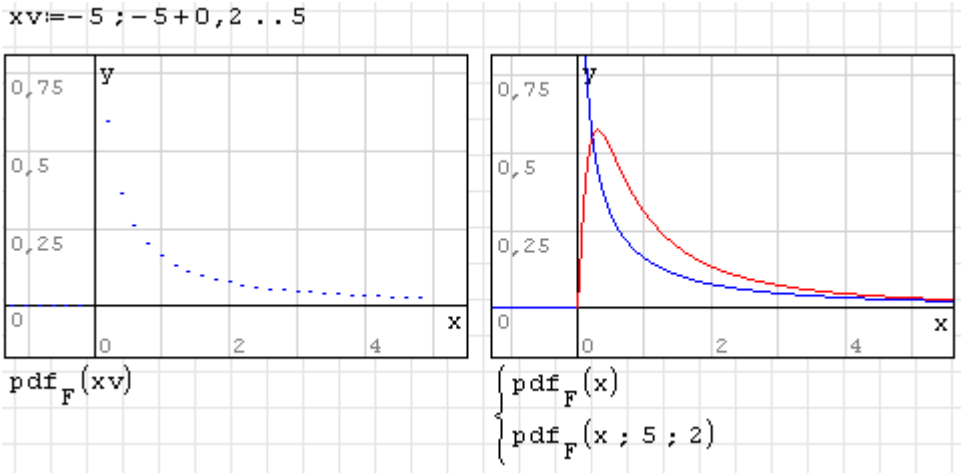
```
pdf.F(x)
pdf.F(x,m,n)
```



Häufigkeitsfunktion der Fisher- oder F-Verteilung mit m Freiheitsgraden im Zähler und n Freiheitsgraden im Nenner. Sie entspricht der Verteilung des Quotienten aus zwei normierten Quadratsummen standardnormalverteilter Zahlen.

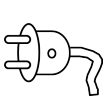
$$f(x,m,n) = \begin{cases} m^{\frac{m}{2}} n^{\frac{n}{2}} \frac{\Gamma\left(\frac{m}{2}+\frac{n}{2}\right)}{\Gamma\left(\frac{m}{2}\right)\Gamma\left(\frac{n}{2}\right)} \frac{x^{\frac{m}{2}-1}}{(mx+n)^{\frac{m+n}{2}}} & x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,m,n)$.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

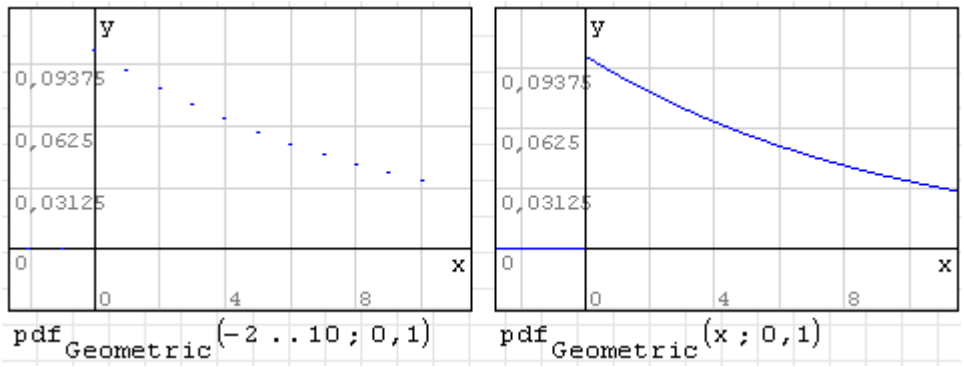
```
pdf.Geometric(x; p)
```



Häufigkeitsfunktion der Geometrischen Verteilung. Sie gibt die Wahrscheinlichkeit an, genau x Fehlversuche zu benötigen, bevor ein Versuch mit der Erfolgswahrscheinlichkeit p erfolgreich ist. Es gilt

$$f(k,p) = p(1-p)^k$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,p)$.



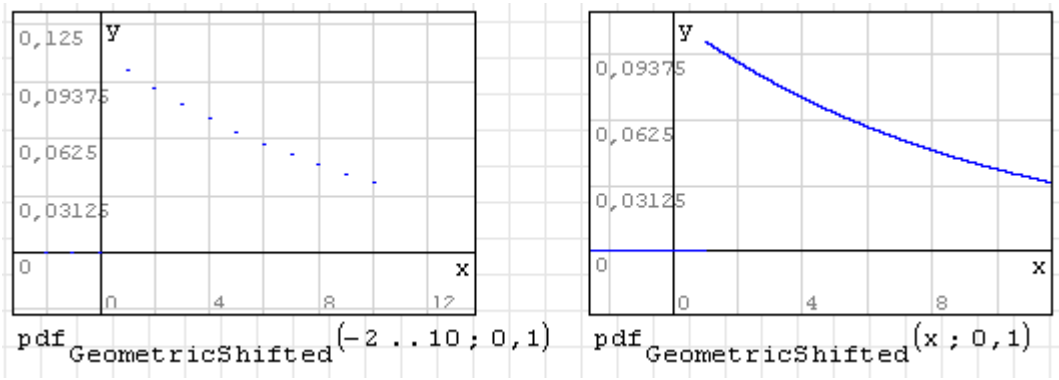
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.GeometricShifted(x; p)
```

Häufigkeitsfunktion der verschobenen Geometrischen Verteilung. Sie gibt die Wahrscheinlichkeit an, genau x Versuche der Erfolgswahrscheinlichkeit p zu benötigen, damit der Erfolg eintritt. Es gilt

$$f(k,p) = p(1-p)^{k-1}$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,p)$.



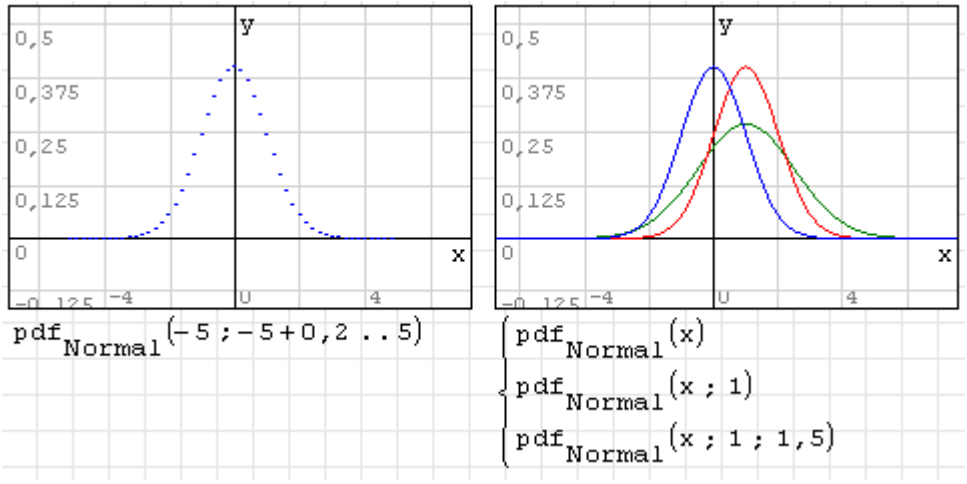
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.Normal(x)
pdf.Normal(x; Mittelwert)
pdf.Normal(x; Mittelwert; Standardabweichung)
```

Häufigkeitsfunktion der Normalverteilung, Voreinstellung ist Mittelwert $\mu = 0$ und Standardabweichung $\sigma = 1$ (Standardnormalverteilung). Es gilt:

$$f(x,\mu,\sigma) = \frac{e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}{\sigma\sqrt{2\pi}}$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $f(x,\mu,\sigma)$ erzeugt.



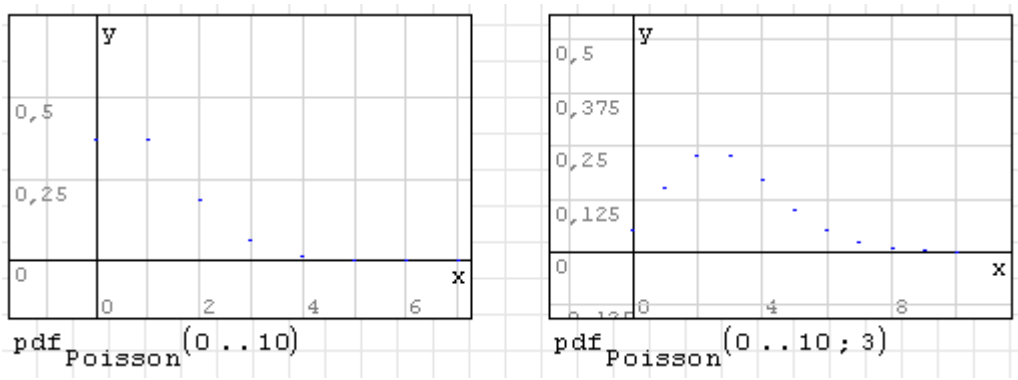
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.Poisson(x)
pdf.Poisson(x; lambda)
```

Häufigkeitsfunktion der Poisson-Verteilung. Gibt die Wahrscheinlichkeit an, mit der ein seltenes Ereignis innerhalb eines Zeitraums oder eines Gebiets genau x mal eintritt, wenn das Ereignis im Mittel lambda (Erwartungswert) mal innerhalb dieses Zeitraums eintritt. Voreinstellung ist $\lambda = 1$. Es gilt:

$$f(x,\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}.$$

Ist x ein Vektor, liefert die Funktion eine plotbare zweispaltige Matrix aus Wertepaaren x und $f(x,\lambda)$. x muss ganzzahlig sein.



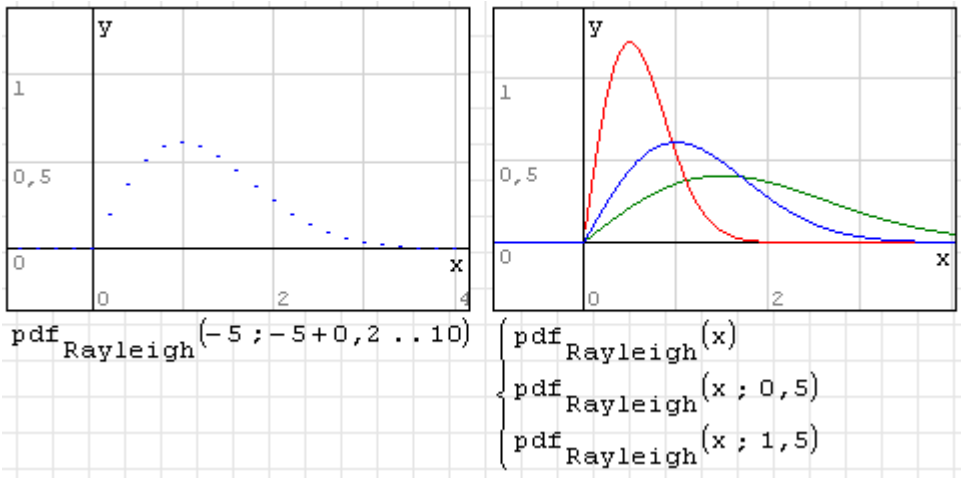
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.Rayleigh(x)
pdf.Rayleigh(x; σ)
```

Häufigkeitsfunktion der Rayleigh-Verteilung. Gibt die Wahrscheinlichkeitdichte an, dass ein Vektor aus zwei mit $\mu = 0$ und σ normalverteilten Komponenten genau den Betrag x hat. Die Standardabweichung ist auf $\sigma = 1$ voreingestellt. Es gilt:

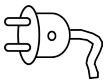
$$f(x,\sigma) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2}$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $f(x,\sigma)$ erzeugt.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

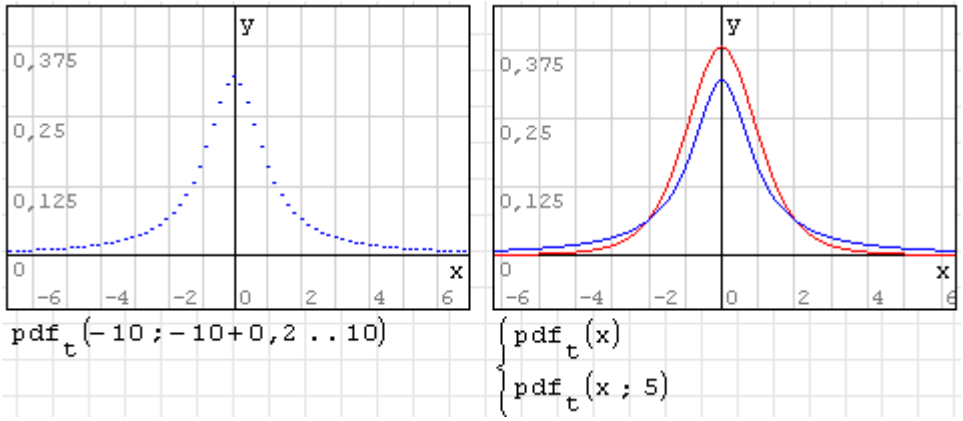
```
pdf.t(x)
pdf.t(x,n)
```



Häufigkeitsfunktion der Studentschen t-Verteilung mit n Freiheitsgraden (Standardwert 1).

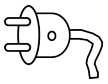
$$f(x,n) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{n\pi}\,\Gamma\left(\frac{n}{2}\right)}\left(1+\frac{x^2}{n}\right)^{-\frac{n+1}{2}}.$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $f(x,n)$ erzeugt.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

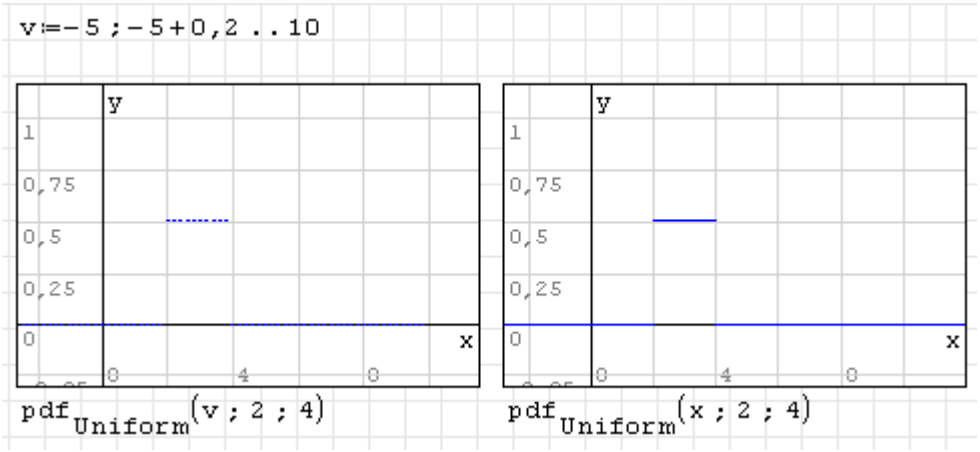
```
pdf.Uniform(x; a; b)
```



Dichtefunktion der stetigen Gleichverteilung im Intervall $[a,b]$ mit

$$f(x,a,b) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{sonst} \end{cases}$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $f(x,a,b)$ erzeugt.



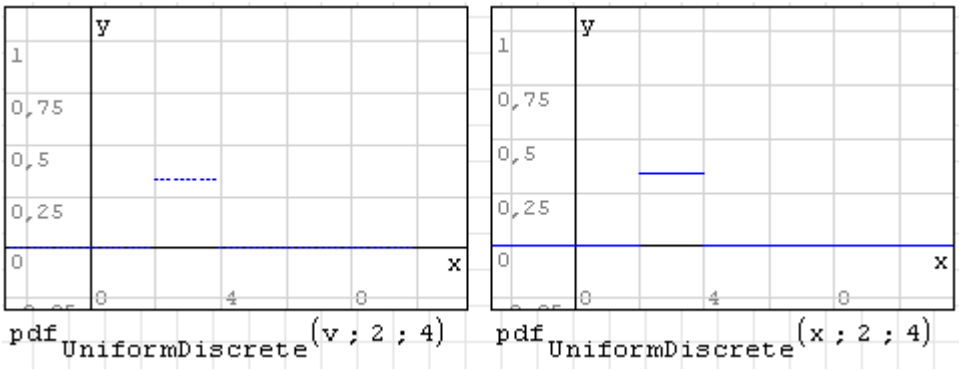
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.UniformDiscrete(x; a; b)
```

Häufigkeitsfunktion der diskreten Gleichverteilung im Intervall $[a,b]$ mit

$$f(x,a,b) = \begin{cases} \frac{1}{b-a+1} & a \le x \le b \\ 0 & \text{sonst} \end{cases}$$

Darin sind x, a und b ganze Zahlen. Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren x und $f(x,a,b)$ erzeugt.



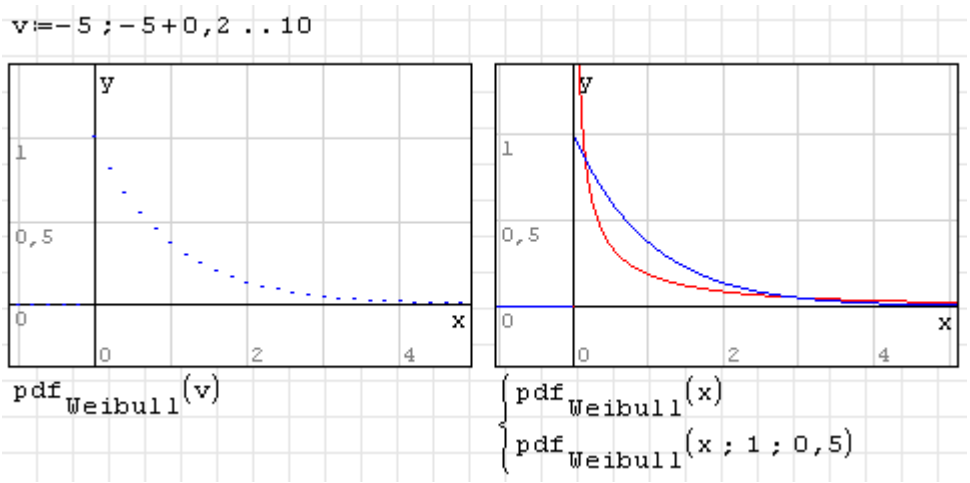
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
pdf.Weibull(t)
pdf.Weibull(t; T; k)
```

Dichtefunktion der Weibullverteilung mit dem Lageparameter T (Voreinstellung $T=1$) und dem Streuungsparameter k (Voreinstellung $k=1$). Es gilt:

$$f(t,T,k) = \frac{k}{T}\left(\frac{t}{T}\right)^{k-1} e^{-(t/T)^k} \quad t \ge 0.$$

Ist x ein Vektor, wird eine plotbare zweispaltige Matrix aus Paaren t und $f(t,T,k)$ erzeugt.

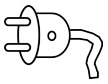


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

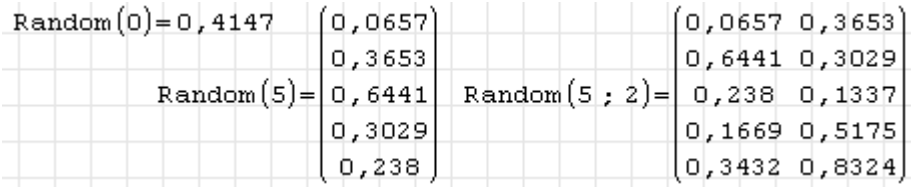
```
perc(Zahl1; Zahl2)
```

Zahl2 Prozent von Zahl1

```
Random(Zeilen)
Random(Zeilen; Spalten)
```

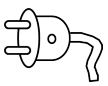


Einzelwert, Vektor oder Matrix gleichverteilter Zufallszahlen zwischen 0 und 1.

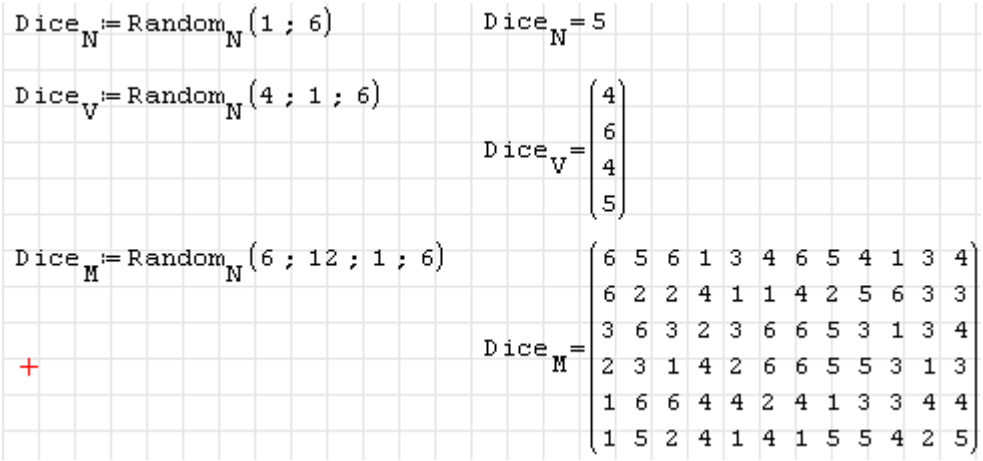


Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Random.N(min; max)
Random.N(Zeilen; min; max)
Random.N(Zeilen; Spalten; min; max)
```



Einzelwert, Vektor oder Matrix gleichverteilter ganzer Zufallszahlen aus dem Intervall min bis max.



Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

```
Random(Zahl)
```

Erzeugt eine ganzzahlige Zufallszahl zwischen Null und `Zahl`. Entspricht der Funktion `Random(0; Zahl)` aus dem Plugin *Statistical Tools*.

```
range(Startwert; Maximalwert)
range(Startwert; Maximalwert; zweiter_Wert)
```

Die Funktion `range()` erzeugt Vektoren mit auf- oder absteigenden Zahlenwerten. Anzugeben sind der erste Wert und der Maximalwert. Optional kann der zweite Wert angegeben werden, dessen Differenz zum Startwert die Schrittweite definiert. Sie ist auf +1 oder -1 voreingestellt, je nach dem, ob der Maximalwert größer oder kleiner als der Endwert ist.

x:rang ⇆ 1 → 5 ↵

$x := 1..5 \quad x = \begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \qquad 2..-2 = \begin{pmatrix}2\\1\\0\\-1\\-2\end{pmatrix} \qquad 0{,}5..3{,}5 = \begin{pmatrix}0{,}5\\1{,}5\\2{,}5\\3{,}5\end{pmatrix}$

Die Schrittweite ist dann die Differenz zwischen Startwert und zweitem Wert.

s:rang ↓ ⇆ 0 → p Strg-G → p Strg-G /4 ↵

$s := 0; \frac{\pi}{4}..2\cdot\pi \qquad s = \begin{pmatrix}0\\0{,}79\\1{,}57\\2{,}36\\3{,}14\\3{,}93\\4{,}71\\5{,}5\\6{,}28\end{pmatrix}$

$x_1 := 0 \quad x_2 := \frac{\pi}{2} \quad x_n := 2\cdot\pi$

$x_1; x_2..x_n = \begin{pmatrix}0\\0{,}5\\1\\1{,}5\\2\end{pmatrix}\pi$

```
rank(M)
```

Liefert den Rang der Matrix `M`. Das ist die Zahl der linear unabhängigen Zeilen in `M`.

$A := \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \quad \mathrm{rank}(A) = 2 \qquad |A| = -1$

Die Werte können auch symbolisch berechnet werden

$$C := \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad |C| = a \cdot d - c \cdot b \qquad \mathrm{rank}(C) = 2$$

$$D := \begin{pmatrix} a & b \\ 2 \cdot a & 2 \cdot b \end{pmatrix} \qquad |D| = 0 \qquad \mathrm{rank}(D) = 1$$

Die Berechnung benötigt die numerisch nicht robust implementierte Determinante `det()`, die bereits bei moderat großen Matrizen und Zahlenwerten versagt.

$$A := \begin{pmatrix} 8 & 0 & 4 & 8 & 8 & 2 & 8 & 3 & 3 & 2 \\ 5 & 3 & 0 & 7 & 4 & 7 & 5 & 4 & 6 & 7 \\ 5 & 8 & 4 & 1 & 8 & 3 & 1 & 8 & 6 & 9 \\ 6 & 3 & 5 & 0 & 0 & 3 & 8 & 9 & 8 & 0 \\ 5 & 8 & 3 & 5 & 8 & 3 & 4 & 6 & 6 & 5 \\ 9 & 6 & 7 & 2 & 8 & 9 & 6 & 3 & 9 & 5 \\ 1 & 9 & 6 & 4 & 5 & 1 & 8 & 0 & 6 & 1 \\ 9 & 5 & 5 & 5 & 1 & 7 & 5 & 8 & 2 & 4 \\ 8 & 4 & 2 & 1 & 9 & 9 & 5 & 2 & 1 & 1 \\ 6 & 5 & 9 & 3 & 9 & 5 & 1 & 8 & 0 & 7 \end{pmatrix}$$

numerische Auswertung:

$\mathrm{rank}(A) =$ ∎

$|A| =$ ∎

symbolische Auswertung:

$\mathrm{rank}(A) = 10$

$|A| = \frac{}{162130935923403000000}$

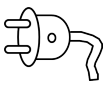
$\mathrm{maple}(|A|) = 784894425$

Bei symbolischer Berechnung können im Einzelfall noch Lösungen erhalten werden, diese sind aber auch durch die interne Zahlendarstellung begrenzt.

Re(z)

Realteil einer komplexen Zahl z. Die Funktion kommt nicht mit Maßeinheiten zurecht.

READ_BLUE()



Einlesen des Blau-Anteils eines Bildes als Grauwertmatrix. Die Anzeige erfolgt hier mit dem Image-Bereich aus dem Plugin *Image Region* (siehe Abschnitt 216), links direkt aus der Datei, rechts aus der eingelesenen Matrix.

```
B := READ_BLUE("WinXPLogo.bmp")
```

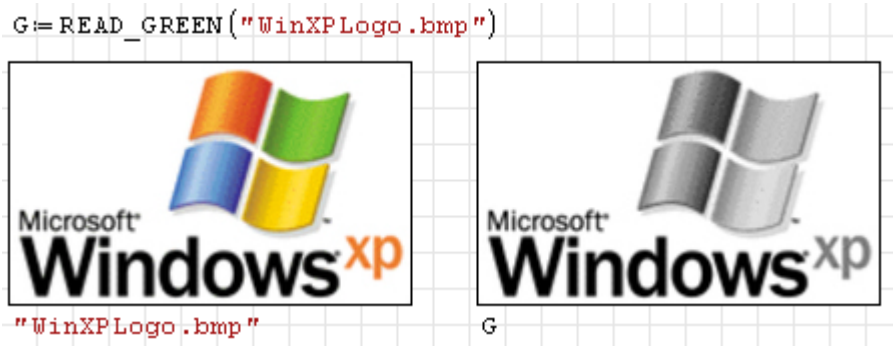
"WinXPLogo.bmp"

B

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

READ_GREEN()

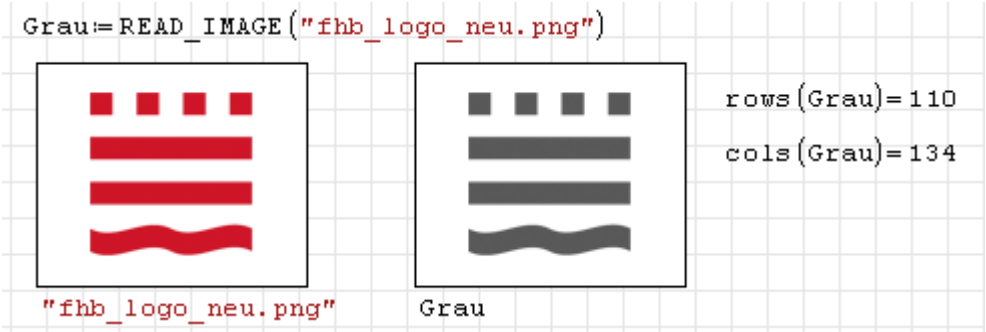
Einlesen des Grün-Anteils eines Bildes als Grauwertmatrix. Die Anzeige erfolgt hier mit dem Image-Bereich aus dem Plugin *Image Region* (siehe Abschnitt 9.4.7), links direkt aus der Datei, rechts aus der eingelesenen Matrix.



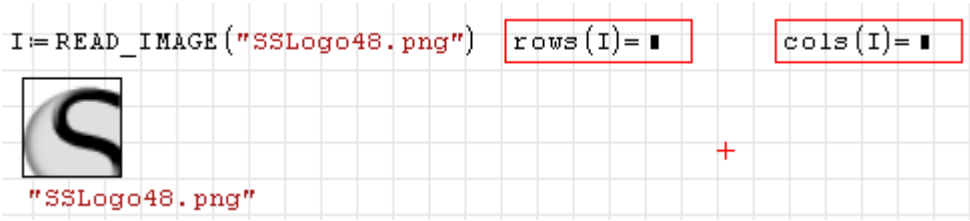
Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

READ_IMAGE(Dateiname)

Einlesen eines Rasterbildes als Grauwertmatrix. Unterstützt werden die gängigen Formate. Die Anzeige erfolgt hier mit dem Image-bereich aus dem Plugin *Image Region* (siehe Abschnitt 9.4.7), links direkt aus der Datei, rechts aus der eingelesenen Matrix.



Bei manchen Bildern kommt es zu Problemen beim Laden. Beispielsweise gelingt es nicht, das SMath-Logo zu laden. Dieses wird auch direkt aus der Datei im Image-Bereich nicht richtig angezeigt.



Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

READ_RED()

Einlesen des Rot-Anteils eines Bildes als Grauwertmatrix. Die Anzeige erfolgt hier mit dem Image-Bereich aus dem Plugin *Image Region* (siehe Abschnitt 9.4.7), links direkt aus der Datei, rechts aus der eingelesenen Matrix.

```
R:=READ_RED("WinXPLogo.bmp")
```

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

`READBIN(Dateiname; Datentyp; Endian; Spalten; Offset; maxZeilen)`

Liest Daten aus Dateien mit Binärzahlen einheitlichen Typs. Der `Dateiname` wird als Zeichenkette angegeben. Der `Datentyp` kann sein:

- `"byte"` (8 Bit Integer ohne Vorzeichen)
- `"int16"` (16 Bit Integer mit Vorzeichen)
- `"uint16"` (16 Bit Integer ohne Vorzeichen)
- `"int32"` (32 Bit Integer mit Vorzeichen)
- `"uint32"` (32 Bit Integer ohne Vorzeichen)
- `"float"` (32 Bit Gleitkommazahl)
- `"double"` (64 Bit Gleitkommazahl)

`Endian` ist die Bitreihenfolge in der Binärdarstellung. Werte können sein:

- 0 - little endian (niedrigstes Bit zuerst, IBM-PC-Standard)
- 1 - big endian (höchstes Bit zuerst, Macintosh-Standard)

`Spalten` gibt die Zahl der Spalten in der Ergebnismatrix an.

`Offset` gibt die Zahl der am Anfang der Datei zu überspringenden Bytes an.

`maxZeilen` gibt die maximal zu lesende Zeilenzahl an.

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

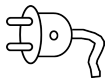
`READBMP()`

Einlesen eines .bmp-Bildes als Grauwertmatrix. Die Anzeige erfolgt hier mit dem Image-Bereich aus dem Plugin *Image Region* (siehe Abschnitt 9.4.7), links direkt aus der Datei, rechts aus der eingelesenen Matrix.

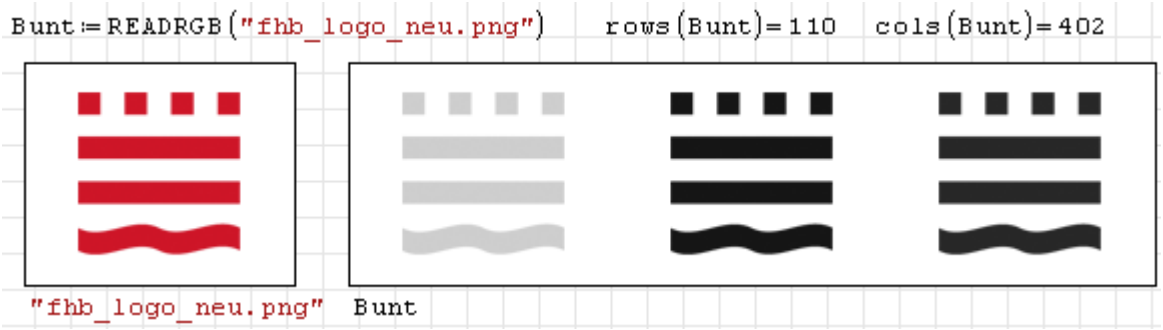
```
Bild:="WinXPLogo.bmp"
I:=READ_IMAGE(Bild)
rows(I)=120   cols(I)=200
```

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

READRGB(Dateiname)



Einlesen von Rasterbildern als RGB-Daten. Die Matrix enthält die Werte für rot, grün und blau nebeneinander angeordnet. Die Anzeige erfolgt hier mit dem Image-Bereich aus dem Plugin *Image Region* (siehe Abschnitt 9.4.7).



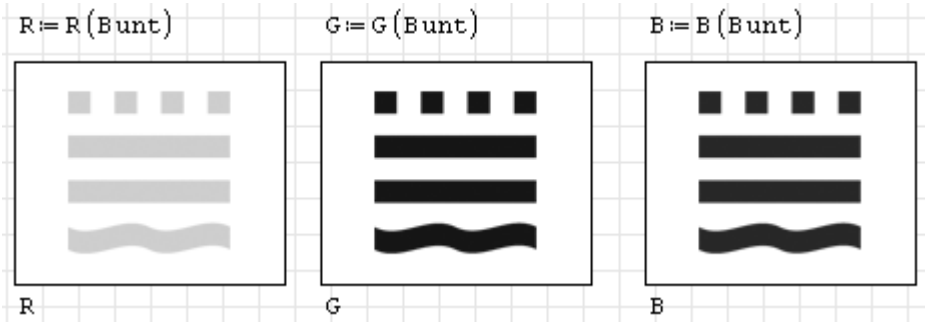
Die folgenden Funktionen trennen die Daten in die Teilmatrizen auf:

$$R(RGB):=\text{eval}\left(\text{submatrix}\left(RGB;1;\text{rows}(RGB);1;\frac{\text{cols}(RGB)}{3}\right)\right)$$

$$G(RGB):=\text{eval}\left(\text{submatrix}\left(RGB;1;\text{rows}(RGB);\frac{\text{cols}(RGB)}{3}+1;2\cdot\frac{\text{cols}(RGB)}{3}\right)\right)$$

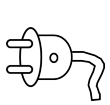
$$B(RGB):=\text{eval}\left(\text{submatrix}\left(RGB;1;\text{rows}(RGB);2\cdot\frac{\text{cols}(RGB)}{3}+1;\text{cols}(RGB)\right)\right)$$

Diese können dann separat angezeigt oder weiterverarbeitet werden.

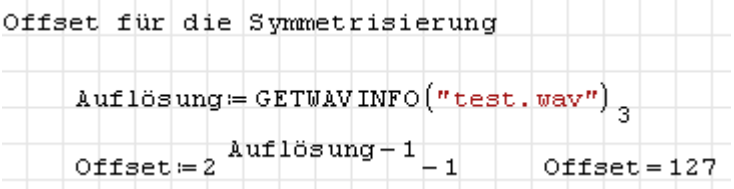


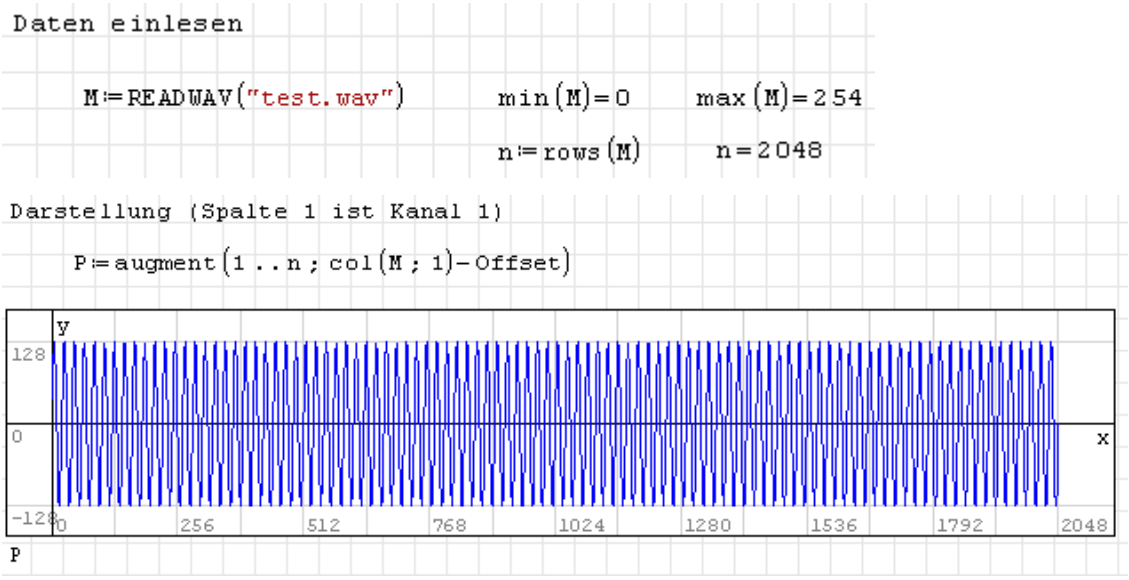
Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

READWAV()



Lesen von .wav-Dateien. Abtastrate und Auflösung können mit der Funktion GETWAVEINFO() bestimmt werden. Die zurückgegebene Matrix hat die Dimension Wertezahl×Kanalzahl.

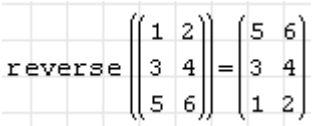




Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

`reverse(Matrix)`

Kehrt die Reihenfolge der Zeilen in einer Matrix um.



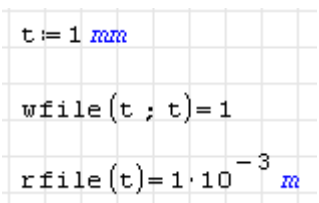
`rfile(Name)`

Liest den Inhalt aus einer internen Austauschdatei, die mit `wfile()` (siehe Seite 247) geschrieben wurde. Beachten Sie, dass das Argument „Name" buchstäblich genommen wird. Es wird nicht ausgewertet und kann keinen Pfad oder Endung enthalten.

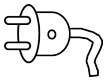
Die Datei wird in einem speziellen Systemverzeichnis (unter XP: *"C:\Documents and Settings\%USERNAME%\Application Data\SMath\user"*, unter Windows 7: *C:\Users\$Benutzer$\AppData\Roaming\SMath\user*) gesucht. Sie können sich dieses Verzeichnis mit der Funktion `SettingsDirectory()` (im Plugin *Custom Functions*, siehe Seite 9.4.3) anzeigen lassen.

Die Funktion ist nicht für den Datenexport oder sonstigen Austausch, sondern nur zur internen Zwischenspeicherung von Daten gedacht und geeignet.

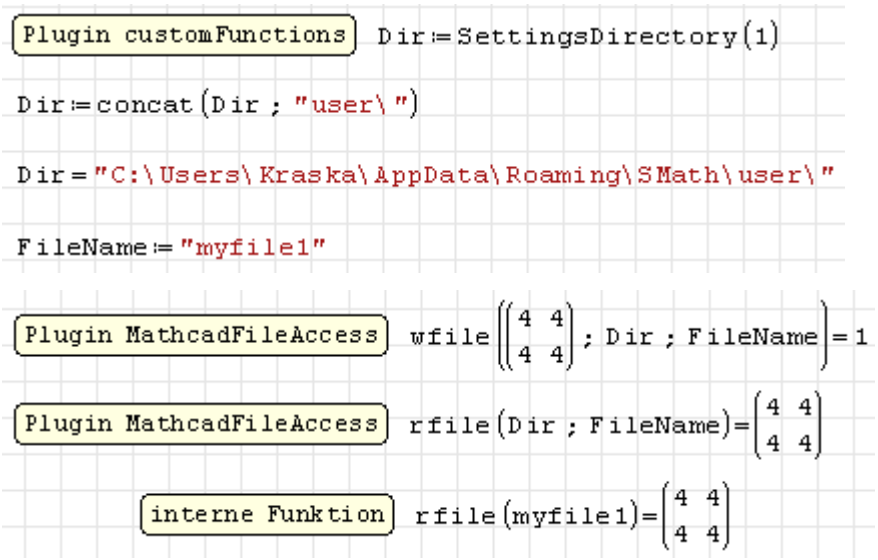
Das folgende Beispiel erzeugt eine Datei namens *t* mit dem Inhalt `'m/1000` und liest sie wieder ein.



```
rfile(Verzeichnis; Dateiname)
```

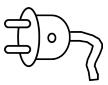


Diese Funktion ist eine flexiblere Variante der eingebauten Funktion `rfile()`, siehe voriger Eintrag. Sie liest einen Ausdruck aus der Datei names `Dateiname` (hier als Zeichenkette anzugeben) im ebenfalls als Zeichenkette anzugebenden `Verzeichnis`.



Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

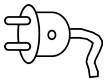
```
Ridder(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten; maxIter)
Ridder(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten; maxIter;
Schalter)
```



Nichtlinearer Löser nach dem Ridder-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen mit den gegebenen Startwerten. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Iterationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Iterationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

```
Rkadapt(y0; x0; x1; nstep; D(x,y))
```



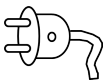
Integration gewöhnlicher Differenzialgleichungssysteme mit dem adaptiven Runge-Kutta-Fehlberg-Verfahren. `D(x,y)` enthält die Ableitungen der Unbekannten nach der unabhängigen Variable und ist eine vektorwertige Funktion der unabhängigen Variable x (`x`) und des n-dimensionalen Unbekanntenvektors $\underline{y}$ (`y`). `y0` ist der Anfangswert $\underline{y}_0 = \underline{y}(x = x_0)$ des Unbekanntenvektors. Die unabhängige Variable durchläuft den Bereich von x_0 (`x0`) bis x_1 (`x1`) in n_{step} (`nstep`) Schritten.

Ergebnis ist eine Matrix aus $1 + n_{\text{step}}$ Zeilen und $1 + n$ Spalten. Die erste Spalte enthält die x-Werte, die weiteren Spalten jeweils die Unbekannten y_1 bis y_n. Die Zeilen enthalten die Werte für die $1 + n_{\text{step}}$ Werte der unabhängigen Variable.

Obwohl der Algorithmus intern mit einer adaptiven Zeitschrittsteuerung arbeitet, werden genau $1 + n_{\text{step}}$ Datensätze ausgegeben, die hinsichtlich der unabhängigen Variable äquidistant sind

Die Funktion ist Bestandteil des Plugins *ODE Solvers* (siehe Abschnitt 9.4.11).

```
rkfixed(y0; x0; x1; nstep; D(x,y))
```



Integration gewöhnlicher Differenzialgleichungssysteme mit dem Runge-Kutta-Verfahren vierter Ordnung bei fester Schrittweite. `D(x,y)` enthält die Ableitungen der Unbekannten nach der unabhängigen Variable und ist eine vektorwertige Funktion der unabhängigen Variable x `(x)` und des n-dimensionalen Unbekanntenvektors $\underline{y}$ `(y)`. `y0` ist der Anfangswert $\underline{y}_0 = \underline{y}(x = x_0)$des Unbekanntenvektors. Die unabhängige Variable durchläuft den Bereich von x_0 `(x0)` bis x_1 `(x1)` in n_{step} `(nstep)` Schritten.

Ergebnis ist eine Matrix aus $1 + n_{\text{step}}$ Zeilen und $1 + n$ Spalten. Die erste Spalte enthält die x-Werte, die weiteren Spalten jeweils die Unbekannten y_1 bis y_n. Die Zeilen enthalten die Werte für die $1 + n_{\text{step}}$ Werte der unabhängigen Variable.

Die Funktion ist Bestandteil des Plugins *ODE Solvers* (siehe Abschnitt 9.4.11).

```
row(Matrix; i)
```

liefert die Zeile i der Matrix.

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad \text{row}(A\,;\,3) = (5\ \ 6)$$

```
rows(Matrix)
```

Liefert die Zahl der Zeilen der Matrix.

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad \text{rows}(A) = 3$$

```
rsort(Name)
```

Sortiert die Spalten einer Matrix anhand der i-ten Zeile.

$$\text{rsort}\left(\begin{pmatrix} 1 & 3 & 1 & 3 & 5 \\ 2 & 4 & 7 & 8 & 6 \end{pmatrix};\ 1\right) = \begin{pmatrix} 1 & 1 & 3 & 3 & 5 \\ 2 & 7 & 4 & 8 & 6 \end{pmatrix}$$

```
Secant(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten; maxIter)
Secant(Funktionen; Untergrenzen; Obergrenzen; Genauigkeiten; maxIter;
Schalter)
```

Nichtlinearer Löser nach dem Sekanten-Verfahren. Sucht die Nullstellen einer Liste von Funktionen für alle darin vorkommenden undefinierten Variablen mit den gegebenen Startwerten. Der Vektor Genauigkeiten enthält eine Toleranz für jede der gegebenen Funktionen. Es werden maximal maxIter Iterationen durchgeführt. Die Funktion liefert einen Vektor mit Lösungsvektoren (falls diese existieren). Wird der optionale Schalter ungleich Null angegeben, wird auch die benötigte Zahl der Iterationen angegeben.

Die Funktion ist Bestandteil des Plugins *Non Linear Solvers* (siehe Abschnitt 9.4.10).

`SettingsDirectory()`

Liefert das Verzeichnis, in welchem SMath benutzerspezifische Daten ablegt. Dieses Verzeichnis wird zum Beispiel von den Funktionen `wfile()` (Seite 329), `rfile()` (Seite 329) oder `dfile()` (Seite 269) benutzt.

```
SettingsDirectory(■)="C:\Users\Kraska\AppData\Roaming\SMath\"
SettingsDirectory(0)="C:\Users\Kraska\AppData\Roaming\SMath\"
```

Das Argument kann weggelassen werden und hat keinen Einfluss auf das Ergebnis.

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`Skewness(y)`

Schiefe β_2 einer durch den Vektor $\underline{y}$ gegebenen Datenreihe.

$$v(y) = \frac{\mu_3(y)}{\mu_2(y)^{\frac{3}{2}}}$$

mit den k-ten zentralen Momenten μ_k.

$y := (3\ 5\ 5{,}5\ 7\ 6\ 9)^T$ Skewness(y) = 0,1329

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`Slope(x; y)`

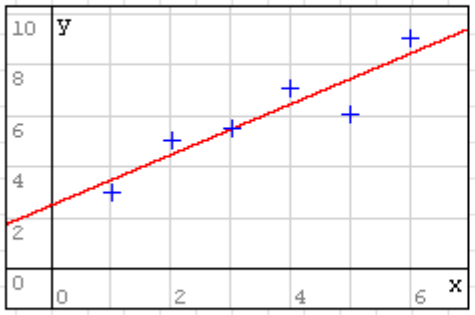
Neigung der Regressionsgerade durch die mit den Vektoren $\underline{x}$ und $\underline{y}$ gegebenen Datenpunkte.

$$x := \begin{pmatrix}1\\2\\3\\4\\5\\6\end{pmatrix} \quad y := \begin{pmatrix}3\\5\\5{,}5\\7\\6\\9\end{pmatrix}$$

$\alpha := \text{Intercept}(x ; y)$ $\alpha = 2{,}4667$

$\beta := \text{Slope}(x ; y)$ $\beta = 0{,}9857$

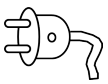
```
Data:=Zeros(1;5)    for i∈1..length(x)
                    Data:=stack(Data;(x_i y_i "+" 10 "blue"))
```




```
{Data
{α+β·x
```

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`Solve(Gl,var)`



Löst das als Vektor oder Liste von boolschen Gleichungen `Gl` gegebene Gleichungssystem nach den als Vektor oder Liste `var` gegebenen Variablen auf. Die Variablen müssen undefiniert sein.

Rückgabewert ist eine Zeilenmatrix von Listen. Jede Liste enthält die Variablenwerte einer Lösung, ausgedrückt als boolsche Gleichungen `Name=Wert`. Die Werte können mit `Assign()` zugewiesen werden. Die Funktion ruft `solve()` in Maxima auf.

$$Gl_1 := \left(x^2 + 3\cdot x\cdot y + y^2 = 0\right) \qquad Gl_2 := (3\cdot x + y = 1)$$

$$L := \mathrm{Solve}\left(Gl, \begin{pmatrix} x \\ y \end{pmatrix}\right) \qquad L = \left\{\left\{\left(x = \left(-\frac{-3+\sqrt{5}}{2}\right)\right)\left(y = \frac{-7+3\cdot\sqrt{5}}{2}\right)\right\}\left\{\left(x = \frac{3+\sqrt{5}}{2}\right)\left(y = \left(-\frac{7+3\cdot\sqrt{5}}{2}\right)\right)\right\}\right\}$$

$$\mathrm{Assign}\left(L_1\right) = \begin{cases} -\frac{-3+\sqrt{5}}{2} \\ \frac{-7+3\cdot\sqrt{5}}{2} \end{cases} \qquad x = -\frac{-3+\sqrt{5}}{2} \qquad y = \frac{-7+3\cdot\sqrt{5}}{2} \qquad \mathrm{Clear}(x, y) = 1$$

$$\mathrm{Assign}\left(L_1\right) = \begin{cases} -\frac{-3+\sqrt{5}}{2} \\ \frac{-7+3\cdot\sqrt{5}}{2} \end{cases} \qquad x = -\frac{-3+\sqrt{5}}{2} \qquad y = \frac{-7+3\cdot\sqrt{5}}{2} \qquad \mathrm{Clear}(x, y) = 1$$

Ist nur eine Gleichung nach einer Variable aufzulösen, müssen die Argumente nicht als Vektor oder Liste gegeben werden. Der Rückgabewert ist dann ein Skalar oder bei mehreren Lösungen eine Liste.

$$\mathrm{Solve}(a\cdot x + b = y, a) = a = \frac{y-b}{x}$$

$$L := \left(\text{Solve}\left(x^3 - 3\cdot x^2 + 5\cdot x = 15,\ x\right)\right) \qquad L = \begin{cases} (x = (-i\cdot\sqrt{5})) \\ (x = i\cdot\sqrt{5}) \\ (x = 3) \end{cases}$$

Ausführlich wird auf die Funktion in Abschnitt 4.13.1 eingegangen.

Die Funktion ist Bestandteil des Plugins *Maxima* (siehe Abschnitt **??**).

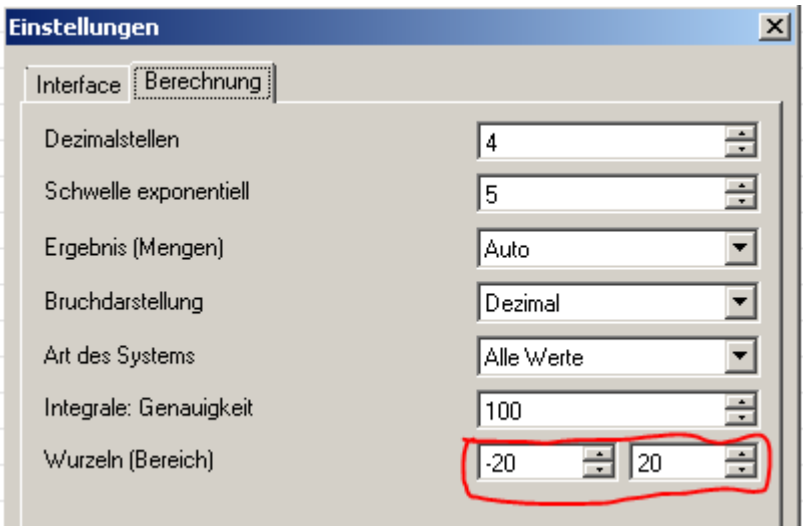
```
solve(expr; x)
solve(expr; x; a; b)
```

Nichtlinearer Löser. Sucht alle Lösungen für den von der skalaren Variablen `x` abhängigen skalaren Ausdruck `expr` im angegebenen Intervall zwischen `a` und `b`.

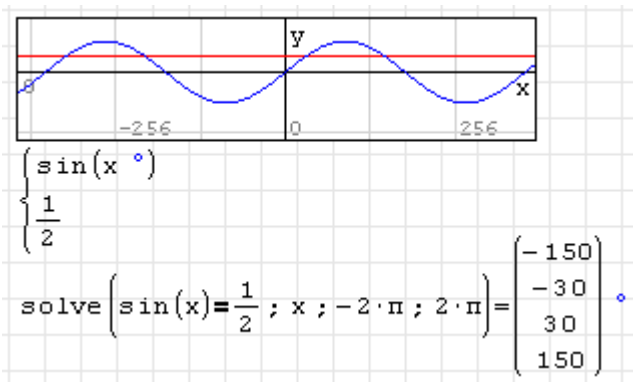
Die Variable `x` und die Intervallgrenzen müssen dimensionslos sein.

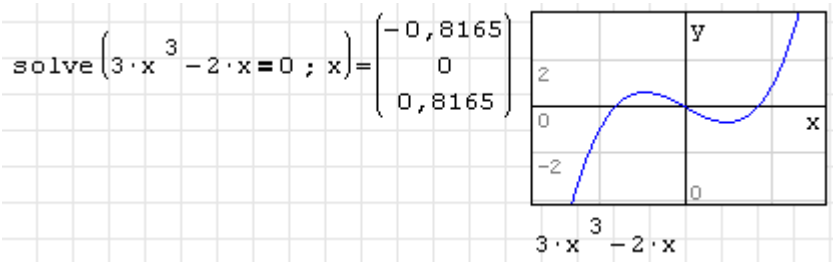
Ist der Suchbereich nicht angegeben, dann wird im voreingestellten Intervall $-20\ldots20$ gesucht. Diese Einstellung kann über das Hauptmenü geändert werden:

Extras> Einstellungen> Berechnung> Wurzeln (Bereich)



Beispiele:

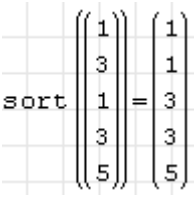




Beachten Sie auch den Abschnitt 4.13.1 zur Gleichungslösung.

```
sort(Spaltenmatrix)
```

Sortiert die Elemente einer Spaltenmatrix (eines Vektors) der Größe nach (aufsteigend).

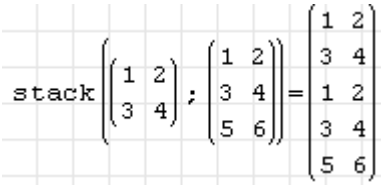



```
sqrt(Zahl)
```

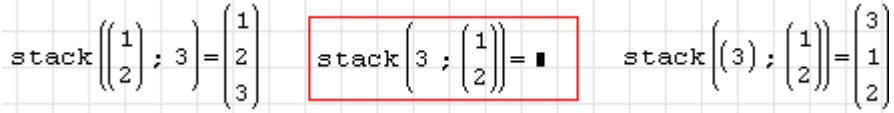
Quadratwurzel der Zahl

```
stack(Matrix1; Matrix2)
stack(Spaltenmatrix; Skalar)
```

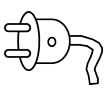
Stapelt zwei gegebene Matrizen gleicher Spaltenzahl.



An Vektoren können einzelne Zahlen (auch andere skalare Ausdrücke wie Zeichenketten) angehängt werden, ohne dass diese als 1×1-Matrizen angegeben werden müssen. Voranstellen kann man die Zahl auf diese Weise aber nicht.




```
StdDev(Daten)
```



Liefert die Standardabweichung der Daten (Matrix, Vektor oder Liste):

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2}$$

$$\mathrm{StdDev}\left(\begin{pmatrix}8\\2\\1\end{pmatrix}\right)=3{,}7859 \qquad \mathrm{StdDev}\left(\begin{pmatrix}1&3&5&1&6&2\\3&2&5&2&3&5\\5&2&1&3&4&1\end{pmatrix}\right)=1{,}645$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12). Hinweis: in der ersten Version des Plugins (November 2012) wurde diese Funktion `stdDev()` geschrieben (kleiner Anfangsbuchstabe).

`str2num(Zeichenkette)`

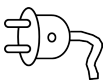
Wandelt die als Zeichenkette gegebene interne Textdarstellung eines Ausdrucks in den Ausdruck selbst um.

$$\mathrm{num2str}\left(\begin{Bmatrix}\text{"abc"}\\\sqrt{2}\end{Bmatrix}\right)=\text{"sys("abc",sqrt(2),2,1)"}$$

$$\mathrm{str2num}\left(\mathrm{num2str}\left(\begin{Bmatrix}\text{"abc"}\\\sqrt{2}\end{Bmatrix}\right)\right)=\begin{Bmatrix}\text{"abc"}\\\sqrt{2}\end{Bmatrix} \qquad \text{Optimierung symbolisch}$$

$$\mathrm{str2num}(\text{"sqrt(x^2+y^2)"})=\sqrt{x^2+y^2} \qquad \text{Optimierung symbolisch}$$

`strjoin(d; s)`



Verbindet alle Zeichenketten in der Variable `s` (beliebig geschachtelte Struktur aus Listen und Matrizen) unter Einschiebung des Trennzeichens `d`

$$\mathrm{strjoin}\left(\text{" "};\begin{pmatrix}\text{"hello"}\\\text{"world!"}\end{pmatrix}\right)=\text{"hello world!"}$$

$$\mathrm{strjoin}\left(\text{" "};\begin{Bmatrix}\text{"hello"}\\\text{"world!"}\end{Bmatrix}\right)=\text{"hello world!"}$$

$$\mathrm{strjoin}\left(\text{", "};\begin{Bmatrix}\text{"bread"}\\\text{"milk"}\\\text{"sugar"}\\\text{"..."}\end{Bmatrix}\right)=\text{"bread, milk, sugar, ..."}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`strlen(Zeichenkette)`

Liefert die Länge einer Zeichenkette. Kann keine Listen oder Matrizen von Zeichenketten verarbeiten.

$$\mathrm{strlen}(\text{"12345"})=5$$

$$\mathrm{strlen}\left(\begin{Bmatrix}\text{"12"}\\\text{"123"}\end{Bmatrix}\right)=\blacksquare\,\blacksquare$$

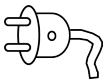
Argument muss ein String sein.

strrep(Ziel; Suchmuster; Ersatz)

Ersetzt alle Vorkommen von Suchmuster in Ziel durch Ersatz, alle Argumente sind Zeichenketten.

```
strrep("abcdefghijkdef"; "def"; "123")="abc123ghijk123"
strrep("abcdefghijkdef"; "def"; "")="abcghijk"
```

strsplit(s; d)

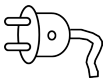


Teilt die Zeichenkette s anhand der Trennkette d. Liefert einen Vektor der Teilketten.

```
strsplit("the QUICK bRoWn FoX..."; " ")=⎛"the"  ⎞
                                        ⎜"QUICK"⎟
                                        ⎜"bRoWn"⎟
                                        ⎝"FoX..."⎠
strsplit("the QUICK bRoWn FoX..."; "o")=⎛"the QUICK bR"⎞
                                        ⎜   "Wn F"     ⎟
                                        ⎝   "X..."     ⎠
strsplit("the QUICK bRoWn FoX..."; "Ro")=⎛"the QUICK b"⎞
                                         ⎝ "Wn FoX..." ⎠
strsplit("the QUICK bRoWn FoX..."; "x")=("the QUICK bRoWn FoX...")
```

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

strtolower(Text)

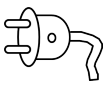


Wandelt Text (Zeichenketten) in Kleinbuchstaben um. Bei verschachtelten Strukturen (Matrizen, Listen) werden alle Zeichenketten umgewandelt.

```
strtolower("the QUICK bRoWn FoX...")="the quick brown fox..."

strtolower(⎛"the QUICK bRoWn FoX..."⎞)=⎛"the quick brown fox..."⎞
           ⎜        √2                ⎟  ⎜        √2            ⎟
           ⎝   "another sTRINg"       ⎠  ⎝   "another string"   ⎠
```

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

strtoupper(Text)



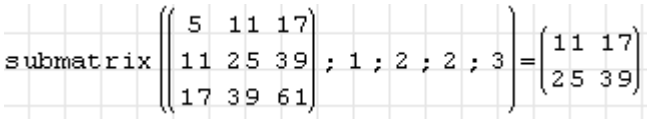
Wandelt Text (Zeichenketten) in Großbuchstaben um. Bei verschachtelten Strukturen (Matrizen, Listen) werden alle Zeichenketten umgewandelt.

```
strtoupper("the QUICK bRoWn FoX...")="THE QUICK BROWN FOX..."

strtoupper(⎛"the QUICK bRoWn FoX..."⎞)=⎛"THE QUICK BROWN FOX..."⎞
           ⎜        √2                ⎟  ⎜        √2            ⎟
           ⎝   "another sTRINg"       ⎠  ⎝   "ANOTHER STRING"   ⎠
```

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

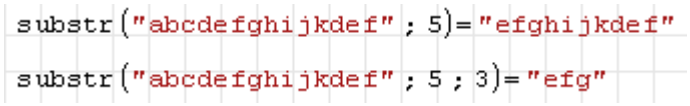
```
submatrix(Matrix; r1; r2; c1; c2)
```

Liefert die Untermatrix von Zeilen `r1` bis `r2` und Spalten `c1` bis `c2`.



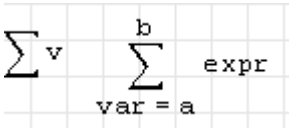

```
substr(Zeichenkette; Start)
substr(Zeichenkette; Start; Länge)
```

Liefert ein Teilstück der Zeichenkette beginnend mit der Position Start. Optional kann eine Länge vorgegeben werden, Voreinstellung: alles ab Startposition.

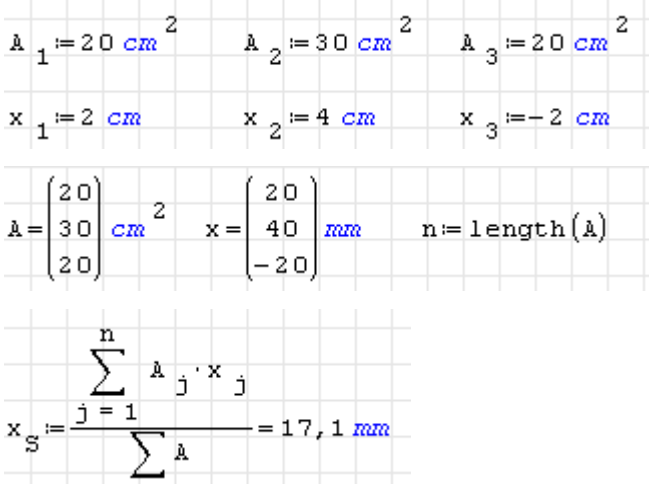



```
sum(v)
sum(expr; j; a; b)
```

Bildet die Summe über alle Elemente des Vektors `v` oder die Summe über alle Werte des Ausdrucks `expr`, wenn die Variable `j` alle ganzzahligen Werte von `a` bis `b` durchläuft. Die entsprechenden Operatordarstellungen sind:



Beispiel:



Die Funktion kann durch das Maxima-Plugin übernommen werden. Dann sind auch symbolische Summenberechnungen möglich .

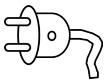
```
MaximaTakeover("sum()")="sum() now use Maxima"
```

$$\sum_{i=1}^{n} i = 3 \cdot i$$

sys()

Listenobjekt (in SMath als Gleichungssystem bezeichnet). Wird benötigt für die Anzeige mehrerer Grafikobjekte in Plots und (bezeichnungsgemäß) für die Darstellung von Gleichungssystemen. Funktionen für reelle oder komplexe Zahlen können auch Listen verarbeiten. Die Verwendung von Listen wird in Abschnitt 4.10 erläutert.

sys2mat(Liste)



Wandelt eine gegebene Liste in eine Matrix um.

$$\text{sys2mat}\left(\left\{\begin{matrix}1\\2\\3\\4\end{matrix}\right\}\right) = \begin{pmatrix}1\\2\\3\\4\end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

time(dummy)

Das ist eine inoffizielle Funktion! Sie ist nicht für den Standardnutzer gedacht und kann in künftigen Versionen wieder entfallen oder geändert werden.

Die Funktion liefert die Zeit seit dem 1.1.1601 bis zum Zeitpunkt der Funktionsauswertung in Sekunden. Das Argument `dummy` ist irrelevant.

Mehrmalige Auswertung liefert entsprechend steigende Werte.

$$time(1) = 1{,}29981009265920 \cdot 10^{10}\ s$$
$$time(1) = 1{,}29981009266030 \cdot 10^{10}\ s$$

Damit sind Zeitmessungen während der Ausführung von Berechnungen möglich:

$$t_0 := time(1)$$
$$20^{20} = 1{,}05 \cdot 10^{26}$$
$$\Delta t := time(1) - t_0$$
$$\Delta t = 3\ ms$$

tr(Matrix)

Spur einer Matrix (Summe der Diagonalelemente)

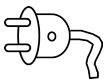
$$\mathrm{tr}\left(\begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{pmatrix}\right) = 91$$

`transpose(Matrix)`

Transponierte Matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^{T} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

`ucfirst(Text)`



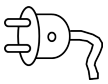
Wandelt den ersten Buchstaben von Zeichenketten in Großbuchstaben um. Bei verschachtelten Strukturen (Matrizen, Listen) werden alle Zeichenketten umgewandelt.

$$\mathrm{ucfirst}("\text{the QUICK bRoWn FoX...}") = "\text{The QUICK bRoWn FoX...}"$$

$$\mathrm{ucfirst}\left(\begin{pmatrix} "\text{The QUICK bRoWn FoX...}" \\ \sqrt{2} \\ "\text{another sTRINg}" \end{pmatrix}\right) = \begin{pmatrix} "\text{The QUICK bRoWn FoX...}" \\ \sqrt{2} \\ "\text{Another sTRINg}" \end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`ucwords(Text)`



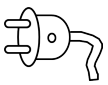
Wandelt Anfangsbuchstaben aller Wörter in Zeichenketten in Großbuchstaben um. Bei verschachtelten Strukturen (Matrizen, Listen) werden alle Zeichenketten umgewandelt.

$$\mathrm{ucwords}("\text{the QUICK bRoWn FoX...}") = "\text{The QUICK BRoWn FoX...}"$$

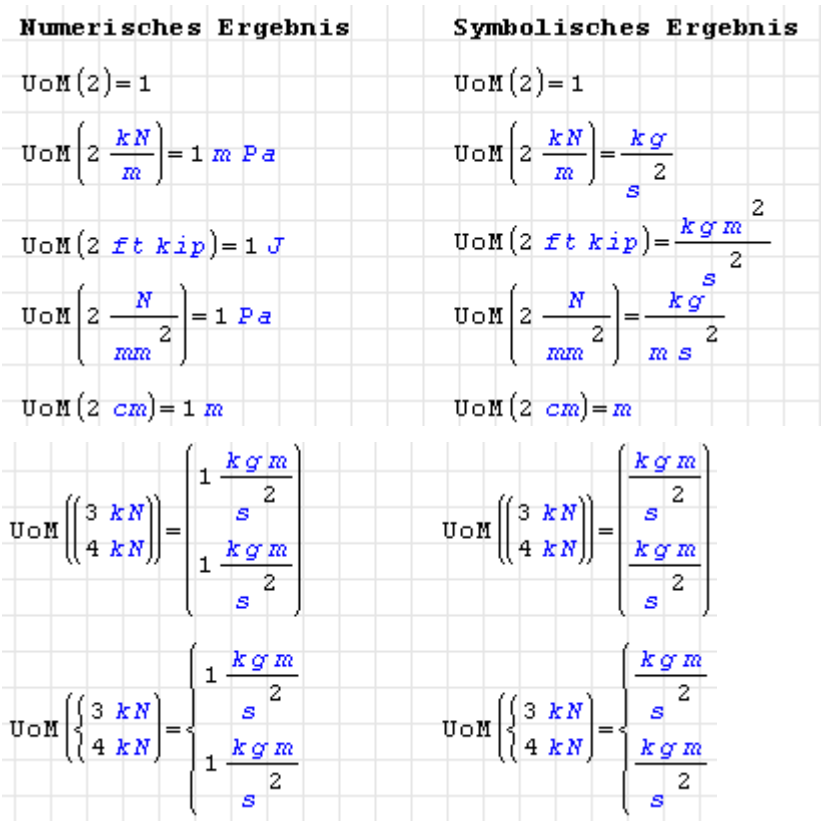
$$\mathrm{ucwords}\left(\begin{pmatrix} "\text{The QUICK bRoWn FoX...}" \\ \sqrt{2} \\ "\text{another sTRINg}" \end{pmatrix}\right) = \begin{pmatrix} "\text{The QUICK BRoWn FoX...}" \\ \sqrt{2} \\ "\text{Another STRINg}" \end{pmatrix}$$

Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`UoM(Ausdruck)`



Liefert die zur Dimension des Ausdrucks passende Basiseinheit. Bei Matrizen oder Listen erfolgt dies elementweise.

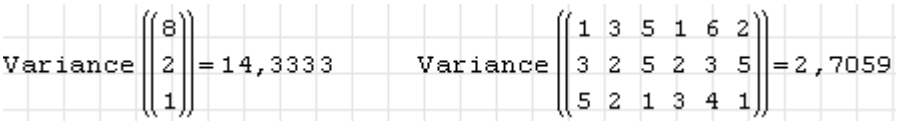


Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

`Variance(Daten)`

Liefert die Varianz der Daten (Matrix, Vektor, Liste):

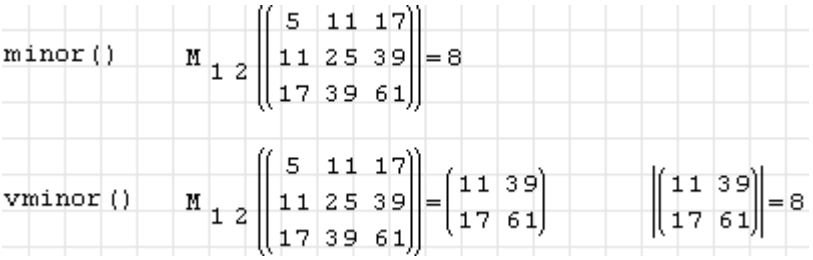
$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2$$



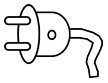
Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`vminor(Matrix; i; j)`

Untermatrix der Matrix, die nach Streichen der Zeile i und der Spalte j übrigbleibt. Die Operatordarstellung ist die gleiche, wie bei der Funktion minor(), die die Determinante der Submatrix liefert.



`WeightedMean(Daten; Gewicht)`



Liefert den gewichteten Mittelwert der Daten (Matrix, Vektor, Liste). Das Argument „Gewicht" muss die gleiche Form wie „Daten" haben.

$$m(x,w) = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w}$$

$$\mathrm{WeightedMean}\left(\begin{pmatrix}8\\2\\1\end{pmatrix};\begin{pmatrix}0,5\\1\\1\end{pmatrix}\right)=2,8$$

Die Funktion ist Bestandteil des Plugins *Statistical Tools* (siehe Abschnitt 9.4.12).

`wfile(Ausdruck; Name)`

Schreibt den Ausdruck in eine Datei *Name*. Beachten Sie, dass das Argument „Name" buchstäblich genommen wird. Es wird nicht ausgewertet und kann keinen Pfad oder Endung enthalten.

Die Datei wird in ein Systemverzeichnis geschrieben (unter XP: *"C:\Documents and Settings\%USERNAME%\Application Data\SMath\user"*, unter Windows 7: *C:\Users\$Benutzer$\AppData\Roaming\SMath\user*). Sie können sich dieses Verzeichnis mit der Funktion `SettingsDirectory()` (im Plugin *Custom Functions*, siehe Seite 9.4.3) anzeigen lassen.

Die Datei enthält den Wert des Ausdrucks in interner Text-Darstellung, wie sie auch in der Zwischenablage benutzt wird.

Die Funktion ist nicht für den Datenexport oder sonstigen Austausch, sondern nur zur internen Zwischenspeicherung von Daten gedacht und geeignet.

Beachten Sie, dass die Funktion `wfile()` nur ausgeführt wird, wenn ihr Wert angezeigt oder einer Variablen zugewiesen wird.

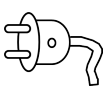
Das folgende Beispiel erzeugt eine Datei namens *t* mit dem Inhalt `'m/1000`

$$t := 1\ mm$$

$$\mathrm{wfile}(t;t)=1$$

$$\mathrm{rfile}(t)=1\cdot 10^{-3}\ m$$

`wfile(Ausdruck; Verzeichnis; Dateiname)`



Diese Funktion ist eine flexiblere Variante der eingebauten Funktion `wfile()`, siehe voriger Eintrag. Sie schreibt den `Ausdruck` in eine Datei names `Dateiname` (hier als Zeichenkette anzugeben) ins ebenfalls als Zeichenkette anzugebende `Verzeichnis`.

Die Datei enthält den Wert des Ausdrucks in interner Text-Darstellung, wie sie auch in der Zwischenablage benutzt wird.

Beachten Sie, dass die Funktion `wfile()` nur ausgeführt wird, wenn ihr Wert angezeigt oder einer Variablen zugewiesen wird.

Plugin customFunctions Dir := SettingsDirectory(1)

Dir := concat(Dir ; "user\")

Dir = "C:\Users\Kraska\AppData\Roaming\SMath\user\"

FileName := "myfile1"

Plugin MathcadFileAccess wfile($\begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$; Dir ; FileName) = 1

Plugin MathcadFileAccess rfile(Dir ; FileName) = $\begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$

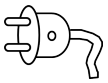
interne Funktion rfile(myfile1) = $\begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

`while(Bedingung; Ausdruck)`

Kopfgesteuerte Schleife, Ausdruck wird ausgeführt, solange die Bedingung wahr ist.

`WRITEBIN(Dateiname; Datentyp; Endian; Matrix)`



Schreiben von Binärdaten. Die Daten in `Matrix` werden in die Datei `Dateiname` als `Datentyp` unter Beachtung von `Endian` geschrieben. Rückgabewert ist die Dateigröße in Byte. Zu den Werten von Datentyp und Endian siehe auch die zugehörige Lesefunktion READBIN(), Seite 327.

Datei := "data1.bin"

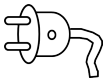
M := $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ n := cols(M) n = 2

Endian := 0

bytes := WRITEBIN(Datei ; "byte" ; Endian ; M) bytes = 6

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

`WRITEWAV(Dateiname; Abtastrate; Auflösung; Matrix)`



Schreibt die Matrix in die angegebene Datei, dabei werden die Werte gemäß Auflösung diskretisiert. Die Abspielgeschwindigkeit wird über die Abtastrate (Werte pro Sekunde) festgelegt. Zurückgegeben wird die Dateigröße in Byte.

N := 2048 SampleRate := 8000 Resolution := 8

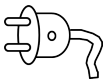
for ii ∈ 1 .. N

$M_{ii} := 127 \cdot \left(\sin\left(2 \cdot \pi \cdot 100 \cdot \frac{ii}{N}\right) + 1\right)$

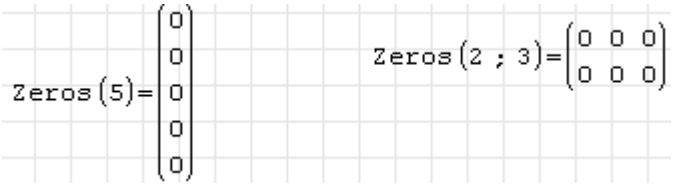
WRITEWAV("test.wav" ; SampleRate ; Resolution ; M) = 2092

Die Funktion ist Bestandteil des Plugins *Mathcad File Access* (siehe Abschnitt 9.4.9).

```
Zeros(Zeilen)
Zeros(Zeilen; Spalten)
```



Erzeugt eine mit Nullen belegte Matrix mit der angegebenen Zeilen- und Spaltenzahl. Fehlt die Spaltenzahl, wird ein Vektor erzeugt.



Die Funktion ist Bestandteil des Plugins *Custom Functions* (siehe Abschnitt 9.4.3).

E. Portables SMath mit Maxima

Hier wird beschrieben, wie eine portable Distribution von SMath Studio zusammengestellt werden kann. Diese lässt sich auf einem USB-Speicher installieren und an jedem Windows-PC benutzen. Maxima wird dabei als Unterverzeichnis in das SMath-Installationsverzeichnis plaziert.

E.1. Portables SMath Studio

Normalerweise speichert SMath Erweiterungen aus der Online-Galerie und die Programmeinstellungen in einem Windows-Systemverzeichnis. Man kann dieses mit der Funktion `SettingsDirectory()` ermitteln. Bei einer Standardinstallation kann man beispielsweise folgende Ausgabe bekommen:

```
SettingsDirectory(▮)="C:\Users\Kraska\AppData\Roaming\SMath\"
```

Stellt man in das Installationsverzeichnis von SMath Studio eine leere Datei des Namens *portable.version* ein, so ist das Installationsverzeichnis gleichzeitig das Einstellungsverzeichnis (settings directory).

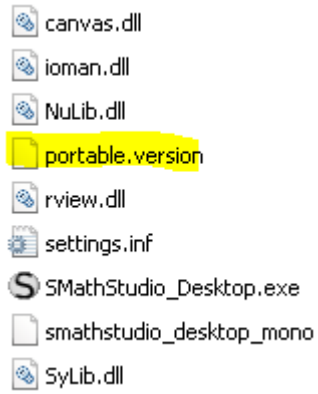
Eine portable SMath-Version kann man also auf folgende Weise erzeugen:

1. Herunterladen von der offiziellen SMath-Studio-Seite. Die aktuelle offizielle Version findet man unter **SMath Studio Forum> Download SMath Studio**. Dort lohnt eventuell ein Blick in das Unterforum **Nightly builds**, welches Entwicklungsversionen enthält, die häufig für die praktische Arbeit stabil genug sind und die meist auch von den aktuellen Plugins unterstützt wird.

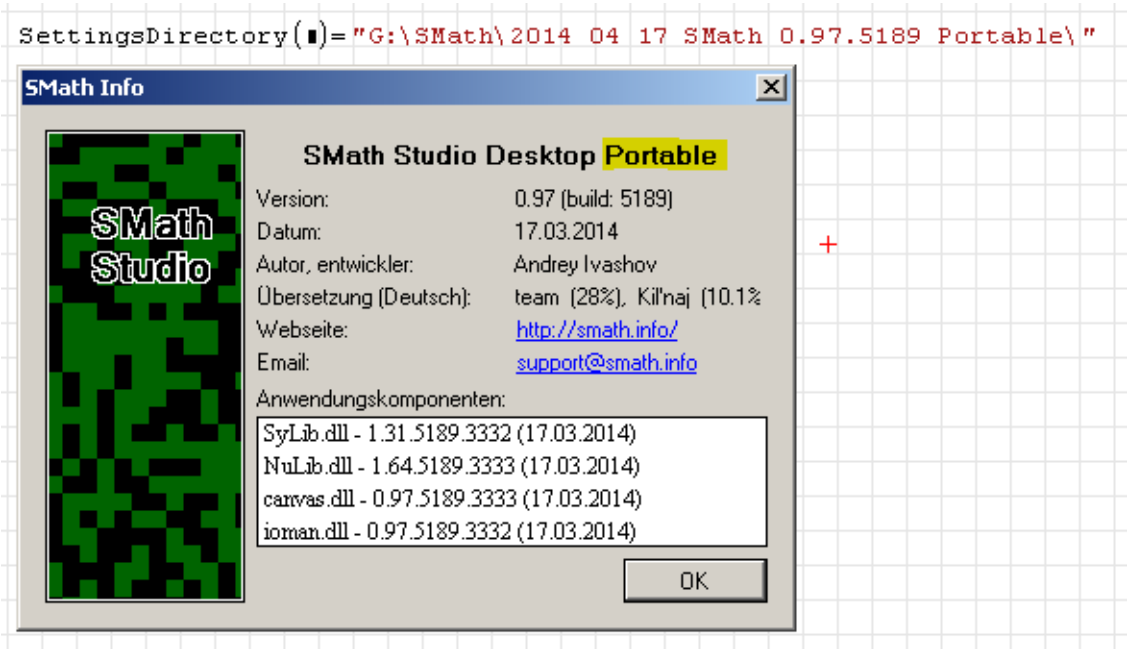
   Sie benötigen die Linux-Version:

2. Legen Sie das Installationsverzeichnis an, z.B. *D:\SMath\2014 04 23 SMath 0.97.5189* (eindeutige Versionsbezeichnung). Wenn Sie eine neue SMath-Version erproben möchten, legen Sie parallel dazu ein neues Verzeichnis an.
3. Entpacken von SMath Studio in das eben genannte Installationsverzeichnis.
4. Erzeugen Sie z.B. mit einem Texteditor die leere Datei *portable.version* im Installationsverzeichnis. Achten Sie darauf, dass Windows nicht noch eine unsichtbare Dateiendung anfügt.



5. Damit ist die portable Version eingerichtet. Sie können sie nun im Dateimanager starten. Überzeugen Sie sich davon, dass Sie tatsächlich eine portable Version haben, indem Sie mit `SettingsDirectory()` das Einstellungsverzeichnis abfragen oder unter **Hilfe> SMath Info** das Impressum anzeigen lassen.



6. Laden Sie das Maxima-Plugin und alle weiteren Plugins, die Sie benutzen wollen, aus der Online-Galerie im Erweiterungsmanager herunter und beenden Sie SMath Studio.

E.2. Portables Maxima

Maxima selbst stellt keine portablen Versionen bereit. Es gibt Windows Installer, der aktuellste ist Maxima-5.33.0-1, in der Version 5.33 funktioniert auch wieder das *draw*-Paket mit SMath.

Mit einigen Modifikationen in der Datei *maxima.bat* kann man aber eine portable Version bekommen. Diese Beschreibung geht davon aus, dass die Installation in das Installationsverzeichnis von SMath erfolgt, der Installationsort also z.B. *D:\SMath\2014 04 23 SMath 0.97.5189* heißt.

1. Download des Maxima-Windows-Installers

2. Installation unter *D:\SMath*. Der Installer erzeugt normalerweise ein Verzeichnis mit einer eindeutigen Versionsbezeichnung. Es können mehrere Versionen nebeneinander liegen.
3. Editieren von *maxima.bat*, zu finden unter *D:\SMath\2014 04 23 SMath 0.97.5189\Maxima...\bin\maxima.bat*.

   Alle Pfade werden relativ zum Pfad von maxima.bat `%~dp0` gesetzt.

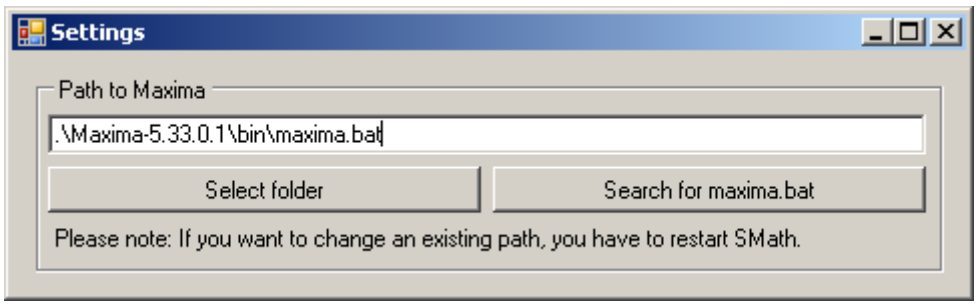
```
1  set PFAD=%cd%
2  cd /d "%~dp0"
3  cd ..
4  cd ..
5  rem begin localisation of Environment Variables
6  setlocal
```

```
23 set lisp=gcl
24 set version=5.33.0
25 set prefix=%cd%\Maxima-5.33.0.1
26 set maxima_prefix=%cd%\Maxima-5.33.0.1
27 set package=maxima
28 set verbose=false
29 set mingw_gccver=3.3.1
30 set path=%maxima_prefix%\gnuplot\bin;%maxima_prefix%\bin;%maxima_prefix%\
31
32 if "%USERPROFILE%" == "" goto win9x
33 if "%MAXIMA_USERDIR%" == "" set MAXIMA_USERDIR=%maxima_prefix%\Temp\user
34 if "%MAXIMA_TEMPDIR%" == "" set MAXIMA_TEMPDIR=%maxima_prefix%\Temp
35 goto startparseargs
36 :win9x
37 if "%MAXIMA_USERDIR%" == "" set MAXIMA_USERDIR=%maxima_prefix%\Temp\user
38 if "%MAXIMA_TEMPDIR%" == "" set MAXIMA_TEMPDIR=%maxima_prefix%\Temp
```

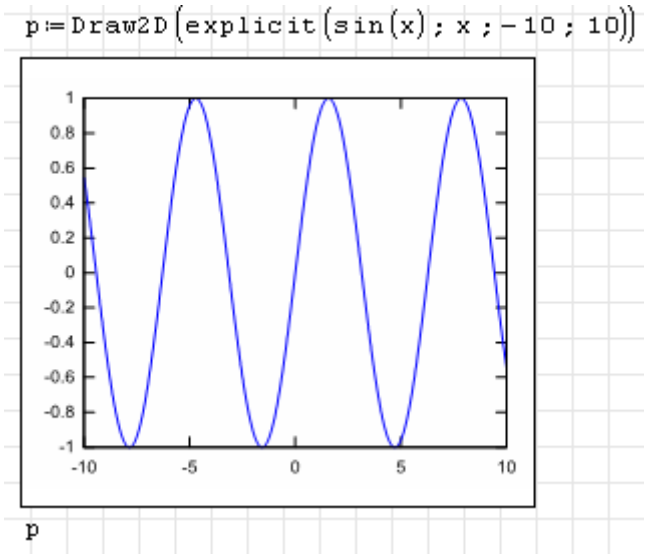
4. Erstellen der Verzeichnisse *User* und *Temp* im Maxima-Verzeichnis
5. Testen von *wxMaxima* durch Aufruf von *D:\SMath\2014 04 23 SMath 0.97.5189\Maxima...\wxMaxima\wxMaxima.exe*

E.3. Konfiguration von SMath

1. Starten Sie SMath neu und öffnen Sie die Pfadeinstellung unter **Einfügen> Maxima> Settings**.
2. Geben Sie den relativen Pfad zu *maxima.bat* an.



3. Testen Sie die Konfiguration, indem Sie ein Diagramm erzeugen:



E.4. Auslieferung

1. Aktualität der Plugins und des interaktiven Handbuchs sicherstellen (Erweiterungsverwaltung)
2. Ältere Plugin- und Handbuchversionen entfernen
3. Funktionstest mit *Plugin Maxima Test.sm* (im Handbuchverzeichnis)
4. Dateihistorie zurücksetzen
5. Einpacken in zip-File.
6. Mathcad EFI konfigurieren
   a) 32-bit-Version der exe verwenden
   b) dlls installieren
   c) Dokumentation anpassen
7. Zip-File in das Extension-Portal hochladen

F. Dachboden

Hier finden sich Abschnitte zu Funktionen, die nicht für die standardmäßige Benutzung empfohlen werden, da sie durch leistungsfähigere oder bedienungsfreundlichere ersetzt wurden. In alten Rechenblättern können diese allerdings noch vorhanden sein oder es mag im Einzelfall Gründe geben, sie auch in neuen Rechenblättern einzusetzen.

F.1. Gleichungslösung mit `solve()` und `roots()`

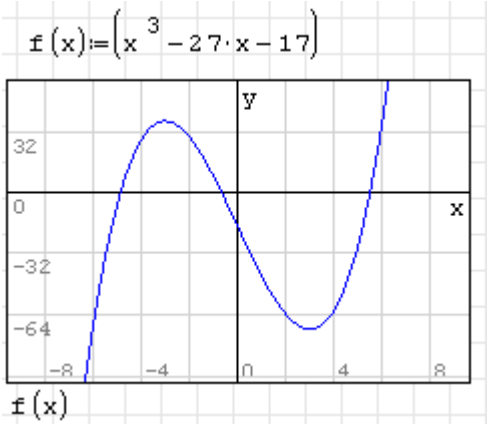
Diese Funktionen sollten Sie nur nutzen, wenn Sie genau wissen, warum Sie die in den vorigen Abschnitten beschriebenen Funktionen nicht benutzen wollen. Das kann z.B. der Fall sein, wenn Sie die Plugins nicht benutzen können oder eine ausführbare Anwendung erzeugen wollen (dies funktioniert aktuell mit Maxima nicht).

SMath bietet zwei eingebaute Funktionen, die auch nichtlineare Gleichungen lösen können: `solve()` und `roots()`. Bei beiden Funktionen ist das erste Argument die Gleichung, das zweite Argument die Unbekannte, nach der aufzulösen ist. Wird nur eine Seite der Gleichung angegeben, dann wird angenommen, dass die Nullstellen dieses Ausdrucks zu suchen sind. Als rechte Seite wird also eine Null ersetzt.

`solve()` sucht innerhalb eines Intervalls nach den Lösungen, `roots()` sucht von einem Startpunkt aus nach einer Lösung und kann Gleichungssysteme nach mehreren Variablen auflösen.

Die Gleichung kann einheitenbehaftet sein, die Unbekannte wird aber als dimensionslos betrachtet. Man kann also nicht direkt nach Längen, Kräften oder ähnlichem suchen. Vielmehr muss man die Variable dann als Vielfaches einer Bezugsgröße einbauen. Das ist insofern lästig, als dass in praktischen Anwendungen meist nach einheitenbehafteten Größen zu suchen ist.

Es ist stets zu empfehlen, die Funktion, deren Nullstellen man sucht, zunächst zu plotten, um eine Vorstellung davon zu bekommen, wie die Funktion aussieht und wo die Nullstellen ungefähr liegen könnten. Auch dabei ist zu beachten, dass die Variable x im Plot dimensionslos ist.



In diesem Beispiel benutzt `solve()` den voreingestellten Suchbereich (**Hauptmenü> Extras> Einstellungen> Berechnung> Wurzeln (Bereich)**). Er ist von -20 bis 20 eingestellt und enthält alle Nullstellen der Funktion $f(x)$.

$$\mathrm{solve}(f(x)\,;\,x)=\begin{pmatrix}-4{,}85\\-0{,}64\\5{,}49\end{pmatrix}$$

Wie man sieht, finden `solve()` alle Nullstellen. Im folgenden Fall werden die Nullstellen von $f(x/10)$ gesucht. Sie entsprechen dem Zehnfachen der oben gefundenen Werte. Nun liegt nur noch eine Nullstelle im voreingestellten Intervall:

$$\mathrm{solve}\left(f\left(\frac{x}{10}\right)\,;\,x\right)=-6{,}39$$

Man kann aber `solve()` explizit ansagen, in welchem Intervall gesucht werden soll. Diese Angabe überstimmt die Voreinstellung, ändert sie jedoch nicht. Das geht nur über das Menü.

$$\mathrm{solve}(f(x)\,;\,x\,;-100\,;\,100)=\begin{pmatrix}-4{,}85\\-0{,}64\\5{,}49\end{pmatrix}$$

Damit man nicht von der Voreinstellung abhängig ist, sollte man stets die Bereichsgrenzen angeben.

Die Funktion `roots()` findet von vornherein nur eine Lösung.

$$\mathrm{roots}(f(x)\,;\,x)=-0{,}64$$

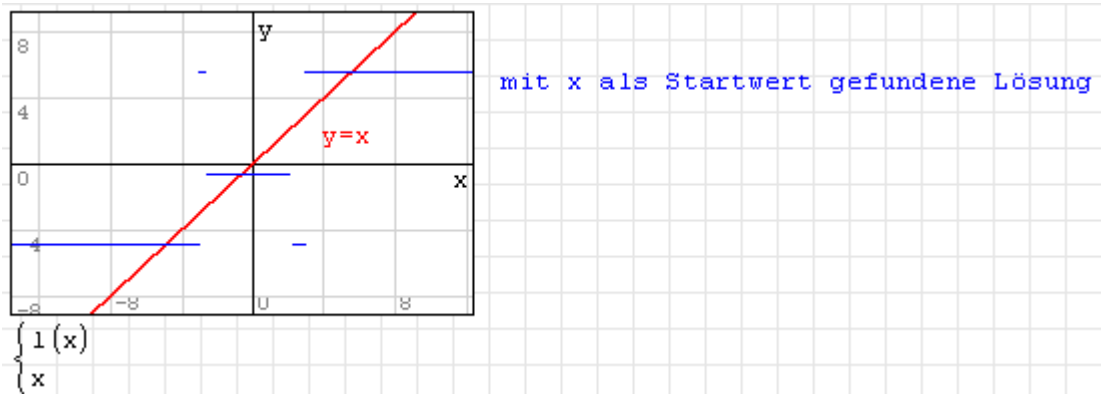
Diese ist vom Startpunkt der Suche abhängig, den der Anwender nicht kennt, wenn er ihn nicht vorgibt. Vermutlich ist Null als Startwert voreingestellt, aber das kann man weder direkt prüfen, noch ändern. Der Startpunkt kann aber als drittes Argument angegeben werden:

$$\mathrm{roots}(f(x)\,;\,x\,;\,4)=5{,}49$$

Man darf sich nicht darauf verlassen, dass immer die zum Startwert nächstliegende Lösung gefunden wird. Das kann man mit verschiedenen Startwerten ausprobieren. Dafür definiert man sich eine Funktion, die den Startwert als Argument bekommt und die damit bestimmte Lösung als Ergebnis liefert. Damit sich SMath nicht beschwert, dass x in der Funktion unbekannt ist, verpacken wir die Rechnung in einen Anweisungsblock (in welchem lokale Variablen möglich sind).

$$1(x_0):=\left|\ \mathrm{roots}(f(y)\,;\,y\,;\,x_0)\right.$$

Ein Plot dieser Funktion (blau) zeigt nun, mit welchem Startwert man welche Lösung bekommt. zum Vergleich ist in Rot die Linie $y = x$ gezeichnet.



Der Plot wurde auf Punktdarstellung umgeschaltet, damit die blaue Kurve der gefundenen Lösungen keine vertikalen Linien enthält.

Gleichungslösung mit Maßeinheiten

Wie bereits erwähnt, werden die Variablen, nach denen `roots()` und `solve()` auflösen, als dimensionslose Zahlen betrachtet. Wenn der Wert einer einheitenbehafteten physikalischen Größe ermittelt werden soll, so muss diese in der zu lösenden Gleichung als Produkt aus Zahlenwert (Variable, nach der gelöst wird) und einer Maßeinheit dargestellt werden. Das Ergebnis ist dann mit der gleichen Maßeinheit zu multiplizieren, um die gesuchte Größe zu bestimmen.

Hier ein Beispiel:

Assmann/Selke, Technische Mechanik 2, Aufgabe 7.7

Eine Druckspindel, $l = 800\,\text{mm}$ lang, mit $F = 20\,\text{kN}$ belastet, soll als Hohlwelle mit $d_i = 25\,\text{mm}$ ausgeführt werden. Für $S_K > 3{,}5$ ist der Außendurchmesser auf volle mm gerundet zu bestimmen.

7-7 $d_a = 33\,\text{mm}$; $F_d = 23\,\text{kN}$

$l := 800\ mm$	Spindellänge
$F_{dzul} := 20\ kN$	Druckkraft
$d_i := 25\ mm$	Innendurchmesser
$S_K := 3{,}5$	Knicksicherheit
$E := 210000\ MPa$	E-Modul Stahl (Annahme)
$l_K := l$	beidseitig gelenkig gelagert (Annahme)

Das Flächenträgheitsmoment I_{min} wird durch den Zahlenwert $\{I_{min}\} = zI_{min}$ und eine beliebige Längeneinheit zur vierten Potenz ausgedrückt. Das Ergebnis ist nicht abhängig von der gewählten Einheit. Die Funktion `roots()` sucht Werte für zI_{min}, die die gegebene Gleichung erfüllen. An der Gleichung selbst muss nichts geändert werden. Die Löser berücksichtigen bei der Auswertung alle bereits definierten Zusammenhänge für die in der Gleichung vorkommenden Größen. Daher kann nach zI_{min} aufgelöst werden, auch wenn es in der Gleichung nicht explizit vorkommt.

Erforderliches Flächenträgheitsmoment

$$I_{min} := zI_{min}\ cm^4$$

$$zI_{min} := roots\left(F_{dzul} = \frac{\pi^2 \cdot E \cdot I_{min}}{S_K \cdot l_K^2}\ ;\ zI_{min}\ ;\ 0\right) \qquad I_{min} = 2{,}16\ cm^4$$

Die Bestimmungsgleichung zur Ermittlung des Durchmessers ist nichtlinear. Die Vorgehensweise ist aber identisch.

Erforderlicher Außendurchmesser

$$D := zD\ mm$$

$$zD := roots\left(I_{min} = \frac{\pi}{64} \cdot \left(D^4 - d_i^4\right)\ ;\ zD\ ;\ \frac{d_i}{mm}\right) \qquad D = 30{,}19\ mm$$

$$D_{gew} := Ceil\left(\frac{D}{mm}\right) mm \qquad \boxed{D_{gew} = 31\ mm}$$

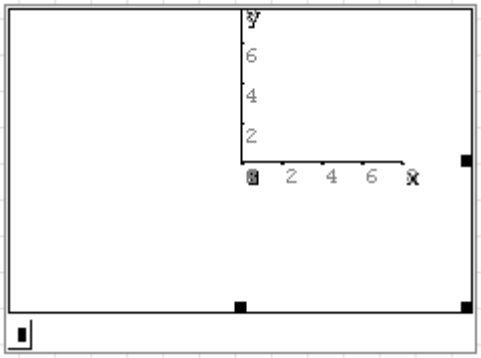
F.2. 3D-Diagramme (Standard)

SMath kann 3D-Diagramme darstellen. Die Funktion ist allerdings noch sehr unausgereift und wurde auch in den letzten zwei Jahren nicht weiterentwickelt:

- Die Ansicht (Zoom, Richtung und Verschiebung) werden nicht korrekt in der Rechenblattdatei gespeichert.
- Die angezeigten Achsen verlieren die Skalierung und Richtung bei Neuberechnung des Blattes.
- Das Koordinatensystem ist linkshändig.
- Bereichsgrenzen und Netzdichte für Funktionen von x und y können nicht beeinflusst werden.

F.2.0.1. 3D-Diagrammbereich erzeugen

Hauptmenü> Einfügen> Grafik> Dreidimensional



Der Diagrammbereich erscheint dann mit der x-Achse nach rechts, der y-Achse nach oben und der z-Achse nach hinten (Linkssystem!). Diese Ansicht wird stets wiederhergestellt, wenn die Datei geladen wird oder wenn der Knopf „Aktualisieren” im Grafikpanel der Seitenleiste gedrückt wird.

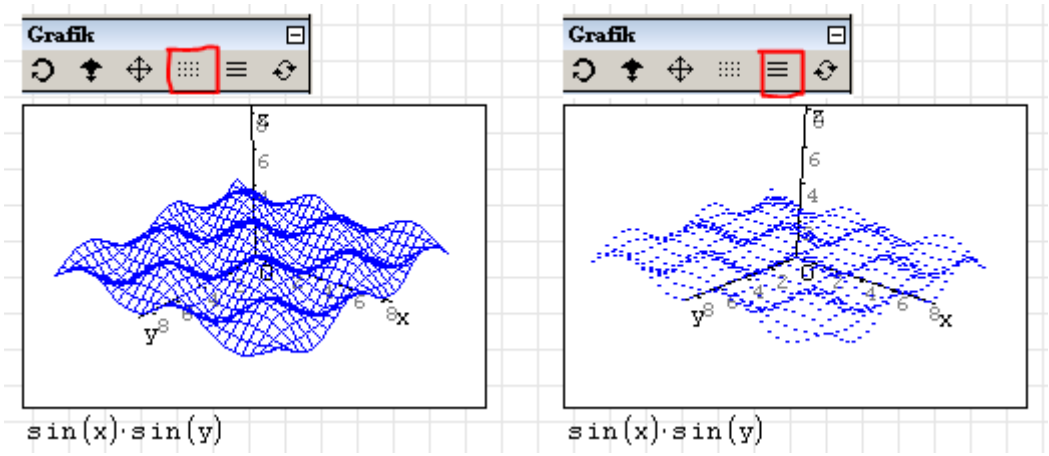
Der Diagrammbereich kann mit der Maus am rechten und unteren Rand in die gewünschte Größe gezogen werden.

Die Einfügemarke links unten nimmt darstellbare Objekte oder Listen darstellbarer Objekte auf.

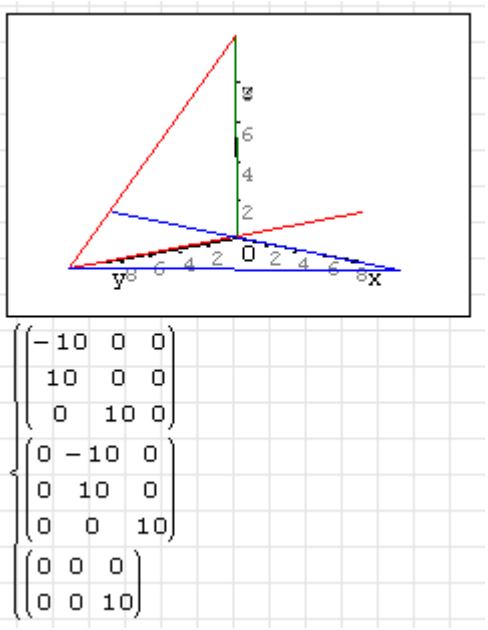
F.2.0.2. Darstellbare Objekte

Funktionen von x und y. Dabei kann weder auf die Zahl der Gitterlinien, noch auf den Wertebereich von x und y Einfluss genommen werden.

Im Grafikpanel der Seitenleiste kann man einstellen, ob man Linien oder Punkte sehen will.



Dreispaltige Matrizen, deren Zeilen jeweils die Koordinaten x, y und z eines Punktes enthalten. Die Punkte einer Matrix werden mit einem Linienzug verbunden. Das Koordinatensystem scheint linkshändig zu sein. Die x-Achse weist nach vorn rechts, die y-Achse weist nach vorn links und die z-Achse weist nach oben. Dies passt nicht zu einem rechtshändigen Koordinatensystem.



$$\left\{\begin{pmatrix} -10 & 0 & 0 \\ 10 & 0 & 0 \\ 0 & 10 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -10 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 10 \end{pmatrix}\right\}$$

Parametrische Flächen Mit der Funktion `CreateMesh()` aus dem Plugin *3DPlotRegion* können Gitternetze aus parametrischen Funktionen

$$\begin{bmatrix} x(u,v) \\ y(u,v) \\ z(x,y) \end{bmatrix}$$

erzeugt werden. Mit der Einstellung

$$\begin{bmatrix} u \\ v \\ f(u,v) \end{bmatrix}$$

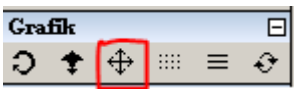
erzeugt diese Funktion Darstellungen analog zur eingebauten Funktionsdarstellung, allerdings sind dann Wertebereich und Netzdichte einstellbar. Siehe dazu auch das Beispiel in Abschnitt 9.4.1.

F.2.0.3. Navigation im 3D-Diagrammfenster

Die Ansicht kann im 3D-Fenster mit der Maus geändert werden. Diese Einstellungen bleiben aber beim Schließen der Datei nicht erhalten.

Beachten Sie: Koordinatenachsen und angezeigte Grafikobjekte passen nur zusammen, wenn vorher der Knopf „Aktualisieren" im Grafikpanel der Seitenleiste gedrückt wird.

Die Navigationsfunktionen werden in der Seitenleiste im Panel „Grafik" angewählt.

Aktion			
Ziehen	Drehen	Zoom	Verschieben
Mausrad	Zoom	Zoom	Zoom

F.3. X-Y Plot (erweiterte 2D-Diagrammfunktion)

Das Plugin *X-Y Plot Region* stellt einen alternativen Diagrammbereich bereit, der den internen Standard-Diagrammbereich ergänzt. Er wird allerdings nicht aktiv weiterentwickelt.

Dieser Bereich bietet zusätzliche Möglichkeiten:

- Darstellung implizit gegebener Kurven $f(x,y)=0$
- Darstellung unterbrochener Linienzüge (Vektoren von Wertepaar-Matrizen)
- Frei formatierbare Achsenbeschriftungen und Diagrammtitel (ohne Formelformatierung)
- Frei einstellbare Linien- und Markerform für Funktionen und Wertematrizen
- Frei einstellbarer Achsenbereich und Gitternetzweite
- Zweite y-Achse
- Legende

Einschränkungen gegenüber dem Standard-2D-Diagramm sind

- keine Textobjekte darstellbar
- Keine Animationsfunktion
- Mauszoom nur für beide Achsen gleichzeitig.

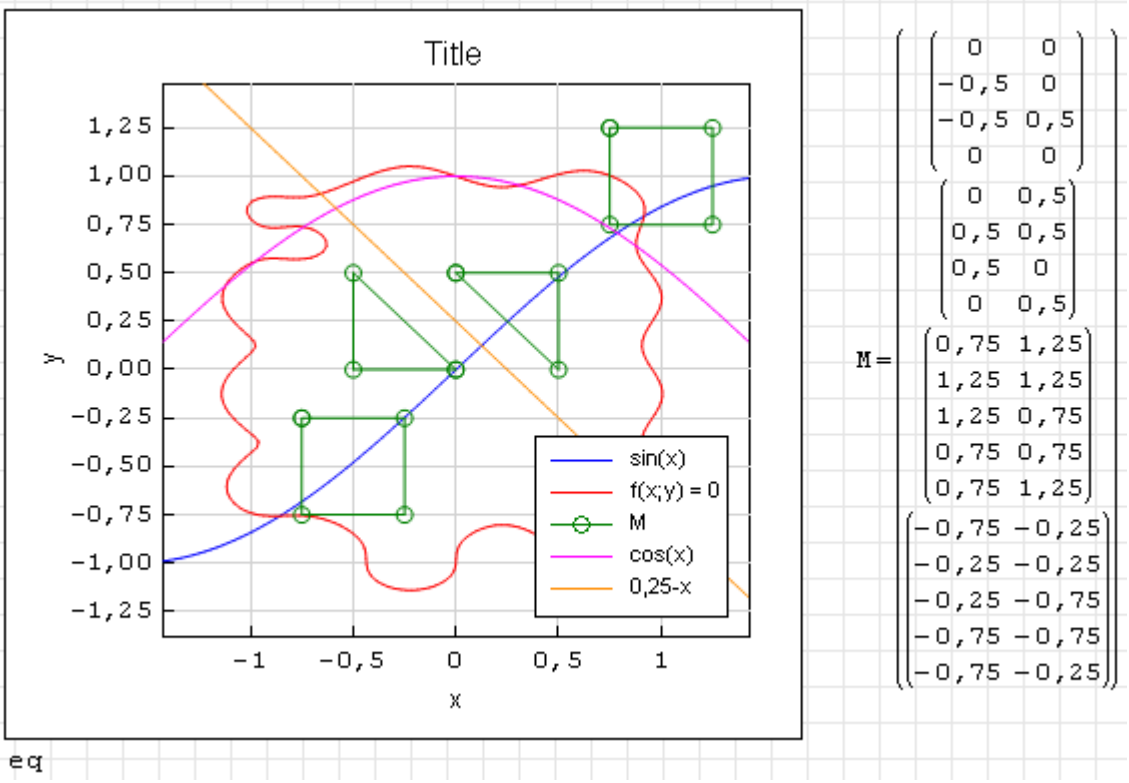
Aufgrund eines Programmfehlers gehen Linienformatierungen und Legendeneinträge verloren, wenn die dargestellten Funktionen nicht auswertbar sind (auch, wenn dies nur vorübergehend passiert).

Hier ein Beispiel für die Darstellungsmöglichkeiten.

$$f(x;y) := x^4 + y^4 + 0{,}4 \cdot \sin(7 \cdot x) + 0{,}3 \cdot \sin(4 \cdot \pi \cdot y) - 1$$

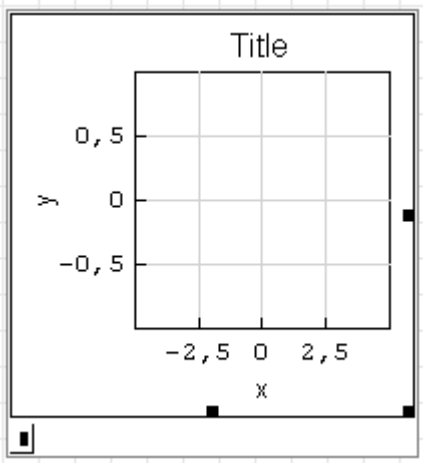
$$M_1 := \begin{pmatrix} 0 & 0 \\ -0{,}5 & 0 \\ -0{,}5 & 0{,}5 \\ 0 & 0 \end{pmatrix} \quad M_2 := \begin{pmatrix} 0 & 0{,}5 \\ 0{,}5 & 0{,}5 \\ 0{,}5 & 0 \\ 0 & 0{,}5 \end{pmatrix} \quad M_3 := \begin{pmatrix} 0 & 0{,}5 \\ 0{,}5 & 0{,}5 \\ 0{,}5 & 0 \\ 0 & 0 \\ 0 & 0{,}5 \end{pmatrix} + 0{,}75 \quad M_4 := M_3 - 1{,}5$$

$$eq := \begin{cases} \sin(x) \\ f \\ M \\ \cos(x) \\ -x + 0{,}25 \end{cases}$$



F.3.1. X-Y Plot einfügen

Ein X-Y Plotbereich kann über das **Hauptmenü> Einfügen> X-Y Plot** eingefügt werden.

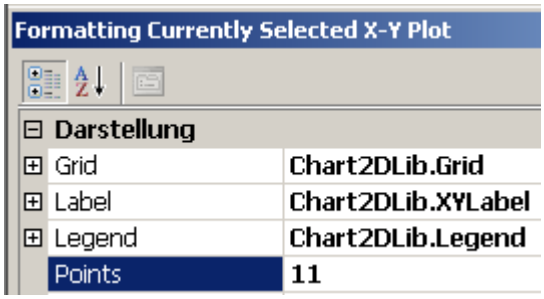


An den Platzhaltern am Rahmen kann die Größe des Bereichs durch Ziehen mit der Maus verändert werden.

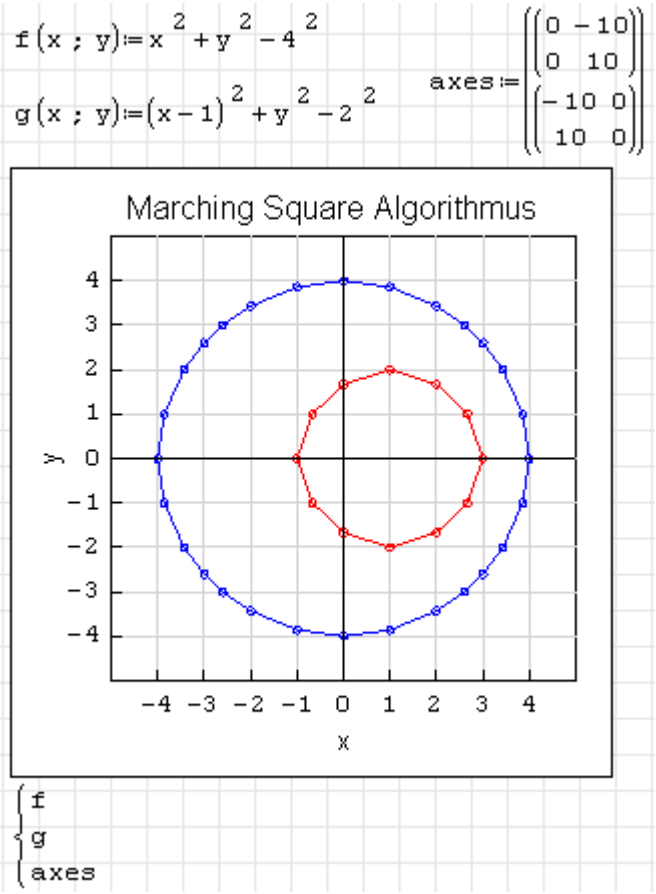
In den Platzhalter links unten schreibt man den darzustellenden Ausdruck.

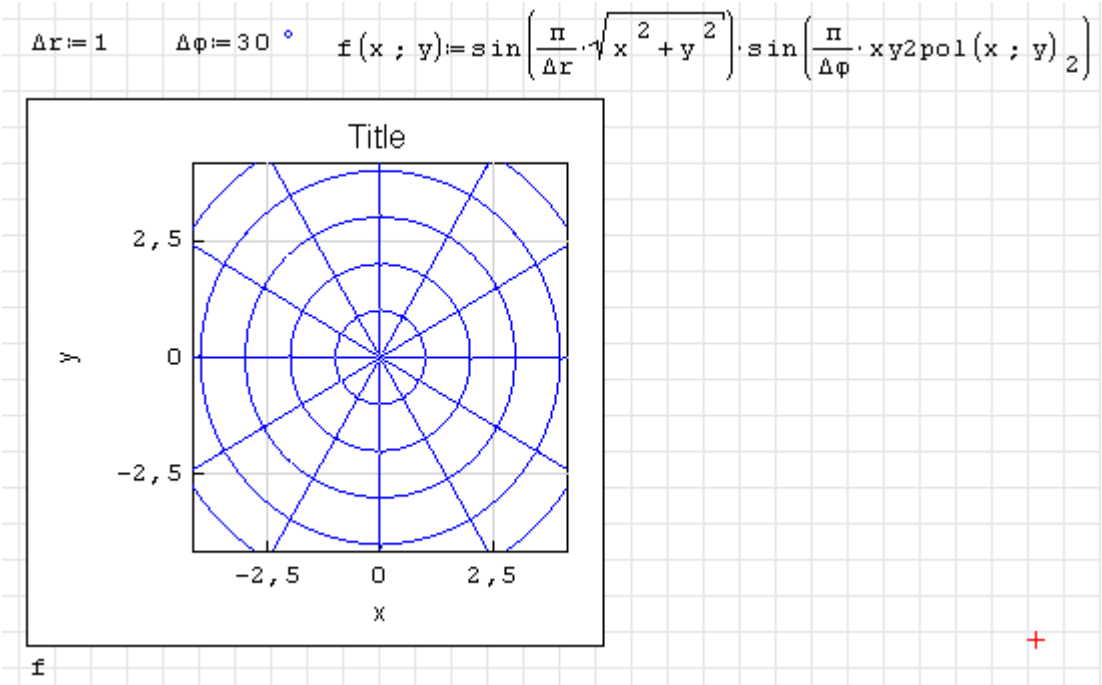
F.3.2. Implizite Funktionen

Der Y-X-Plotbereich kann implizit gegebene Funktionen darstellen. Die Funktion ist als $f(x,y)$ zu definieren, deren Nullstellen dargestellt werden. Dafür wird der sogenannte Marching-Squares-Algorithmus verwendet. Dabei fallen nur Punkte auf einem regelmäßigen Gitter an, dessen Dichte mit der Format-Einstellung „Points" vorgegeben wird. Hier ein Beispiel (Points = 11)



Ein weiteres Beispiel mit einer komplexeren Funktion, die Nullstellen bilden ein Polarkoordinatenraster mit wählbarem Radius- und Winkelschritt.





F.3.3. Mausfunktionen im X-Y Plot

Doppelklick links: Öffnen des Format-Editors (alternativ zum Kontextmenü-Eintrag „Format")

Klick links: Aktivierung des Diagramms für die Bearbeitung (Größe ändern, Achsen mit der Maus skalieren und verschieben, Zugriff auf das Kontextmenü).

Ziehen an den Randmarkern: Größe der Diagrammfläche ändern, dabei bleiben die Achsgrenzen erhalten, d.h. die Achsen werden entsprechend gedehnt oder gestaucht.

Mausrad: Skalierung der Achsen (x und y gleichzeitig). Eine eventuell aktive y2-Achse ist davon nicht betroffen.

Ziehen: Achsen verschieben.

F.4. Diagramme mit plot2D/3D

In Maxima gibt es zwei Wege zu Diagrammen zu kommen:

1. Das Paket *draw* mit den Funktionen `draw2d()` und `draw3d()`. Diese werden für den Standardanwender zur Benutzung empfohlen.
2. Die Funktionen `plot2d()` und `plot3d()`. Diese sind weniger flexibel, wurden daher nicht so stark im Maxima-Plugin unterstützt, wie das Paket *draw*.

Dieser Abschnitt zur Dokumentation der Funktionen `plot2d()` und `plot3d()` fällt daher auch sehr knapp aus.

Es stehen folgende Plot-Befehle in Maxima zur Verfügung

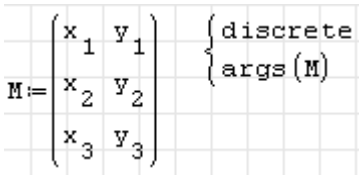
`contour_plot(expr; xrange; yrange...)` Höhenliniendiagramm für eine einzelne Funktion

`implicit_plot(expr_list; xrange; yrange...)` Darstellung einer oder mehrerer implizit gegebener Funktionen. Muss mit `load(implicit_plot)` geladen werden.

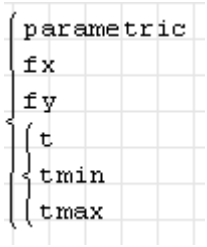
`plot2d(expr_list; xrange,....)` Darstellung von expliziten Funktionen

`plot2d(expr_list;...)` parametrische Funktionen oder Punktmengen, hier ist die Option `xrange` nicht Pflicht.

- Diskrete Punktmengen: Stellt Listen von Koordinatenpaaren dar, z.B. mittels `args(Matrix)` erzeugt)



- Parametrische Funktion:



`plot3d(expr; xrange; yrange...)` Parametrische oder explizit gegebene Funktion

F.4.1. Optionen

(immer als Listen mit dem Schlüsselwort zuerst angeben)

Option	Bedeutung
`adapt_depth`	(5) Maximale Teilungstiefe der adaptiven Kurvenabtastung
`axes`	(true): beide Achsen zeichnen false: keine Achse zeichnen x oder y: eine Achse zeichnen
`azimuth`	(30) Drehwinkel um z in °
`box`	(true): Kastenrand zeichnen false: keinen Kastenrand zeichnen
`color`	(blue, red, green, magenta, black. cyan): Farben für die Kurven oder die Ober- und Unterseite von 3D-Flächen (wenn keine Palette verwendet wird)
`color_bar`	(false): keine Farblegende zeichnen true: Farblegende zeichnen
`color_bar_tics`	(): Achsmarken an der Farblegende, Startwert oder Startwert und Inkrement oder Startwert, Inkrement und Endwert.
`elevation`	(60): Drehung um x in °
`gnuplot_term`	() Ausgabe-Format-Spezifikation (Text nach „set terminal”)
`gnuplot_out_file`	(maxplot.xxx) Ausgabedateiname
`grid`	(30, 30): Abtastpunkte in x- und y-Richtung bei 3D-Diagrammen
`grid2d`	(false): Gitternetz in der xy-Ebene
`iterations`	(9): Iterationen bei `mandelbrot` und `julia`
`label`	Liste mit text, x, y. Text wird an den gegebenen Koordinaten angezeigt.
`legend`	(Funktionsnamen): Liste der Legendeneinträge für die Kurven false: keine Legende zeichnen
`logx`	(): wenn vorhanden, wird x logarithmisch skaliert
`logy`	(): wenn vorhanden, wird y logarithmisch skaliert
`mesh_lines_color`	(black): Farbe der Gitterlinien, wenn eine Farbpalette benutzt wird. false: keine Gitterlinien anzeigen
`nticks`	(29): Punktezahl für die Kurvenabtastung
`palette`	({hue; 0,25, 0,7; 0,8; 0,5) Farbpalette für die Flächenfärbung false: Flächen nicht färben
`png_file`	(): Ausgabe in PNG-Datei
`point_type`	(Liste) Symbole für Kurvenart points oder linespoints
`same_xy`	(): gleiche Skalierung für die x- und y-Achse
`same_xyz`	(): gleiche Skalierung für die x-, y- und z-Achse
`style`	(lines), Parameter: Linienstärke, Farbe (1: blau, 2: rot, 3: magenta, 4: orange, 5: braun, 6: lime, 7: aqua points: Parameter: Radius, Farbe (wie bei lines), Symbol Linespoints: Linienstärke, Punktradius, Farbe, Symbol impulses:
`svg_file`	(): Ausgabe als SVG-Datei
`t`	(-3, 3): Parameterbereich für parametrische Kurven
`title`	(): Diagrammtitel
`transform_xy`	(false): kartesische Koordinaten

	polar_to_xy: Polarkoordinaten spherical_to_xy: Kugelkoordinaten
`yx_ratio`	Verhältnis zwischen vertikaler und horizontaler Bildabmessung
`x, y, z`	min, max: Kurvenbereich oder Anzeigebereich
`x/y/zlabel`	(): Achsenbeschriftung
`x/y/ztics`	(): Achsmarken. Startwert oder Startwert und Inkrement oder Startwert, Inkrement und Endwert.
`xy_scale`	sx, sy. Skalierfaktoren des Diagrammberechs
`zmin`	minimale z-Koordinate in 3D-Plots

point_type		
bullet	1	
circle	2	
plus	3	
times	4	
asterisk	5	
box	6	
square	7	
triangle	8	
delta	9	
wedge	10	
nabla	11	
diamond	12	
lozenge	12	

Literaturverzeichnis

Assmann, B./Selke, P. (2009): Technische Mechanik 2. Festigkeitslehre. Oldenbourg 127, 128

Buchmayr, B. (2002): Werkstoff- und Produktionstechnik mit Mathcad. Modellierung und Simulation in Anwendungsbeispielen. Springer 16

Henning, G./Jahr, A./Mrowka, U. (2004): Technische Mechanik mit Mathcad, Matlab und Maple. Vieweg 16

Krawietz, A. (1997): Maple V für das Ingenieurstudium. Springer 16

Werkle, H. (2012): Mathcad in der Tragwerksplanung. Elektronische ArbeitsblÃ€tter fÃŒr Statik, Stahlbetonbau, Stahlbau und Holzbau. Vieweg+Teubner 16

Index