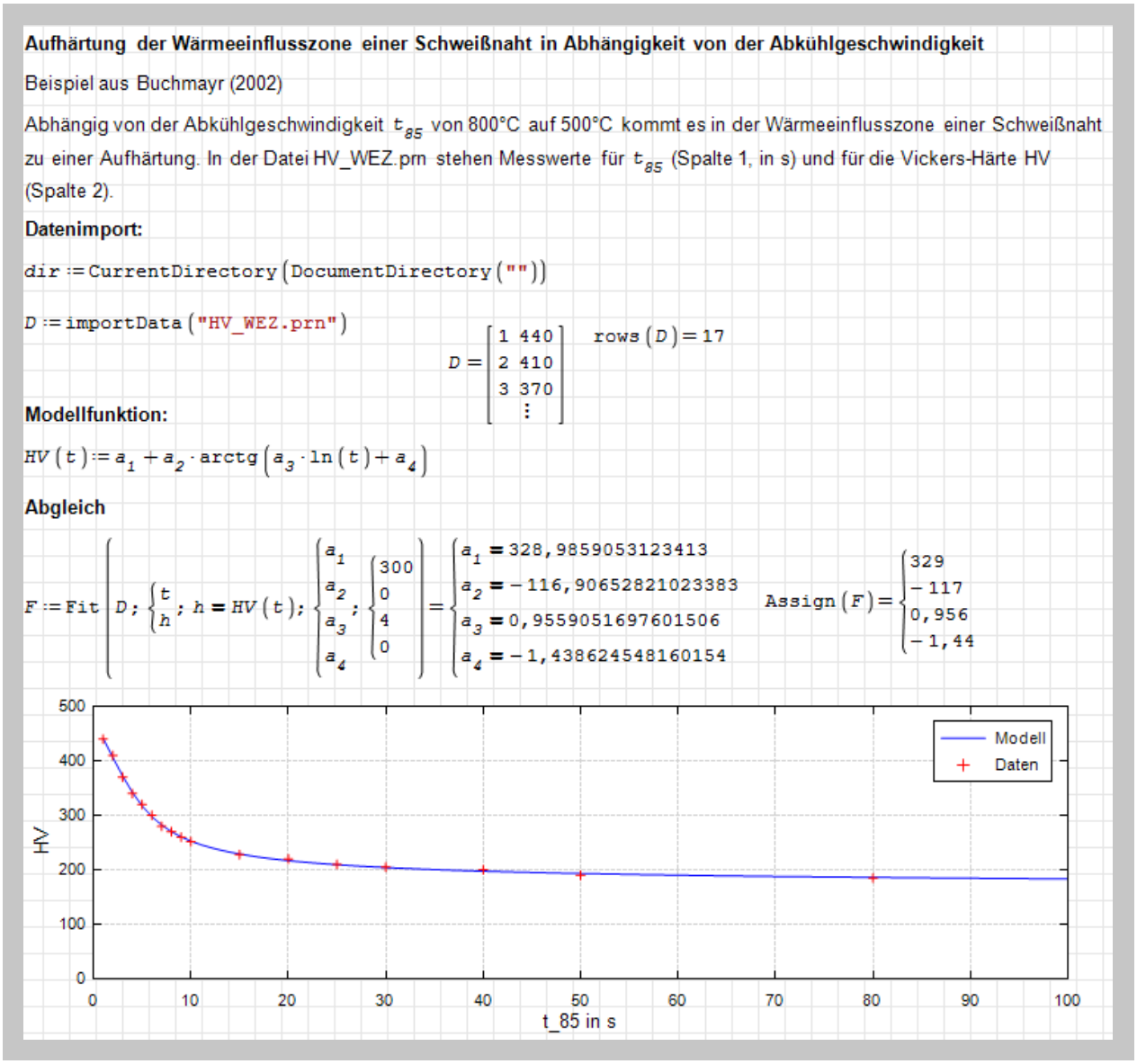
Technisches Rechnen mit SMath

Matrizen, Diagramme, Datenverarbeitung

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



29. September 2023

Martin Kraska
Technisches Rechnen mit SMath. Matrizen, Diagramme, Datenverarbeitung.
Technische Hochschule Brandenburg, 2023
DOI: 10.25933/opus4-2948

Inhaltsverzeichnis

1 Einleitung **5**
- 1.1 Einstellungen für die Ergebnisdarstellung . . . 6
- 1.2 Benötigte Plugins . . . 7

2 Matrizen und Vektoren **8**
- 2.1 Matrizen eingeben . . . 9
- 2.2 Größe ändern . . . 10
- 2.3 Nullmatrix, Einheitsmatrix, Diagonalmatrix . . . 11
- 2.4 Zufallszahlen . . . 12
- 2.5 Vektoren mit aufsteigenden oder absteigenden Zahlenwerten . . . 13
- 2.6 Matrizen mit Schleifen erzeugen . . . 14
- 2.7 Extrahieren und Kombinieren . . . 16

3 Rechnen mit Matrizen **18**
- 3.1 Algebra . . . 19
- 3.2 Vektorprodukte . . . 21
- 3.3 Funktionen auf Vektor- oder Matrixelemente anwenden . . . 23

4 Datenimport **24**
- 4.1 Datenimport aus Textdateien . . . 25
- 4.2 Große Matrizen anzeigen . . . 26
- 4.3 Suchen und Sortieren . . . 27
- 4.4 Eindimensionale Interpolation . . . 28
- 4.5 Zweidimensionale Interpolation . . . 29

5 Diagramme **30**
- 5.1 X-Y Plot . . . 32
- 5.2 Einstellungen und Mausfunktionen . . . 33
- 5.3 Achsen, Beschriftungen, Legende . . . 34
- 5.4 Datenreihen darstellen . . . 35
- 5.5 Achsenbereiche automatisch steuern . . . 36
- 5.6 Explizite Funktionen darstellen . . . 38
- 5.7 Implizite Funktionen darstellen . . . 39
- 5.8 Textmarken darstellen . . . 40
- 5.9 Plotten mit Maßeinheiten . . . 41
- 5.10 Geometrische Grundformen . . . 43
- 5.11 Diagramme mit Maxima . . . 45

6 Tabellen **46**
- 6.1 Tabellen erzeugen . . . 47
- 6.2 Zeilen- und Spaltentitel . . . 48
- 6.3 Maßeinheiten in Tabellen . . . 49

7 Datenanalyse **51**
7.1 Deskriptive Statistik 53
7.2 Quantile 54
7.3 Boxplots 55
7.4 Histogramme 56

8 Lineare Regression **57**
8.1 Beispiel 58

9 Nichtlineare Regression **60**
9.1 Beispiel 1: Kubische Parabel durch 5 Zufallspunkte 61
9.2 Beispiel 2: Kreis durch 4 Punkte 62
9.3 Beispiel 3: Anpassung eines Modells an eingelesene Daten 63

Literatur **65**

1 Einleitung

Inhalt

1.1 Einstellungen für die Ergebnisdarstellung 6
1.2 Benötigte Plugins 7

SMath ist ein Programm zur Durchführung und Dokumentation technischer Berechnungen. Ein SMath-Dokument besteht aus frei auf dem Papier angeordneten Formeln, Texten, Diagrammen und Bildern. Formeln kann man aus taschenrechnerartigen Paletten zusammenklicken oder als erfahrener Anwender ganz ohne Maus, nur mit der Tastatur erstellen. Großartig ist die umfassende Unterstützung von Maßeinheiten. Die Formeln sind gleichzeitig Rechenanweisung und Dokumentation. Formeln erscheinen in natürlicher mathematischer Schreibweise (nicht wie Programmcode), so dass auch Dritte, die das Programm nicht kennen, sie verstehen können.

SMath entstand ursprünglich als Hobbyprojekt des Sankt Petersburger Programmierers Andrey Ivashov. Inzwischen wird es von der SMath GmbH (ООО Эсмат) entwickelt und betreut, deren Geschäftsführer Andrey Ivashov ist. Für die Betreuung von Kunden außerhalb Russlands gibt es eine Niederlassung in Israel (smath.co.il).

SMath hat eine integrierte Erweiterungsverwaltung (extension manager). Diese bietet Zugriff auf eine Online-Galerie, in der die Community Programmerweiterungen (plugins), Anwendungsbeispiele, Dokumentationen und Code-Bausteine (snippets) veröffentlichen kann. Dort ist auch die vorliegende Anleitung zu finden.

Diese Anleitung überschneidet sich mit einigen Abschnitten des Handbuchs Kraska (2020), ist jedoch eine aktualisierte und thematisch fokussierte Alternative. Themen sind:

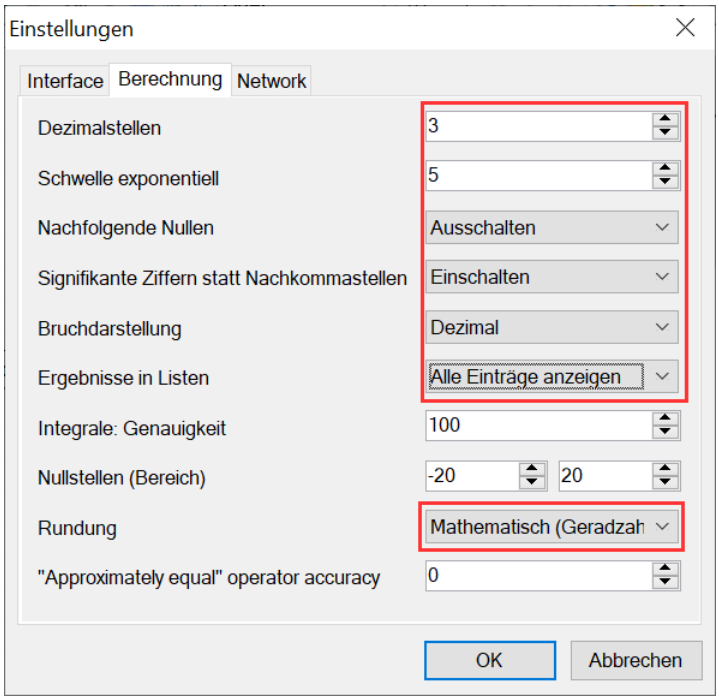
- Rechnen mit Matrizen (Kapitel 2 und 3)
- Daten erzeugen, importieren und verarbeiten (Kapitel 4)
- Daten in Diagrammen und Tabellen darstellen (Kapitel 5 und 6)
- Statistische Modelle an Daten anpassen, z.B. Funktionen an Versuchsergebnisse)
- Hinweise zur Programmierung finden Sie in den Abschnitten 2.6 und 7.2.

Es wird vorausgesetzt, dass Sie grundsätzlich mit dem Bedienkonzept von SMath vertraut sind, z.B. aus der Anleitung „Technisches Rechnen mit SMath Studio - Erste Schritte" (Kraska 2023).

Hinweis: Der Umgang mit großen Datenmengen und deren grafische Aufbereitung ist nicht die große Stärke von SMath. Für einfache Versuchsauswertungen z.B. in der Werkstoffprüfung reichen die Funktionen aber allemal.

1.1 Einstellungen für die Ergebnisdarstellung

Sicherheitshalber wird hier noch mal auf die Einstellungen zur Ergebnisdarstellung hingewiesen. Unter **Extras**> **Einstellungen**> **Berechnung** kann man die Voreinstellungen für die Ergebnisdarstellung ändern. Sie wirken sich nur auf neue Formeln aus. Für bestehende Formeln ändert man die Einstellungen im Kontextmenü des Formelbereichs.



Die Einstellungen besagen:

- 3 Dezimalstellen sollen angezeigt werden. Diese Genauigkeit ist für technische Berechnungen meist sinnvoll.
- Ab dem Wert 10^5 werden Zahlen in Exponentialschreibweise dargestellt.
- Nachfolgende Nullen an Dezimalbrüchen werden weggelassen, also 1 statt 1,00
- Die Dezimalstellen geben die Mantissenlänge (Zahl der signifikanten Ziffern) und nicht die der Nachkommastellen an. Zusammen mit der Einstellung Dezimalstellen = 3 gibt das z.B. $\pi = 3{,}14$.
- Brüche werden als Dezimalbruch (und nicht mit Bruchstrich) angezeigt.
- Rundungen erfolgen mathematisch (Geradzahlregel, wenn auf die letzte zu erhaltende Stelle nur noch eine 5 folgt, dann wird zur nächstliegenden Geradzahl gerundet).

1.2 Benötigte Plugins

In dieser Anleitung werden Funktionen benutzt, die mit Plugins bereitgestellt werden.

- *Custom Functions* (Matrixfunktionen)
- *DotNumerics* (numerische Funktionsbibliothek)
- *ImageRegion* (Darstellung von Maxima-Diagrammen)
- *Mathcad Toolbox* (Verzeichnisfunktionen)
- *Maxima Plugin* (Statistische Auswertung, Boxplot, Histogramme, logarithmische Diagramme)
- *StatisticalTools* (Zufallszahlen)
- *Table Region* (Tabellen)
- *X-Y Plot Region* (Diagramme)

Plugins können bei Bedarf aus SMath heraus installiert werden:

- **Extras> Plug-ins**
- Rechts oben: „Lokale Speicherung" auf „Online Galerie" umstellen.
- In der Liste „Plug-ins" das gewünschte Plugin auswählen.
- „Installieren" drücken und Installation abwarten.

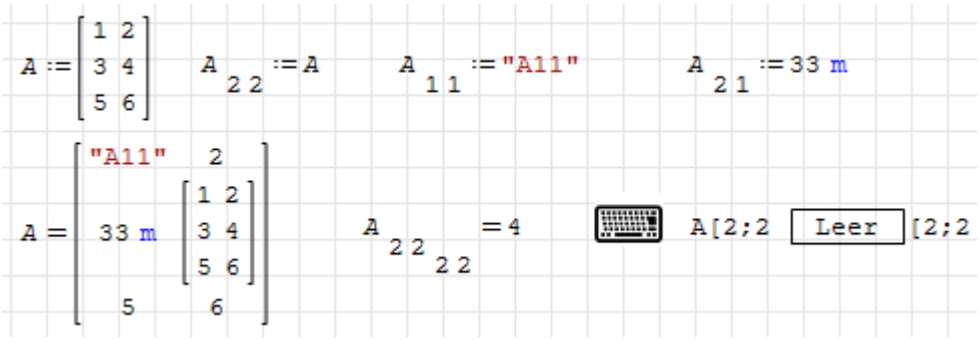
Meist funktioniert das installierte Plugin auch ohne Neustart von SMath.

2 Matrizen und Vektoren

Inhalt

2.1 Matrizen eingeben 9
2.2 Größe ändern 10
2.3 Nullmatrix, Einheitsmatrix, Diagonalmatrix 11
2.4 Zufallszahlen 12
2.5 Vektoren mit aufsteigenden oder absteigenden Zahlenwerten 13
2.6 Matrizen mit Schleifen erzeugen 14
2.7 Extrahieren und Kombinieren 16

Matrizen sind rechteckige Anordnungen von mathematischen Ausdrücken. Eine Matrix mit m Zeilen und n Spalten heißt $m \times n$-Matrix. Vektoren im Sinne der linearen Algebra sind Matrizen mit einer einzigen Spalte.



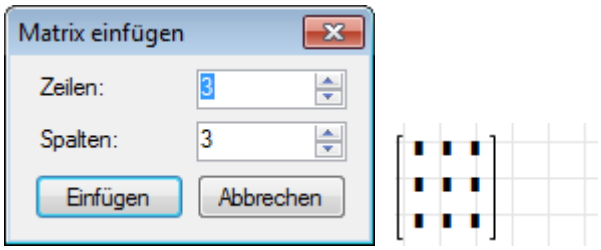
Matrixelemente können durch Elementindizes angesprochen werden, die mit der eckigen Klammer [erzeugt werden. Ein zweiter Index wird mit dem Argumenttrennzeichen (Semikolon) erzeugt. Dann adressiert der erste Index die Zeile, der zweite Index die Spalte.

2.1 Matrizen eingeben

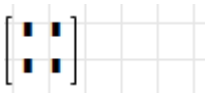
Dialog „Matrix einfügen". Er wird aufgerufen durch:

- Befehl [▦] aus der Matrix-Palette
- Strg-M oder
- **Einfügen> Matrix**

Zeilen- und Spaltenzahl angeben, Einfügen erzeugt ein entsprechend großes Feld von Platzhaltern



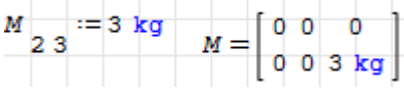
Tastatur: Ein 2×2 Platzhalter wird erzeugt durch Eintippen von `mat` [⇆] oder `mat(`



Zuweisung von Werten zu Elementen: Elemente werden mit Elementindizes angesprochen. Sie werden durch `[` oder mit dem Symbol [▪] in der „Matrizen"-Palette erzeugt.

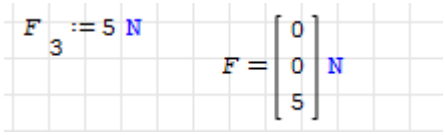
Die erzeugten Matrizen oder Vektoren sind gerade so groß, wie die höchsten vorkommenden Indizes.

`M[2;3` Leer `:3kg` [⇆] Enter



Verwendet man nur *einen* Elementindex, wird ein Spaltenvektor erzeugt.

`F[3` Leer `:3N` [⇆] Enter

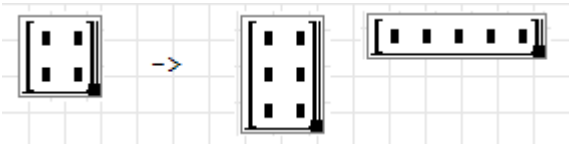


Dabei haben wir uns das Ergebnis in Newton (N) anzeigen lassen.

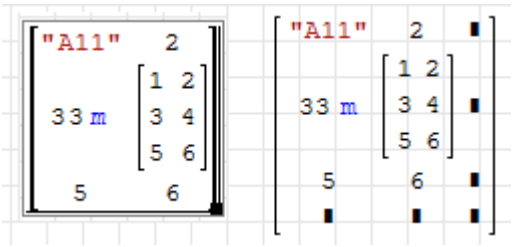
Elementindizes (mit `[` erzeugt) unterscheiden sich in der Formatierung von Textindizes (mit `.` erzeugt) durch einen größeren Abstand vom Formelzeichen. Zudem ist die Schrift in Textindizes kleiner.

2.2 Größe ändern

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

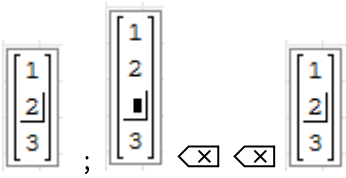


Beim Vergrößern einer Matrix entstehen unten neue Zeilen oder rechts neue Spalten.

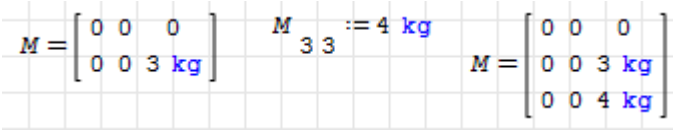


Elemente in Vektoren einfügen/löschen: Bei Vektoren (einspaltigen Matrizen) lassen sich Zeilen einfügen, indem man am Ende eines Eintrags das Argumenttrennzeichen ; (Semikolon) eingibt.

Entsprechend kann man auch leere Platzhalter löschen, indem man das (unsichtbare) Argumenttrennzeichen davor mit ⌫ löscht.



Zuweisen an Matrixelemente außerhalb der bisherigen Größe. Die Matrix wird auf das erforderliche Maß vergrößert.



Bei Matrizen kann man keine Spalten oder Zeilen direkt einfügen oder löschen. Man muss dafür Funktionen benutzen. Daher ist es häufig sinnvoll, Daten in einem externen Programm zu erfassen und dann zu importieren

2.3 Nullmatrix, Einheitsmatrix, Diagonalmatrix

Einheitsmatrix gegebener Dimension erzeugen:

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

Diagonalmatrix aus einem Vektor :

$$\texttt{diag}\left(\begin{bmatrix}1\\2\\3\end{bmatrix}\right)=\begin{bmatrix}1&0&0\\0&2&0\\0&0&3\end{bmatrix}$$

Nullmatrix gegebener Dimension erzeugen (`Zeros()` aus Plugin *Custom Functions*):

$$\texttt{Zeros}(2)=\begin{bmatrix}0\\0\end{bmatrix}\qquad \texttt{Zeros}(2;3)=\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}$$

$$\texttt{matrix}(2;4)=\begin{bmatrix}0&0&0&0\\0&0&0&0\end{bmatrix}$$

Man kann auch Matrizen mit Null Spalten oder Null Zeilen erzeugen, z.B. um dann in einer Programmschleife Spalten oder Zeilen hinzuzufügen.

$$\texttt{matrix}(0;2)=\texttt{mat}(0;2)$$

$$\texttt{matrix}(3;0)=\texttt{mat}(3;0)$$

Einsmatrix gegebener Dimension (`Ones()` aus Plugin *Custom Functions*):

$$\texttt{Ones}(2)=\begin{bmatrix}1\\1\end{bmatrix}\qquad \texttt{Ones}(2;3)=\begin{bmatrix}1&1&1\\1&1&1\end{bmatrix}$$

2.4 Zufallszahlen

Das Plugin *StatisticalTools* bietet Funktionen zur Erzeugung von Zufallszahl-Matrizen.

Gleichverteilte reelle Zufallszahlen zwischen 0 und 1. Die Funktionsargumente geben die gewünschte Matrixgröße an.

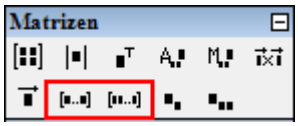
$$\mathrm{Random}(5)=\begin{bmatrix}0{,}0787\\0{,}6839\\0{,}1124\\0{,}1472\\0{,}9406\end{bmatrix}\qquad \mathrm{Random}(5;3)=\begin{bmatrix}0{,}8439 & 0{,}5629 & 0{,}2685\\0{,}1561 & 0{,}1612 & 0{,}6583\\0{,}5344 & 0{,}0177 & 0{,}2768\\0{,}603 & 0{,}034 & 0{,}832\\0{,}0248 & 0{,}0748 & 0{,}0724\end{bmatrix}$$

Gleichverteilte ganzzahlige Zufallszahlen in einem bestimmten Intervall. Die Funktionsargumente geben die gewünschte Matrixgröße an.

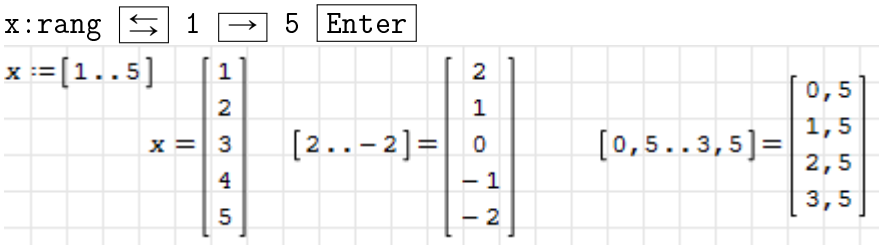
$$\mathit{Random}_N(3;-5;5)=\begin{bmatrix}2\\-3\\-1\end{bmatrix}\qquad \mathit{Random}_N(3;3;-5;5)=\begin{bmatrix}3 & 0 & -3\\-2 & 3 & -1\\4 & 0 & 0\end{bmatrix}$$

2.5 Vektoren mit aufsteigenden oder absteigenden Zahlenwerten

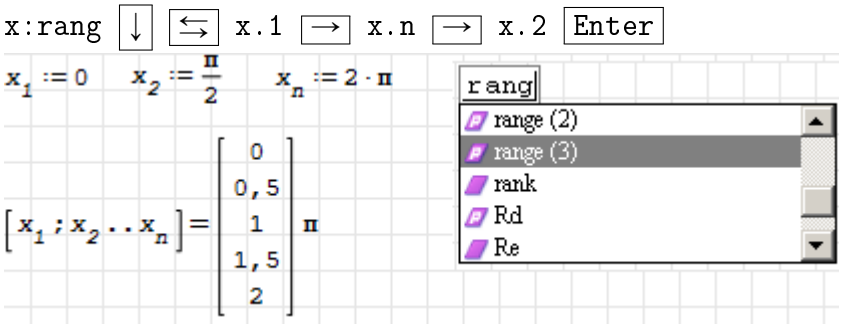
Für Diagramme oder in Programmschleifen braucht man häufig Vektoren mit aufeinanderfolgenden Zahlenwerten. Diese können mit der Funktion `range()` erzeugt werden. Eingabe über die Tastatur oder aus der Matrizen-Palette der Seitenleiste.



Zahlenfolgen mit Schrittweite +1 oder -1. Anzugeben sind das erste und das letzte Element.



Zahlenfolgen mit beliebiger Schrittweite. Anzugeben sind das erste, das zweite und das letzte Element:



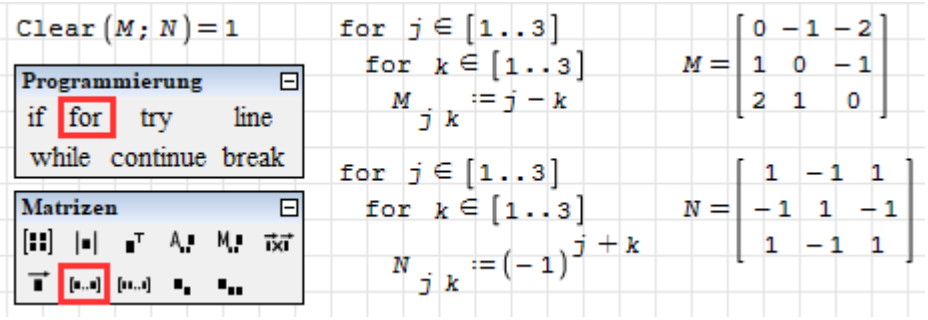
Hier wurde π als Faktor (Maßeinheit) hinter das Ergebnis geschrieben.

2.6 Matrizen mit Schleifen erzeugen

Schleifen kann man verwenden, wenn sich die Elemente einer Matrix oder eines Vektors mit einer Berechnungsvorschrift aus den Indizes ergeben.

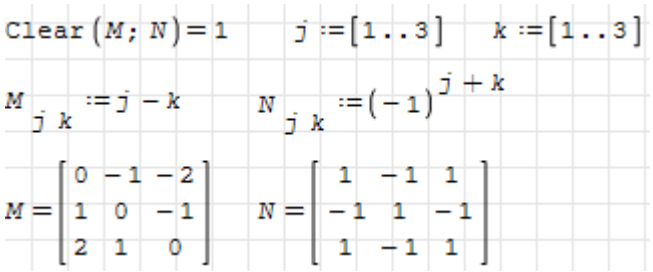
Explizite Schleifen: Diese werden mit der `for`-Anweisung gebildet (for eintippen oder die Programierung-Palette der Seitenleiste benutzen). Anzugeben sind

- der Name der Laufvariable (Vorsicht mit der Verwendung von i, dies ist die imaginäre Einheit),
- ein Vektor mit den zu durchlaufenden Werten (mit der Funktion `range()` oder aus der Matrizen-Palette),
- und der Schleifenkörper (die Berechnungsanweisung). Die `for`-Anweisung entnimmt man der Programierung-Palette der Seitenleiste.

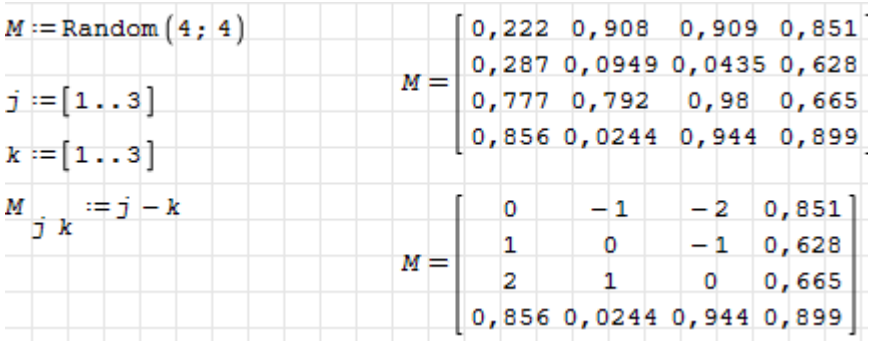


Implizite Schleifen entstehen, wenn man die Indizes direkt als vektorwertige Variable definiert. Dann wird die Zuweisung für alle Werte der Variable ausgeführt. Diese Variable wird daher auch Bereichsvariable (*range variable, ranged index*) genannt.

Das ist zu expliziten Schleifen gleichwertig, sieht aber übersichtlicher aus.



Es werden nur die in der Schleife adressierten Elemente überschrieben. Wenn die Matrix größer ist, bleiben vorhandene Elemente bestehen. Deshalb wurden die Matrizen in den Beispielen oben sicherheitshalber mit `Clear()` gelöscht.



Im folgenden Beispiel ist die nur teilweise Überschreibung der Matrix allerdings beabsichtigt. Die zweite Zeile der Matrix wird Null gesetzt.

$$M := Random_N(3; 4; 0; 10) = \begin{bmatrix} 2 & 6 & 3 & 8 \\ 6 & 5 & 5 & 7 \\ 6 & 1 & 10 & 7 \end{bmatrix}$$

$$M_{2[1..4]} := 0 \qquad M = \begin{bmatrix} 2 & 6 & 3 & 8 \\ 0 & 0 & 0 & 0 \\ 6 & 1 & 10 & 7 \end{bmatrix}$$

2.7 Extrahieren und Kombinieren

Abmessungen ermitteln: Zahl der Spalten (*columns*), Zahl der Zeilen (*rows*) und Zahl der Elemente:

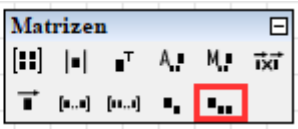
$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad \text{cols}(A) = 2 \quad \text{rows}(A) = 3 \quad \text{length}(A) = 6$$

Zeilen oder Spalten extrahieren:

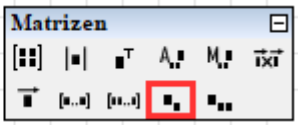
$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad \text{col}(A; 1) = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad \text{row}(A; 3) = \begin{bmatrix} 5 & 6 \end{bmatrix}$$

Elemente extrahieren: Dies erfolgt mit einem Index bei Vektoren (auch bei Zeilenvektoren) und mit zwei Indizes bei Matrizen.

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

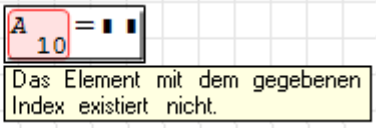


$$b := \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \quad b_2 = 3$$



Lineare Indizierung: Verwendet man bei einer Matrix nur einen Index, werden die Elemente zeilenweise durchgezählt.

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad A_2 = 2 \quad A_3 = 3$$



Das funktioniert auch beim Zuweisen von Werten. Dabei werden dynamisch so viele Zeilen wie nötig hinzugefügt.

$$A_7 := \pi \qquad A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 3{,}14 & 0 \end{bmatrix}$$

Untermatrizen: Zusammenhängende Teilgebiete können mit `submatrix()` herausgegriffen werden:

$$M := \begin{bmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{bmatrix} \quad \text{submatrix}(M; 1; 2; 2; 3) = \begin{bmatrix} 11 & 17 \\ 25 & 39 \end{bmatrix}$$

Flexibler geht es mit **vektorwertigen Indizes**. Diese geben an, welche Zeilen und welche Spalten extrahiert werden sollen. Im Beispiel werden die erste und dritte Zeile und alle Spalten extrahiert. Das entspricht einer Streichung der zweiten Zeile.

$$M := \begin{bmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{bmatrix} \quad M_{[1\ 3][1..3]} = \begin{bmatrix} 5 & 11 & 17 \\ 17 & 39 & 61 \end{bmatrix} \quad \begin{matrix} [1\ 3] & \text{1x2-Matrix} \\ [1..3] & \text{Bereich (range)} \end{matrix}$$

Verwendet man einen Indexvektor in umgekehrter Reihenfolge, kann man die Matrix horizontal oder vertikal spiegeln:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}_{[3..1][1..3]} = \begin{bmatrix} g & h & i \\ d & e & f \\ a & b & c \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}_{[1..3][3..1]} = \begin{bmatrix} c & b & a \\ f & e & d \\ i & h & g \end{bmatrix}$$

Kombiniert man skalare und vektorielle Indizes, so kann man Zeilen oder Spalten herausgreifen.

$$M := Random_N(3; 4; 0; 10) = \begin{bmatrix} 0 & 6 & 6 & 9 \\ 9 & 5 & 6 & 0 \\ 1 & 9 & 9 & 4 \end{bmatrix}$$

$$\text{row}(M; 2) = [9\ 5\ 6\ 0] \qquad M_{2[1..\text{cols}(M)]} = [9\ 5\ 6\ 0]$$

$$\text{col}(M; 3) = \begin{bmatrix} 6 \\ 6 \\ 9 \end{bmatrix} \qquad M_{[1..\text{rows}(M)]3} = \begin{bmatrix} 6 \\ 6 \\ 9 \end{bmatrix}$$

Erweitern: (Nebeneinanderstellen): Matrizen gleicher Zeilenzahl oder Matrizen und Skalare werden mit der Funktion `augment()` nebeneinander gestellt.

$$\text{augment}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}; \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}\right) = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 3 \\ 5 & 6 & 5 \end{bmatrix}$$

$$\text{augment}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}; \text{"o"}; 10; \text{"red"}\right) = \begin{bmatrix} 1 & 2 & \text{"o"} & 10 & \text{"red"} \\ 3 & 4 & \text{"o"} & 10 & \text{"red"} \\ 5 & 6 & \text{"o"} & 10 & \text{"red"} \end{bmatrix}$$

`augment()` nebeneinander gestellt. : Matrizen gleicher Spaltenzahl kann man mit der Funktion `stack()` stapeln.

$$\text{stack}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}; \begin{bmatrix} 1 & 4 \\ 3 & 8 \end{bmatrix}\right) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 1 & 4 \\ 3 & 8 \end{bmatrix}$$

3 Rechnen mit Matrizen

Inhalt

3.1 Algebra 19
3.2 Vektorprodukte 21
3.3 Funktionen auf Vektor- oder Matrixelemente anwenden . . 23

Hier besprechen wir die elementaren Rechenoperationen mit Matrizen und auch Vektoren.

In Abschnitt 3.3 kommt auch schon mal ein Diagramm vor. Auf Diagramme wird genauer in Kapitel 5 eingegangen.

Suchen und Sortieren finden Sie in Abschnitt 4.3.

3.1 Algebra

Multiplikation und Addition/Subtraktion von Matrizen mit einer Zahl. Es werden die ganz normalen Rechenoperationen verwendet:

$$\begin{bmatrix}1&2\\3&4\end{bmatrix}+5=\begin{bmatrix}6&7\\8&9\end{bmatrix}\qquad \begin{bmatrix}1&2\\3&4\end{bmatrix}-10=\begin{bmatrix}-9&-8\\-7&-6\end{bmatrix}$$

$$\begin{bmatrix}1&2\\3&4\end{bmatrix}\cdot 3=\begin{bmatrix}3&6\\9&12\end{bmatrix}\qquad 5\cdot\begin{bmatrix}1&2\\3&4\end{bmatrix}=\begin{bmatrix}5&10\\15&20\end{bmatrix}\qquad \begin{bmatrix}1&2\\3&4\end{bmatrix}\cdot\frac{1}{3}=\begin{bmatrix}0{,}333&0{,}667\\1&1{,}33\end{bmatrix}$$

Matrixmultiplikation. Die Spaltenzahl der ersten Matrix muss gleich der Zeilenzahl der zweiten Matrix sein. Es wird der normale Multiplikationsoperator verwendet.

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

Matrixinversion. Am einfachsten erzeugt man sie durch den Exponent -1.[1]

$$\begin{bmatrix}a&b\\2\cdot a&3\cdot b\end{bmatrix}^{-1}=\begin{bmatrix}\frac{3}{a}&-\frac{1}{a}\\-\frac{2}{b}&\frac{1}{b}\end{bmatrix}\qquad \begin{bmatrix}5&11&17\\11&25&23\\17&39&61\end{bmatrix}^{-1}=\begin{bmatrix}4{,}906&-0{,}062&-1{,}344\\-2{,}188&0{,}125&0{,}562\\0{,}031&-0{,}062&0{,}031\end{bmatrix}$$

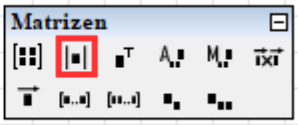
Hier wurde das linke Ergebnis symbolisch angezeigt. a und b haben keine Werte.

Die Inverse existiert nur, wenn die Matrix nicht singulär ist, also ihre Determinante ungleich Null ist

Determinante: $\det C$ wird symbolisch angezeigt mit Ctrl-.

$$C:=\begin{bmatrix}a&b\\c&d\end{bmatrix}\qquad |C|=a\cdot d-c\cdot b$$

$$A:=\begin{bmatrix}0&1\\1&0\end{bmatrix}\qquad |A|=-1$$



Rang: Der Rang einer Matrix ist die Zahl der linear unabhängigen Zeilen oder Spalten.

$$A:=\begin{bmatrix}0&1\\1&0\end{bmatrix}\qquad \operatorname{rank}(A)=2{,}000$$

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

$D^{-1}=$ ∎ ∎

Die Matrix ist nicht invertierbar (Determinante gleich Null)

In D ist die zweite Zeile genau 2 mal die erste Zeile, sie ist also linear von der ersten Zeile abhängig. Darum ist der Rang nicht 2 sondern 1.

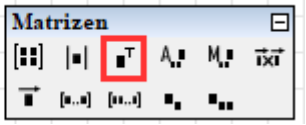
Spur: Die Spur einer Matrix ist gleich der Summe ihrer Diagonalelemente.

[1] Es gibt aber auch eine Funktion inverse(), die bietet aber keine Vorteile.

$$M := \begin{bmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 17 & 39 & 61 \end{bmatrix} \quad \mathrm{tr}(M) = 91 \quad \sum_{j=1}^{3} M_{j,j} = 91$$

Transposition: Spiegelung der Matrix an ihrer Hauptdiagonale. Eingabe über die Matrizen-Palette oder als Funktion `transpose()`

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



3.2 Vektorprodukte

Die Operationen werden hier mit symbolischen Vektoren demonstriert. Ihre Elemente sind nicht-definierte Variablen. Bei den entsprechenden Formeln wird symbolisch ausgewertet Ctrl-. oder → in der Arithmetik-Palette

$$a := \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad b := \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad c := \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Skalarprodukt: Das ist die Summe der Produkte gleich indizierter Elemente.

$$a \cdot b = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$
$$a \cdot a = a_1^{\,2} + a_2^{\,2} + a_3^{\,2}$$
$$a^2 = a_1^{\,2} + a_2^{\,2} + a_3^{\,2}$$

Das Skalarprodukt kann auch als Matrixprodukt geschrieben werden, wobei der linke Faktor transponiert wird. Beachte: Das Ergebnis ist eine 1×1 Matrix.

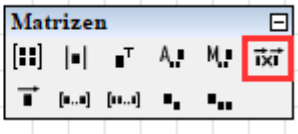
$$a^T \cdot b = \begin{bmatrix} a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3 \end{bmatrix}$$

Betrag eines Vektors: Das ist die Wurzel aus dem Skalarprodukt mit sich selbst. Er entspricht der Euklidischen Norm (Funktion `norme()`)

$$\sqrt{a^2} = \sqrt{a_1^{\,2} + a_2^{\,2} + a_3^{\,2}} \qquad \sqrt{v_a^{\,2}} = 2{,}24 \qquad \text{norme}\left(v_a\right) = 2{,}24$$

Kreuzprodukt: Das Kreuzprodukt zweier Vektoren ist ein Vektor, der auf beiden Vektoren senkrecht steht. Sein Betrag ist das Produkt beider Beträge mal dem Sinus des eingeschlossenen Winkels. Einfügen aus der Matrixpalette oder **Einfügen> Operator**

$$a \times b = \begin{bmatrix} a_2 \cdot b_3 - a_3 \cdot b_2 \\ a_3 \cdot b_1 - a_1 \cdot b_3 \\ a_1 \cdot b_2 - a_2 \cdot b_1 \end{bmatrix}$$



Das Kreuzprodukt funktioniert in SMath Studio nur für Vektoren mit 3 Elementen.

Dyadisches Produkt: Das ist das Matrixprodukt zweier Vektoren, wobei der rechte Faktor transponiert wird.

$$a \cdot b^T = \begin{bmatrix} a_1 \cdot b_1 & a_1 \cdot b_2 & a_1 \cdot b_3 \\ a_2 \cdot b_1 & a_2 \cdot b_2 & a_2 \cdot b_3 \\ a_3 \cdot b_1 & a_3 \cdot b_2 & a_3 \cdot b_3 \end{bmatrix}$$

Spatprodukt: Das ist das Volumen eines von drei Vektoren (mit je drei Elementen) aufgespannten Parallelepipeds (Spat). Es kann mit einer Kombination aus Kreuz- und Skalarprodukt berechnet werden.

$$(a \times b) \cdot c = \left(a_2 \cdot b_3 - a_3 \cdot b_2\right) \cdot c_1 + \left(a_3 \cdot b_1 - a_1 \cdot b_3\right) \cdot c_2 + \left(a_1 \cdot b_2 - a_2 \cdot b_1\right) \cdot c_3$$

Das Spatprodukt lässt sich auch als Determinante einer Matrix berechnen, die aus den nebeneinandergestellten Vektoren besteht.

$$v_a := \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \quad v_b := \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \quad v_c := \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \qquad \left(v_a \times v_b\right) \cdot v_c = 12$$

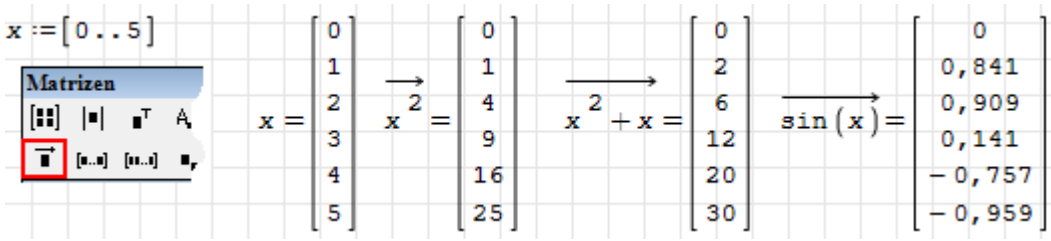
$$M := \text{augment}\left(v_a ; v_b ; v_c\right) = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 1 \\ -2 & 0 & 3 \end{bmatrix} \qquad |M| = 12$$

3.3 Funktionen auf Vektor- oder Matrixelemente anwenden

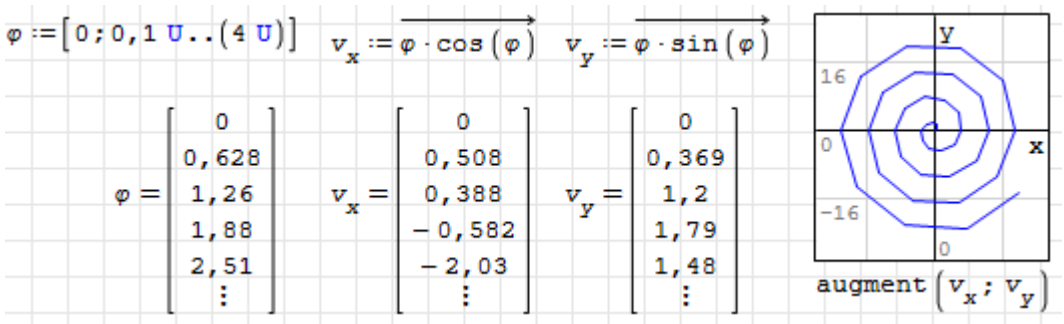
Dafür gibt es mehrere Optionen:

- Die Funktion `vectorize()` wendet einen gegebenen Ausdruck auf alle Elemente der darin vorkommenden Variablen an. Das entspricht dem Rechnen mit Tabellenspalten in Programmen wie Excel.
- Explizite Programmschleifen (werden später behandelt)
- Implizite Schleifen mit Bereichsvariablen.

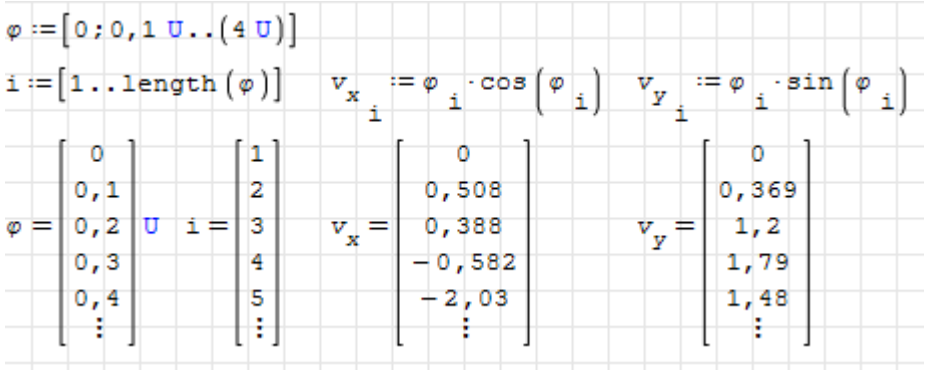
Vektorisierung von Ausdrücken: Die Funktion `vectorize()` kann über die Tastatur mit der dynamischen Hilfe oder über das Matrix-Panel der Seitenleiste eingegeben werden.



Hier ein Anwendungsbeispiel: Darstellung einer Spirale. „U" ist eine Winkeleinheit, die Umdrehung (360°).



Implizite Schleifen: Diese Variante wurde bereits für die Erstellung von Matrizen vorgestellt. Sie ist lesbarer, da man nicht wissen muss, was der Pfeil über dem Ausdruck bedeutet:



4 Datenimport

Inhalt

4.1 Datenimport aus Textdateien 25
4.2 Große Matrizen anzeigen 26
4.3 Suchen und Sortieren 27
4.4 Eindimensionale Interpolation 28
4.5 Zweidimensionale Interpolation 29

Beim Umgang mit großen Datenmengen ist es sehr wichtig, dass alle Formeln numerisch ausgewertet werden, sonst können Rechenzeiten und Speicherbedarf unangenehm werden.

Wir besprechen hier den Datenimport, die Anzeige großer Datenmengen, das Sortieren und Suchen sowie die Dateninterpolation.

Zum Rechnen mit Datenspalten siehe Abschnitt 3.3.

In Abschnitt 4.4 kommen wieder mal Diagramme vor. Darauf gehen wir im genauer in Kapitel 5 ein.

Zudem zeigen wir im Abschnitt 4.4 eine Tabelle. Die werden näher in Kapitel 6 erläutert.

4.1 Datenimport aus Textdateien

Es ist sinnvoll, relative Pfadnamen (bezogen auf das aktuelle Verzeichnis) zu verwenden. Dafür muss das aktuelle Verzeichnis gleich dem Speicherort des Rechenblatts (Dokuments) sein. Das stellen Sie mit den Funktionen `CurrentDirectory()` und `DocumentDirectory()` sicher. Sie sind im Plugin *Mathcad Toolbox* definiert.

```
dir := CurrentDirectory(DocumentDirectory(""))

dir = "C:\FHB\Software\SMath\SMath Skript\Bilder\"
```

Die Funktion `importData()` liest Textdateien. Es gibt zwei Formen.

```
ImportData(name)
ImportData(name; ddec; darg; dcol; r1; r2; c1; c2; fsym)
```

Bedeutung der Argumente (soll die Voreinstellung benutzt werden, ist der Wert 0 anzugeben).

`name`	Dateiname, absolut oder relativ zum Rechenblatt
`ddec`	Dezimaltrennzeichen (Voreinstellung wie im Programm, z.B Komma)
`darg`	Argumenttrennzeichen (Voreinstellung wie im Programm, z.B Semikolon)
`dcol`	Spaltentrennzeichen (Voreinstellung: Leerzeichen)
`r1, r2`	Erste und letzte zu lesende Zeile (Voreinstellung: 1, Zeilenzahl)
`c1, c2`	Erste zu letzte lesende Spalte (Voreinstellung: 1, Spaltenzahl)
`fsym`	1: Zahlen und Text, 0: nur Zahlen

Beispieldatei: *data.txt*

```
1 "abc" 2,2
2 "cde" 3,3
3 "efg" 4,4
"dfr" 2 3
```

Mit `fsym=1` können Zahlen und Text gelesen

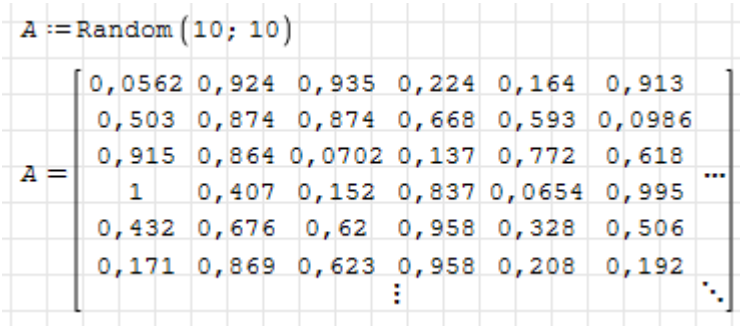
```
D := importData("data.txt"; ","; ";"; " "; 0; 0; 1; 3; 1)

D = [   1    "abc" 2,2 ]
    [   2    "cde" 3,3 ]
    [   3    "efg" 4,4 ]
    [ "dfr"    2     3  ]
```

Mit `fsym=0` (letztes Argument) können nur Zahlenwerte gelesen werden, z.B. die dritte Spalte der Datei. Hier wird für den Zeilenbereich die Voreinstellung benutzt.

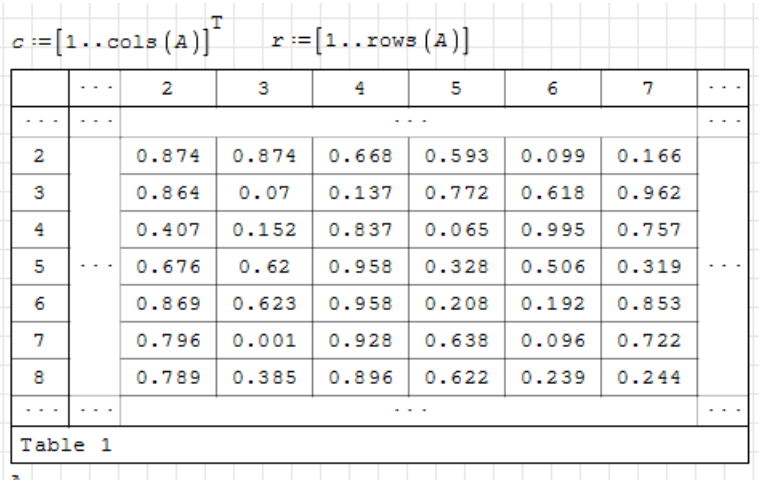
```
D := importData("data.txt"; ","; ";"; " "; 0; 0; 3; 3; 0)

D = [ 2,2 ]
    [ 3,3 ]
    [ 4,4 ]
    [  3  ]
```

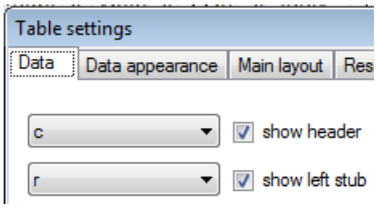
4.2 Große Matrizen anzeigen

Bei der Ergebnisanzeige von Matrizen kann man an der rechten unteren Ecke mit der Maus ziehen, um die Anzahl der angezeigten Zeilen oder Spalten zu reduzieren. Es erscheinen dann Auslassungszeichen in der Matrix.



Eine Alternative mit erweiterten Formatierungsmöglichkeiten ist der Tabellenbereich (Plugin *Table Region*). Hier kann eingestellt werden, wie groß der anzuzeigende Bereich ist und man kann den gezeigten Bereich verschieben und so die gesamte Matrix sichten.





4.3 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.

$$\text{findrows}\left(\begin{bmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{bmatrix};4;2\right)=\begin{bmatrix}3&4\\3&4\end{bmatrix}\quad\text{findrows}\left(\begin{bmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{bmatrix};5;1\right)=\begin{bmatrix}5&6\end{bmatrix}$$

$$\text{findrows}\left(\begin{bmatrix}1&2\\3&4\\1&2\\3&4\\5&6\end{bmatrix};7;1\right)=0$$

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

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

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

$$\text{rsort}\left(\begin{bmatrix}1&3&1&3&5\\2&4&7&8&6\end{bmatrix};1\right)=\begin{bmatrix}1&1&3&3&5\\2&7&4&8&6\end{bmatrix}$$

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

$$\text{sort}\left(\begin{bmatrix}1\\3\\1\\3\\5\end{bmatrix}\right)=\begin{bmatrix}1\\1\\3\\3\\5\end{bmatrix}$$

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

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

Man kann dies benutzen, um absteigend zu sortieren:

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

4.4 Eindimensionale Interpolation

SMath bietet eingebaute Funktionen für die eindimensionale Interpolation. Das bedeutet, dass aus zwei Vektoren für X und Y ein y-Wert für einen gegebenen x-Wert berechnet wird.

- `ainterp(vx; vy; x)` kubische Spline-Interpolation nach Akima[1]. 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

`vx`	x-Daten (Vektor)
`vy`	y-Daten (Vektor)
`x`	x-Wert

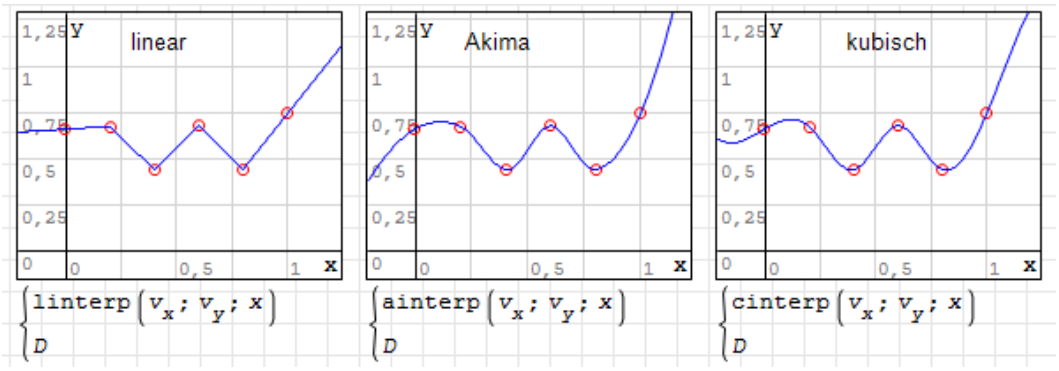
Beispiel: Interpolation von Zufallszahlen im Bereich von 0 bis 1

$$v_x := [0; 0{,}2 .. 1] \qquad v_y := \text{Random}(\text{length}(v_x))$$

$$v_x = \begin{bmatrix} 0 \\ 0{,}2 \\ 0{,}4 \\ 0{,}6 \\ 0{,}8 \\ 1 \end{bmatrix} \quad v_y = \begin{bmatrix} 0{,}6559 \\ 0{,}6709 \\ 0{,}4379 \\ 0{,}6734 \\ 0{,}44 \\ 0{,}7397 \end{bmatrix} \qquad \text{linterp}(v_x; v_y; 0{,}3) = 0{,}5544$$

$$D := \text{augment}(v_x; v_y; \text{"o"}; 10; \text{"red"})$$

Grafische Darstellung der drei Interpolationsarten zusammen mit den Datenpunkten:



[1] Erläuterung des Verfahrens (TU Wien): `https://www.ads.tuwien.ac.at/docs/lva/mmgdv/k1___011.htm`

4.5 Zweidimensionale Interpolation

Für die Interpolation von zweidimensionalen Tabellendaten steht die Funktion `InterpBilinear()` zur Verfügung (Plugin *Custom Functions*). 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.

$$X := [20\ 50\ 100\ 150\ 200\ 250]$$

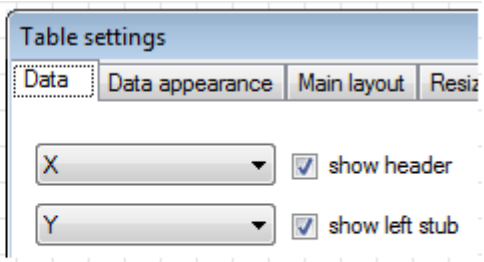
$$Y := \begin{bmatrix} 15 \\ 20 \\ 25 \\ 40 \\ 50 \end{bmatrix} \quad M := \begin{bmatrix} 4 & 9 & 17 & 25 & 35 & 45 \\ 4 & 10 & 19 & 28 & 39 & 50 \\ 5 & 11 & 20 & 31 & 42 & 54 \\ 5 & 12 & 23 & 35 & 47 & 60 \\ 6 & 14 & 26 & 38 & 51 & 66 \end{bmatrix}$$

$$\text{InterpBilinear}(X; Y; M; 30; 40) = 7{,}3333$$

Die Matrix und die Vektoren können mit dem Tabellenbereich (Plugin *Table Region*) übersichtlich 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 Diagramme

Inhalt

5.1 X-Y Plot 32
5.2 Einstellungen und Mausfunktionen 33
5.3 Achsen, Beschriftungen, Legende 34
5.4 Datenreihen darstellen 35
5.5 Achsenbereiche automatisch steuern 36
5.6 Explizite Funktionen darstellen 38
5.7 Implizite Funktionen darstellen 39
5.8 Textmarken darstellen 40
5.9 Plotten mit Maßeinheiten 41
5.10 Geometrische Grundformen 43
5.11 Diagramme mit Maxima 45

Diagramme kennen Sie schon aus der Einführung. Hier gehen wir genauer auf die Darstellung von Daten ein.

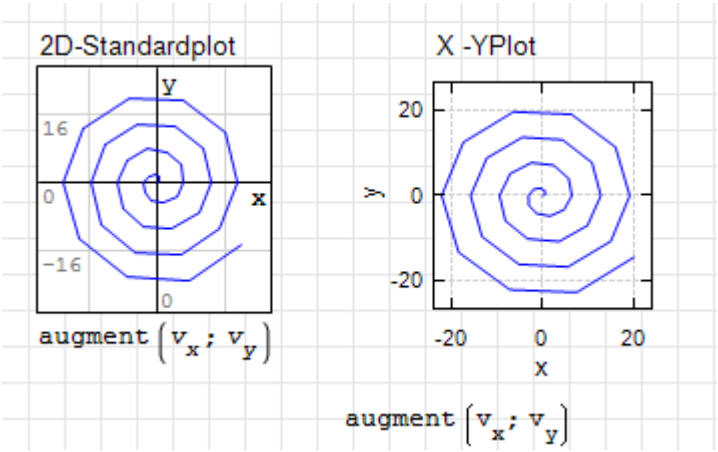
Das Datenformat für den Standard-2D-Plot und den X-Y Plot (Plugin *X-Y Plot Region*) sind sehr ähnlich. Wir konzentrieren uns hier auf den X-Y Plot, der folgende Möglichkeiten bietet:

- Darstellung expliziter Funktionen $f(x)$
- Darstellung impliziter Funktionen $f(x, y) = 0$
- Darstellung zweispaltiger Matrizen (zeilenweise x, y-Koordianten) als Linienzüge oder Punkte dargestellt.
- Matrizen der Form $\begin{bmatrix} x & y & \text{Text} & \text{Größe} & \text{Farbe} \end{bmatrix}$ stellen Text an der Position x, y dar.
- Es gibt eine Bibliothek geometrischer Grundformen,
- Titel, Achsbeschriftungen und Legende sowie Linien- und Datenpunktformat sind frei formatierbar (im Einstellungsmenü)
- Achsengrenzen, Gitterweite und andere Merkmale sind über Variablen vom Rechenblatt aus steuerbar (wichtig für automatische Bereichsanpassung an Daten).

Gegenüber dem Standard-2D-Plot fehlt hauptsächlich die Möglichkeit, das Diagramm zu animieren.

Gegenüber dem ZedGraph-Plot (Plugin *ZedGraph Region*) und Maxima fehlt die Option für logarithmische Achsen.

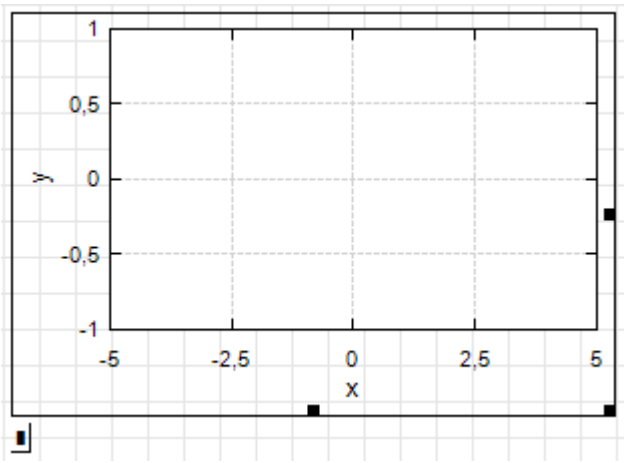
Einige visuelle Unterschiede zeigt das folgende Bild.



Mit dem Maxima-Plugin kann man auch Diagramme erstellen (wir nutzen in dieser Anleitung Histogramme und Boxplots sowie logarithmische Achsen. Eine umfassende Dokumentation zu Grafik mit Maxima findet sich in Kraska (2020).

5.1 X-Y Plot

Ein X-Y Plot-Bereich kann über das **Einfügen**> **Diagramme**> **X-Y Plot** eingefügt werden.



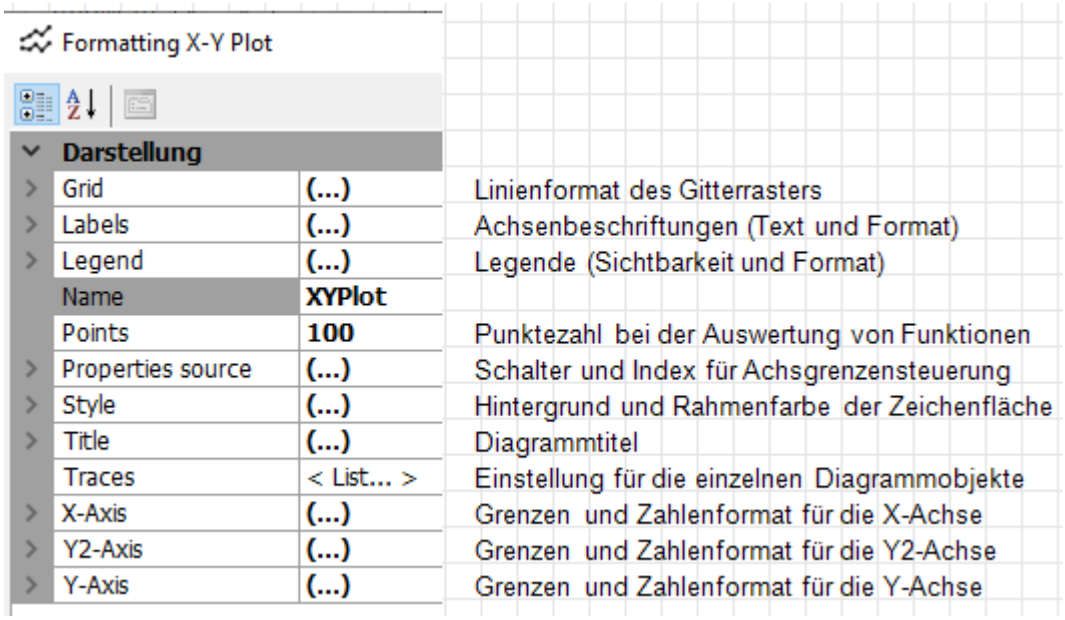
In den Platzhalter schreibt man einen Ausdruck oder eine Liste von Ausdrücken.

Der X-Y Plot kann folgende Objekte darstellen:

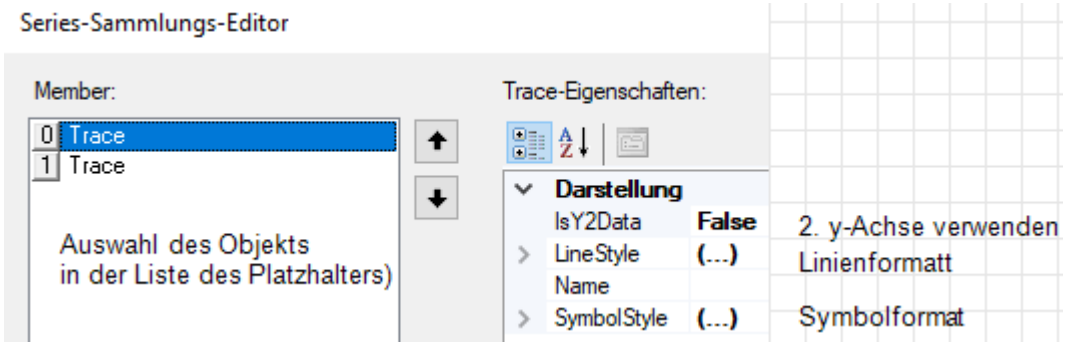
- Ausdrücke mit einer freien (undefinierten) Variable $f(x)$ werden als Kurve gezeichnet
- Ausdrücke mit zwei freien Variablen $f(x, y)$ werden als implizite Funktion $f(x, y) = 0$ gezeichnet
- Zweispaltige Matrizen werden zeilenweise als x, y-Koordianten betrachtet und als Linienzüge oder Punkte dargestellt.
- Matrizen der Form $\begin{bmatrix} x & y & \text{Text} & \text{Größe} & \text{Farbe} \end{bmatrix}$ stellen Text an der Position x, y dar.
- Es gibt eine Bibliothek geometrischer Grundformen,
- Titel, Achsbeschriftungen und Legende sowie Linien- und Datenpunktformat sind frei formatierbar (im Einstellungsmenü)
- Achsengrenzen, Gitterweite und andere Merkmale sind über Variablen vom Rechenblatt aus steuerbar (wichtig für automatische Bereichsanpassung an Daten).

5.2 Einstellungen und Mausfunktionen

Doppelklick auf das Diagramm oder Rechte Maustaste> Format... öffnet den Dialog „Formattting X-Y Plot"



Unter „Traces" gibt es einen Unterdialog, in dem die Eigenschaften der Diagrammobjekte (die Elemente der Liste im Platzhalter) gesetzt werden können.

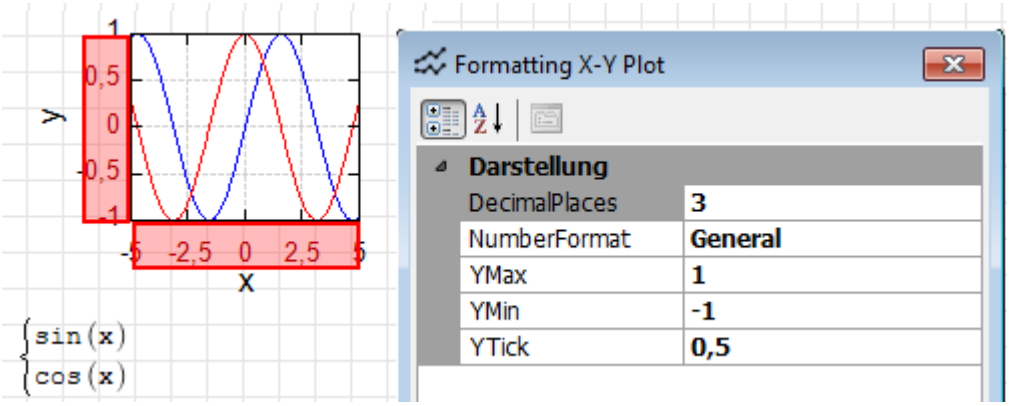


Mausfunktionen im X-Y Plot

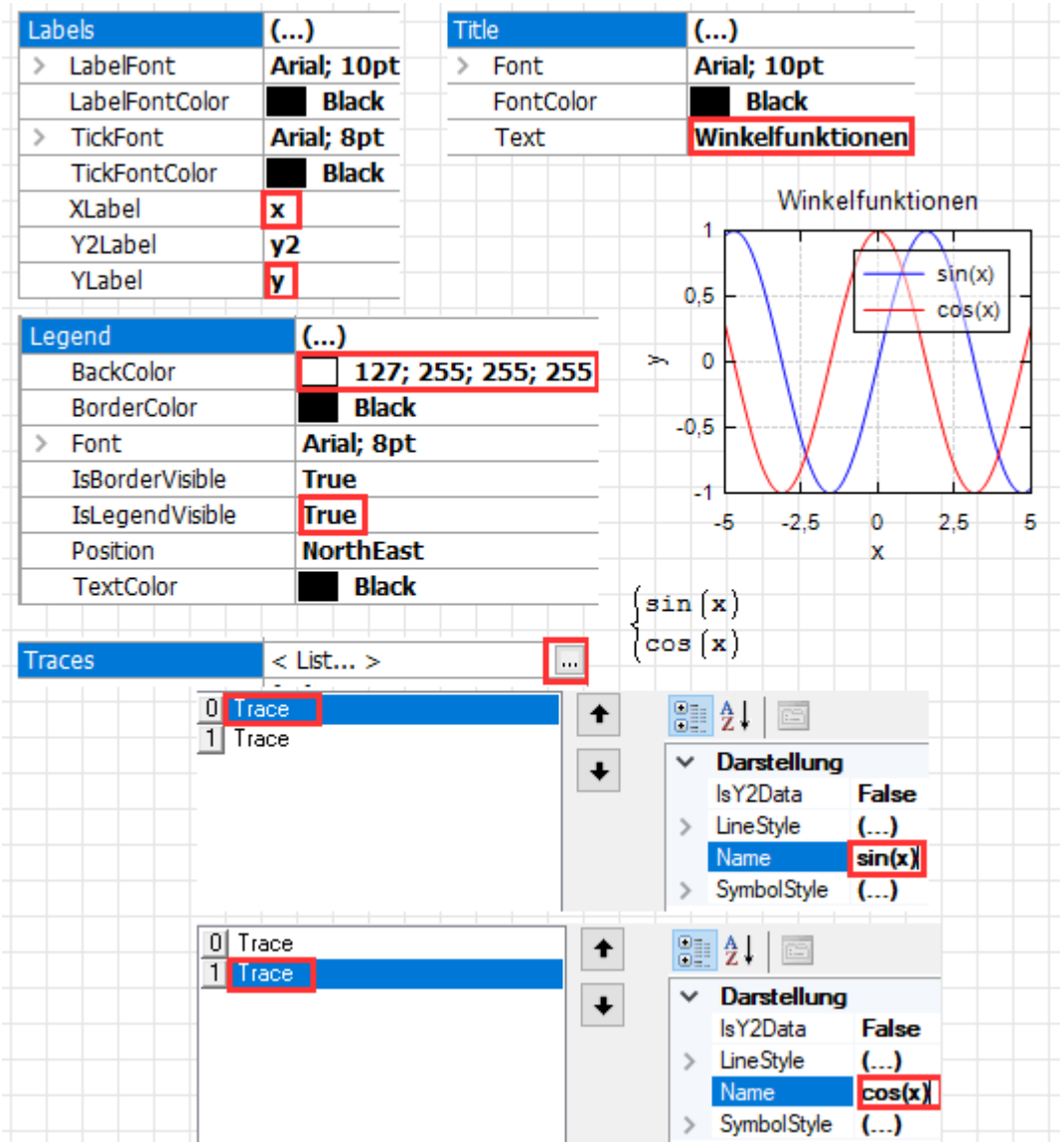
- **Doppelklick links**: Öffnen des Format-Editors (alternativ zu **Kontextmenü> Format**)
- **Klick rechts**: Öffnen des Kontextmenüs
- **Ziehen an den Randmarken**: Größe der Diagrammfläche ändern, dabei bleiben die Achsgrenzen erhalten, d.h. die Achsen werden entsprechend gedehnt oder gestaucht.
- **Mausrad**: Skalierung der Achsen (x und y gleichzeitig). Eine eventuell aktive y2-Achse ist davon nicht betroffen.
  - **Shift-Mausrad**: Nur x-Achse zoomen
  - **Strg-Mausrad**: Nur y-Achse zoomen
- **Ziehen der Diagrammfläche**: Achsen verschieben.

5.3 Achsen, Beschriftungen, Legende

Doppelklick auf die Achsmarken (*ticks*) öffnet einen direkten Einstellungsdialog für den Wertebereich und den Gitterabstand.



Doppelklick auf das Diagramm öffnet den vollständigen Einstellungsdialog. Die wichtigsten Einstellungen sind Achsentitel, Diagrammtitel, und Legendeneinträge. Die Legendenfarbe `127,255.255,255` ist ein halbtransparentes Weiß. Transparenz ist sinnvoll, wenn die Legende über den Kurven liegt.



5.4 Datenreihen darstellen

Das Datenformat für den Plot von Datenpunkten sind zweispaltige Matrizen. Jede Zeile stellt einen Datenpunkt dar.

Die Matrizen können mit `augment()` aus zwei gleichlangen Spaltenvektoren kombiniert werden.

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \end{bmatrix}$$

Aus einer mehrspaltigen Datenmatrix D kann man z.B. so die 3. über der 1. Spalte plotten:

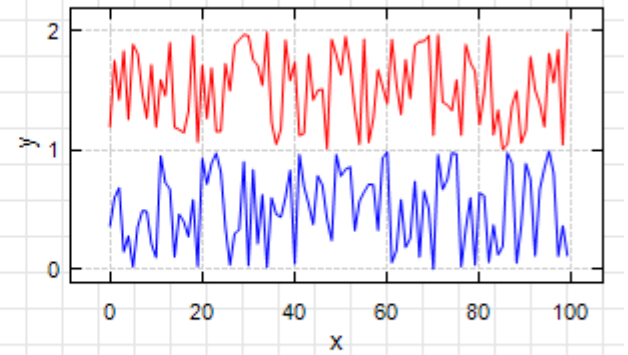
```
augment(col(M;1);col(M;3))
```

Mehrere Datensätze können durch Listen kombiniert werden. Standardmäßig nehmen die Kurven dann die Farben blau, rot, grün, magenta, orange und braun an.

$n := 100 \quad x := [0..(n-1)] \quad y_1 := \text{Random}(n) \quad y_2 := \text{Random}(n) + 1$

$$\text{augment}(x; y_1) = \begin{bmatrix} 0 & 0{,}3621 \\ 1 & 0{,}5996 \\ 2 & 0{,}6829 \\ 3 & 0{,}1453 \\ \vdots & \end{bmatrix}$$

$$\text{augment}(x; y_2) = \begin{bmatrix} 0 & 1{,}1952 \\ 1 & 1{,}7514 \\ 2 & 1{,}4206 \\ 3 & 1{,}8326 \\ \vdots & \end{bmatrix}$$

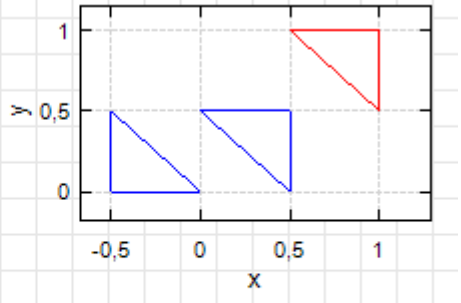


$$\begin{cases} \text{augment}(x; y_1) \\ \text{augment}(x; y_2) \end{cases}$$

Sollen die Datensätze gleichfarbig sein, fasst man die Matrizen zu einem Spaltenvektor zusammen. Im Beispiel hier werden Matrix M_1und M_2 in einen Zeilenvektor geschrieben, der dann transponiert wird. So spart man Platz.

Noch mehr Platz spart man, wenn man die Definitionen der Matrizen in einer Liste zusammenfasst. Dann sind nämlich alle Definitionen oben links schon bekannt.

$$\begin{cases} M_1 := \begin{bmatrix} 0 & 0 \\ -0{,}5 & 0 \\ -0{,}5 & 0{,}5 \\ 0 & 0 \end{bmatrix} \\ M_2 := \begin{bmatrix} 0 & 0{,}5 \\ 0{,}5 & 0{,}5 \\ 0{,}5 & 0 \\ 0 & 0{,}5 \end{bmatrix} \\ M_3 := \begin{bmatrix} 0 & 0{,}5 \\ 0{,}5 & 0{,}5 \\ 0{,}5 & 0 \\ 0 & 0{,}5 \end{bmatrix} + 0{,}5 \end{cases} \quad \begin{cases} [M_1 \; M_2]^T \\ M_3 \end{cases}$$



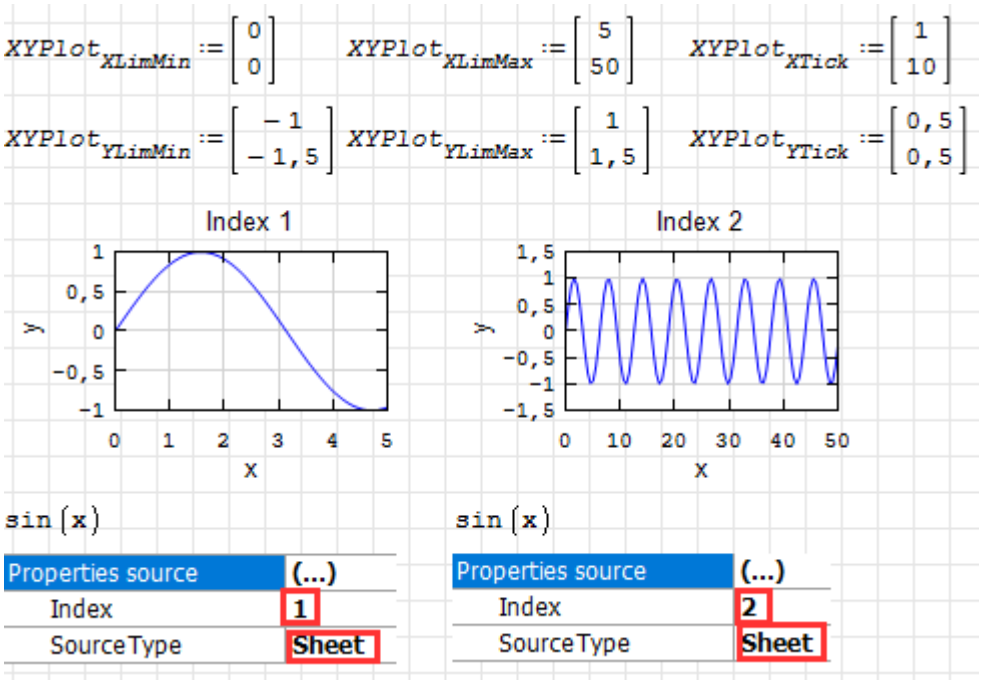
5.5 Achsenbereiche automatisch steuern

Gerade bei der Datenanalyse ist es vorteilhaft, wenn sich die Achsgrenzen automatisch den Daten anpassen.

Im X-Y Plot kann man Minimalwert, Maximalwert und Gitterschrittweite über spezielle Variable steuern.

- Die speziellen Variablen werden mit Werten versorgt. Diese Werte sind Spaltenvektoren, ihre Länge entspricht der Zahl der Diagramme, die Sie steuern wollen.
- Im Einstellungsmenü setzen Sie
  - **Properties source**> **Source Type** auf „Sheet" (Steuerung über das Rechenblatt)
  - **Properties source**> **Index** auf auf den Index in den Wertevektoren, den Sie für das aktuelle Diagramm verwenden wollen.

Im folgenden Beispiel wird das linke Diagramm von den ersten Vektorelementen gesteuert, das rechte von den zweiten.



Auf der nächsten Seite folgt noch ein Beispiel, wo die Achsgrenzen tatsächlich aus den Daten berechnet werden

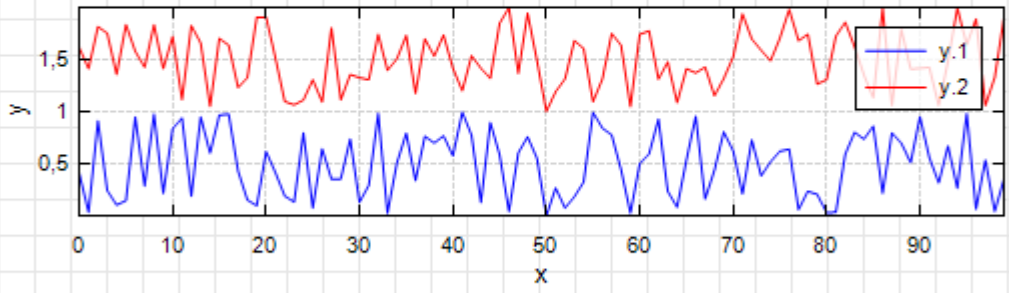
Hier ein Beispiel, wo die Achsgrenzen tatsächlich aus den Daten berechnet werden. Die Vektoren haben hier nur ein Element, weil nur ein Diagramm gesteuert werden soll.

$n := 100 \quad x := [0..(n-1)] \quad y_1 := \text{Random}(n) \quad y_2 := \text{Random}(n) + 1$

Achsengrenzen für Diagramm automatisch setzen

$XYPlot_{XLimMin} := [\min(x)] \quad XYPlot_{XLimMax} := [\max(x)] \quad XYPlot_{XTick} := [10]$

$XYPlot_{YLimMin} := [\min(y_1)] \quad XYPlot_{YLimMax} := [\max(y_2)] \quad XYPlot_{YTick} := [0,5]$



$\begin{cases} \text{augment}(x; y_1) \\ \text{augment}(x; y_2) \end{cases}$

Hier haben wir einen Blattbereich (**Einfügen> Blattbereich**) genutzt. Diesen kann man zusammenklappen, wenn man bestimmte Details einer Rechnung für bessere Übersicht ausblenden will. Klicken auf [-] am linken Rand blendet alles aus, was zwischen den beiden Linien des Bereichs liegt.

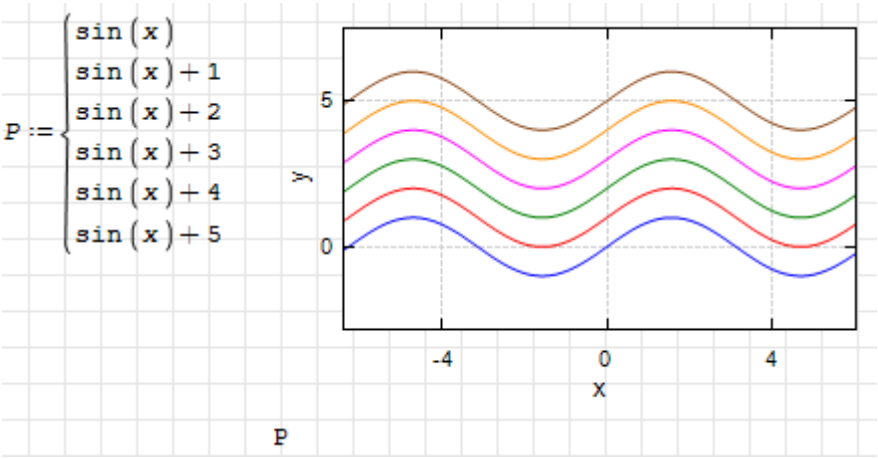
Achsengrenzen für Diagramme

5.6 Explizite Funktionen darstellen

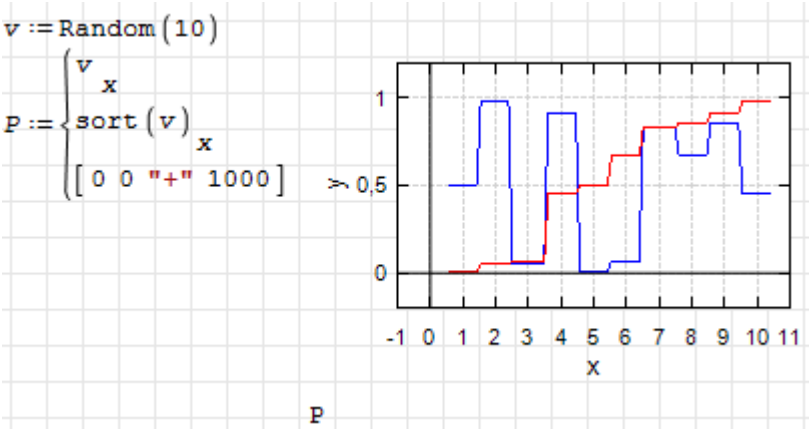
Explizite Funktionen sind Ausdrücke mit einer freien (nicht definierten) Variable, die den Funktionswert direkt liefern. Mitunter müssen die Funktionen vorher definiert werden, können also nicht direkt in den Platzhalter geschrieben werden.

Die Variable durchläuft die Werte der x-Achse.

Mehrere Funktionen können wie im Standard-Diagrammbereich durch Listen kombiniert werden. Standardmäßig nehmen die Kurven dann die Farben blau, rot, grün, magenta, orange und braun an.



Die unabhängige Variable kann auch als Index an Matrizen und Vektoren benutzt werden.

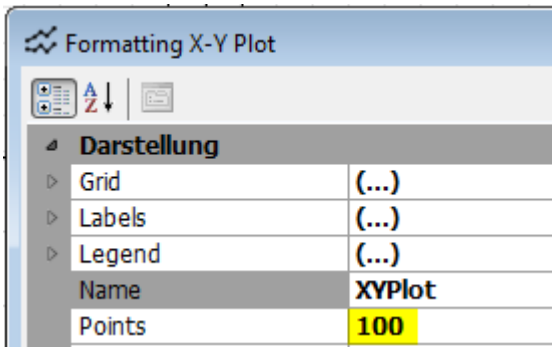


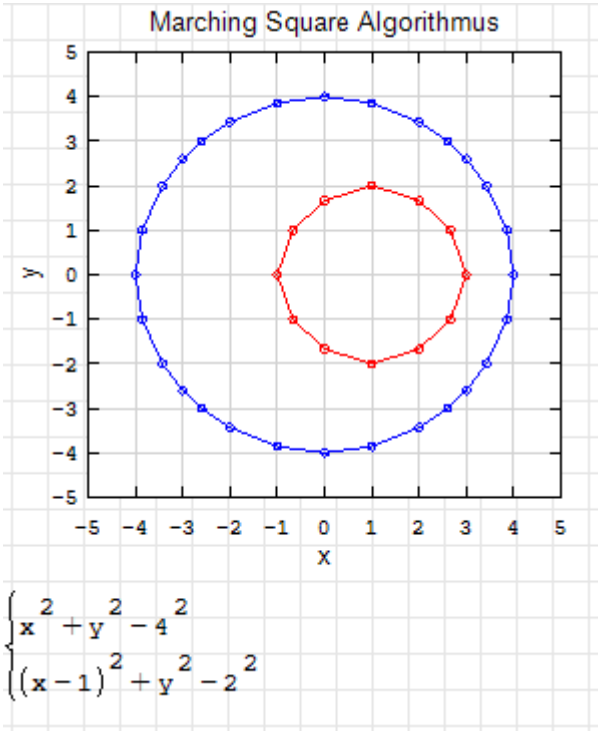
Hier sieht man auch, wie ein Achsenkreuz erzeugt werden kann. Es besteht aus einem + Symbol, das ausreichend groß skaliert wird.

5.7 Implizite Funktionen darstellen

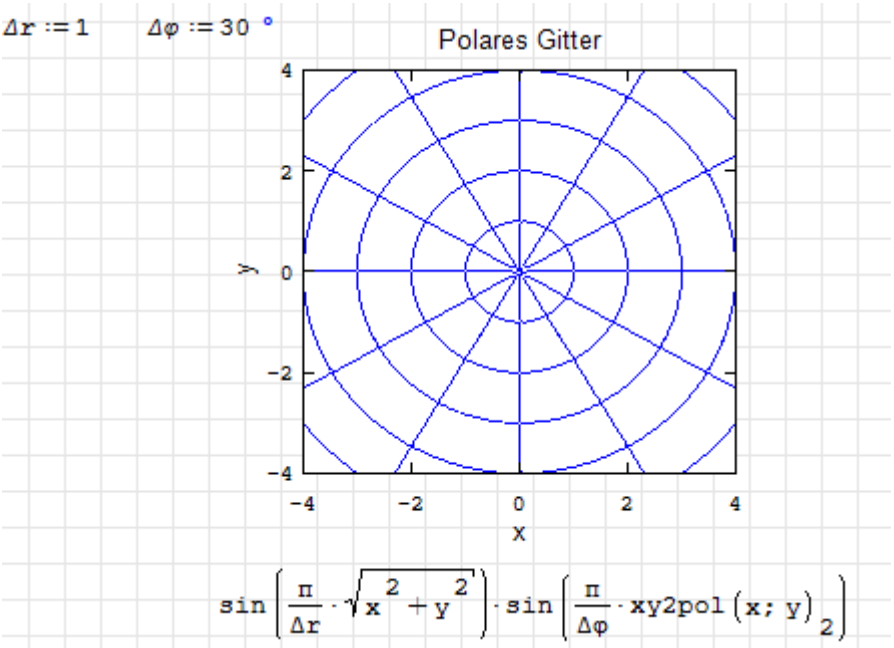
Implizite Funktionen erkennt der X-Y Plot-Bereich an zwei darin vorkommenden undefinierten Variablen.

Die Funktion ist als Ausdruck mit zwei undefinierten Variablen zu definieren, wobei die erste (für die x-Achse) x oder t heißen muss. Die Nullstellen dieses Ausdrucks werden dargestellt. 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.



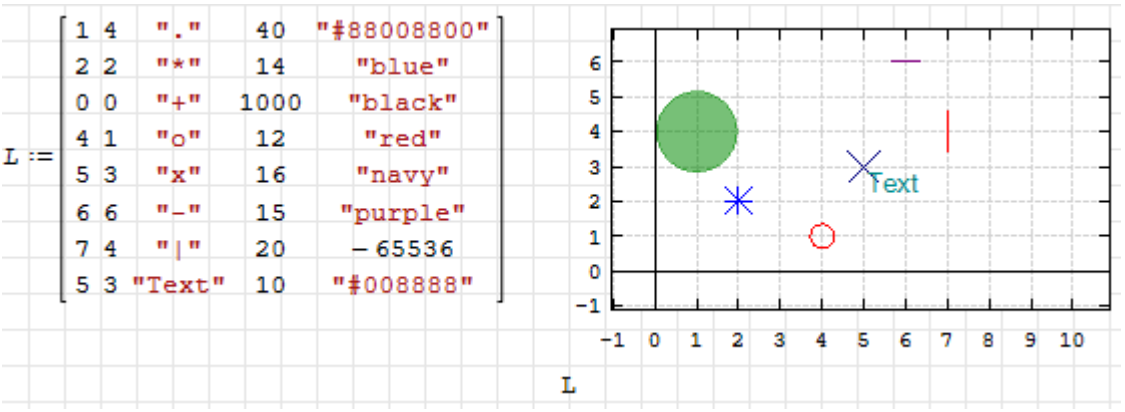
5.8 Textmarken darstellen

Einzelne Zeichen oder Textzeilen können ähnlich wie beim Standard-Diagrammbereich frei platziert 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 (optional, Einheit Punkt)
Farbe	Textfarbe (optional)

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 Diagrammen verwendet werden.



Wenn die Farbspalte fehlt, kann die Farbe im Formatdialog geändert werden (für alle in einer Matrix definierten Texte gleichzeitig)

Wenn auch die Größenspalte fehlt, ist für die Zeichen `. * + o x - |` die Einstellung im Formatdialog wirksam, ansonsten hat der Text dann Standardgröße.

Farben: Es gibt mehrere Wege, Farben für Textobjekte in Diagrammen zu spezifizieren:

1. Zeichenkette mit HTML-Farbnamen[1], 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, die man auch im Einstellungsdialog des X-Y Plots findet.t. Die Namen werden intern von Leerzeichen befreit und in Kleinbuchstaben umgewandelt, `"darkblue"`, `"DarkBlue"`, `"DARKBLUE"` und `"dark blue"` bezeichnen also alle die gleiche Farbe.

[1] siehe z.B bfw.ac.at/020/farbtabelle.html

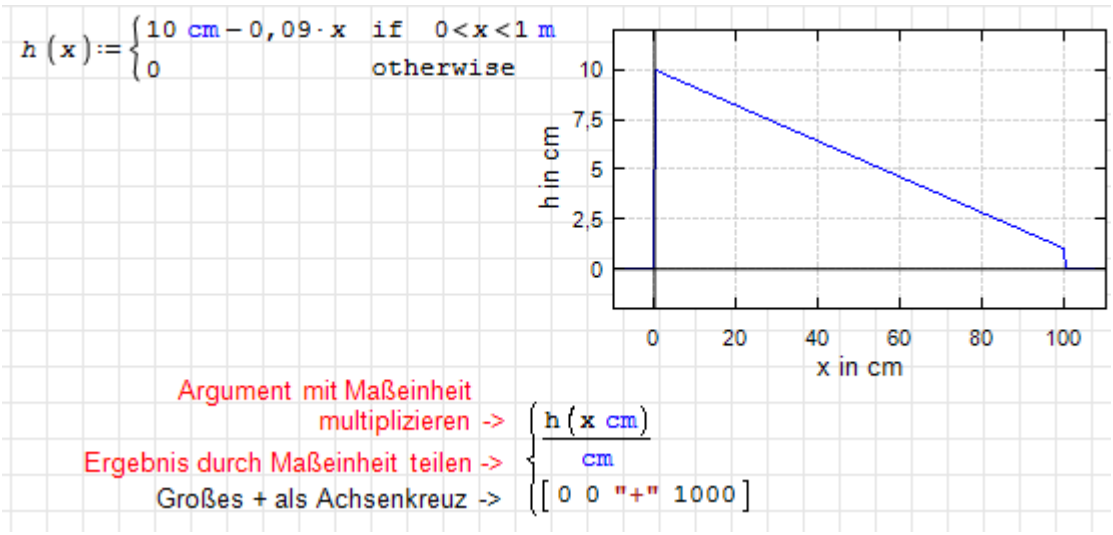
5.9 Plotten mit Maßeinheiten

Wenn Größen mit Maßeinheiten dargestellt werden sollen, ist bei allen Diagrammtypen (SMath Standard, X-Y Plot und Maxima) folgendes zu beachten:

- Die Diagrammkoordinaten sind dimensionslos.
- Funktionen $f(x)$ oder $f(x, y)$ erhalten ihre Argumente ohne Einheiten. Wenn die Funktion Argumente mit Einheiten verlangt, muss man die freien Variablen mit der gewünschten Einheit multiplizieren.
- Funktionswerte oder Koordinaten in Datenmatrizen mit Einheiten müssen durch die gewünschten Einheiten geteilt werden. Anderenfalls werden sie in in Basiseinheiten dargestellt.
- Der Anwender muss selbst sicherstellen, dass die Achsbeschriftungen zu den verwendeten Einheiten passen.

Im folgenden Beispiel erwartet die Funktion $h(x)$ das Argument mit einer Längeneinheit. Daher muss das x im Diagramm mit der gewünschten Einheit (hier cm) multipliziert werden.

Wenn die Funktionswerte in cm geplottet werden sollen, dann muss der Ausdruck durch diese Einheit geteilt werden.

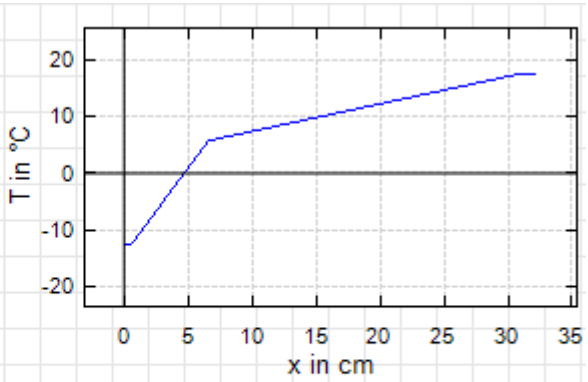


Seit 2019 funktioniert das auch für nicht-absolute Temperatureinheiten. Vorher musste man die Nullpunktverschiebung explizit rückgängig machen:

$$X := \begin{bmatrix} 0 \\ 0,6 \\ 6,6 \\ 30,6 \\ 32,1 \end{bmatrix} \text{cm} \quad T := \begin{bmatrix} -12,6 \\ -12,5 \\ 5,69 \\ 17,3 \\ 17,6 \end{bmatrix} \text{°C}$$

Vor 2019 $\dfrac{T_1 - 0\ \text{°C}}{\Delta\text{°C}} = -12,6$

Neu 2019 $\dfrac{T_1}{\text{°C}} = -12,6$



augment $\left(\dfrac{X}{\text{cm}}, \dfrac{T}{\text{°C}}\right)$

[0 0 "+" 1000]

5.10 Geometrische Grundformen

Im X-Y Plot-Bereich sind einige Grundformen definiert, bei denen man Position, Größe, Linienart, Linienstärke, Linienfarbe und Füllfarbe vorgeben kann. Die Grundformen werden durch Spaltenvektoren definiert (hier als transponierter Zeilenvektor geschrieben):

$$\begin{bmatrix} \text{Typ} & \text{Punkte} & \text{Linienfarbe} & \text{Linienart} & \text{Linienstärke} & \text{Füllfarbe} \end{bmatrix}^T$$

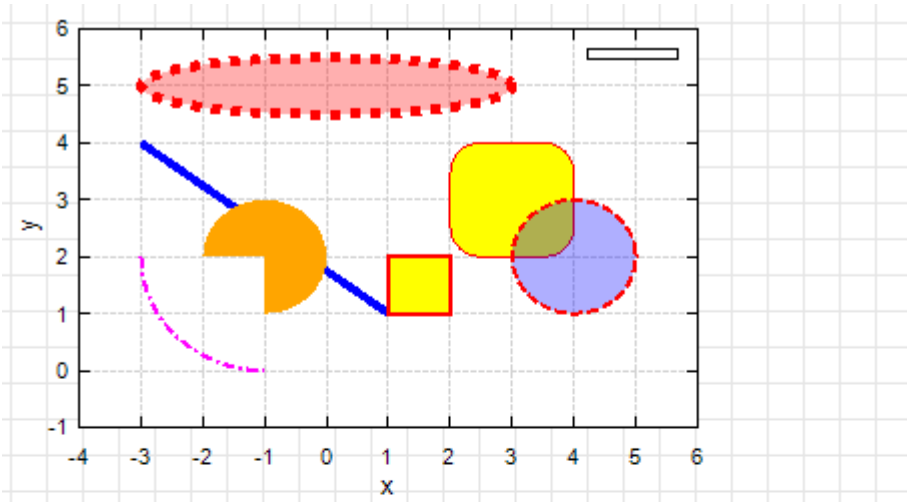
Typ Schlüsselwort für die Grundform

Punkte Koordinatenmatrix oder -vektor, abhängig vom Typ

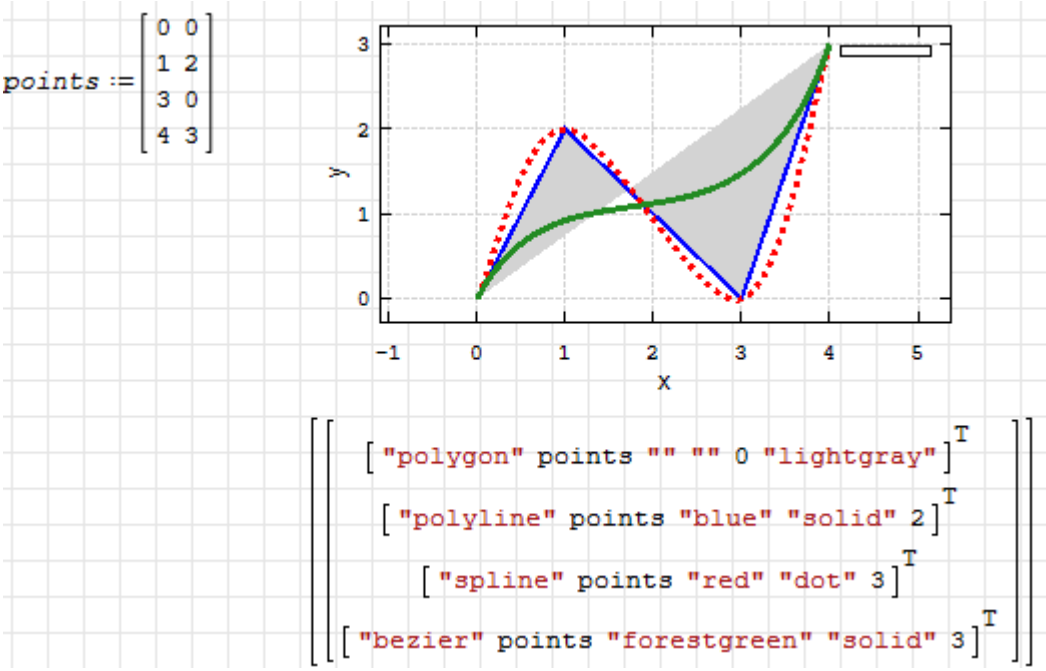
Linienfarbe Zeichenkette oder Zahl. 0 bedeutet: Keine Linie

Linienart Schlüsselwort: `"solid"`, `"dash"`, `"dot"`, `"dashdot"` oder `"dashdotdot"`

Füllfarbe Zeichenkette oder Zahl. Fehlt die Angabe, bleibt das Objekt ungefüllt




```
[["line" [1 1 -3 4]^T "blue" "solid" 4]^T
 ["rect" [1 1 1 1]^T "red" "solid" 2 "yellow"]^T
 ["roundrect" [2 2 2 2 0,5]^T "red" "solid" 1 "yellow"]^T
 ["circle" [4 2 1]^T "red" "dash" 2 "#500000ff"]^T
 ["ellipse" [0 5 3 0,5]^T "red" "dot" 5 "#50ff0000"]^T
 ["arc" [-1 2 2 2 180° 90°]^T "magenta" "dashdot" 2]^T
 ["pie" [-1 2 1 1 (-90)° 270°]^T "" "" 0 "orange"]^T]
```



Objekt	Koordinatenmatrix	Füllung
`line`	$\begin{bmatrix} x_1 & y_1 & x_2 & y_2 \end{bmatrix}^T$	
`rect`	$\begin{bmatrix} x_1 & y_1 & \Delta x & \Delta y \end{bmatrix}^T$	x
`roundrect`	$\begin{bmatrix} x_1 & y_1 & \Delta x & \Delta y & r \end{bmatrix}^T$	x
`circle`	$\begin{bmatrix} x_c & y_c & r \end{bmatrix}^T$	x
`ellipse`	$\begin{bmatrix} x_c & y_c & r_x & r_y \end{bmatrix}^T$	x
`arc`	$\begin{bmatrix} x_c & y_c & r_x & r_y & \varphi_1 & \varphi_2 \end{bmatrix}^T$	
`pie`		x
`polygon`	$\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix}^T$	x
`polyline`		
`spline`		
`bezier`	$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix}^T$	

Tabelle 5.1: Belegung der Koordinatenmatrizen in X-Y Plot-Geometrieobjekten

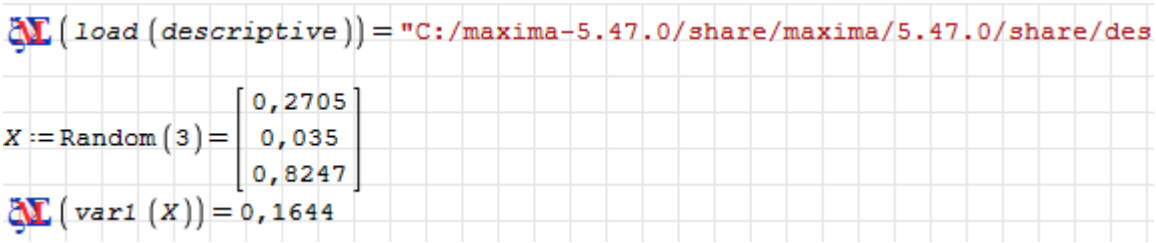
5.11 Diagramme mit Maxima

Das *Maxima*-Plugin macht dem SMath-Anwender das Maxima-Paket *descriptive* zugänglich. Dieses stellt neben Statistikfunktionen auch die Diagrammtypen Boxplot und Histogramm bereit.

Installation: Das *Maxima*-Plugin wird wie jedes andere Plugin über **Extras> Plugins** installiert. Wichtig dabei: Umschalten von „Lokale Speicherung" auf „Online Galerie".

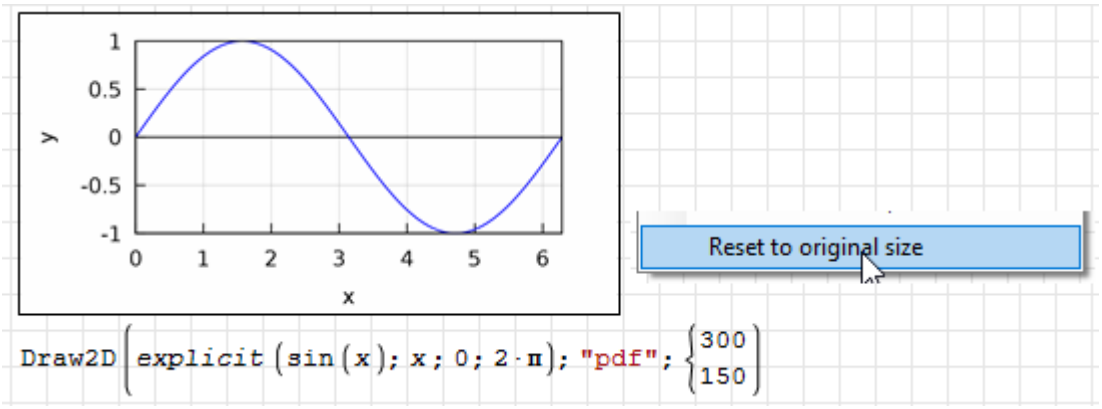
Das Plugin benötigt eine Standard-Maxima-Installation auf dem Rechner. Unter **Einfügen> Maxima> Setting** können Sie entweder eine vorhandene Installation wählen (Pfad dorthin angeben) oder Maxima installieren lassen.

`Maxima()`:Die wichtigste Funktion ist `)`, welche eine blau-rote Operatorform annimmt. Sie sendet Ausdrücke zu Maxima und liefert deren Ergebnis zurück.



`Draw2D()` **und** `Draw3D()`: Die Funktionen erzeugen Bilddateien. Diese kann man mit dem Image-Bereich (aus dem Plugin *ImageRegion*) darstellen.

1. Man erzeugt also mit **Einfügen> Image** einen solchen Bereich.
2. In dessen Platzhalter schreibt man die Grafikfunktion mit den passenden Grafikbefehlen. Diese werden im SMath-Handbuch Kraska (2020) beschrieben.



Als drittes Argument der Draw-Funktion kann man die Größe (Einheit: Punkte) angeben. Den Image-Bereich kann man mit der Maus in der Größe verändern, dann wird das Bild skaliert.

Besser ist es, gleich die passende Größe zu erzeugen. Mit **Kontextmenü> Reset to original size** kann man die Größe des Image-Bereichs an die Grafikdatei anpassen

Beispiele für Maxima-Diagramme in dieser Anleitung:

- Boxplots (Abschnitt 7.3) und Histogramme (Abschnitt 7.4)
- Diagramm mit logarithmischer x-Achse (Abschnitt 9.3)

6 Tabellen

Inhalt

6.1 Tabellen erzeugen 47
6.2 Zeilen- und Spaltentitel 48
6.3 Maßeinheiten in Tabellen 49

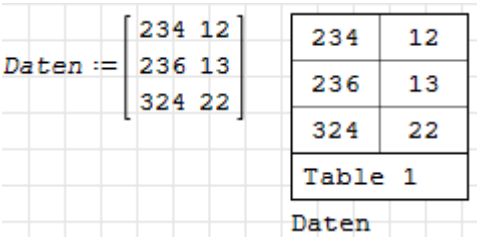
Das Plugin *TableRegion* stellt einen Tabellenbereich bereit, der für die Anzeige von Text, Zahlen oder SMath-Formeln genutzt werden kann. Besondere Stärken sind:

- Umgang mit Maßeinheiten,
- Zahlenformatierung,
- Darstellbarkeit von Formeln oder Symbolen wie im Formelbereich.

6.1 Tabellen erzeugen

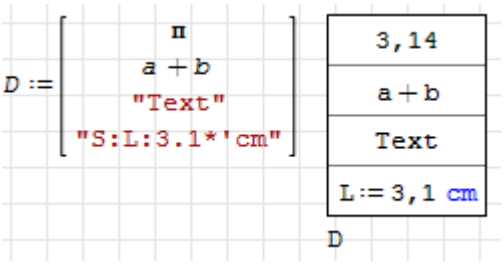
Eine Tabelle wird eingefügt mit **Einfügen> Table** oder mit `Shift-Ctrl-T`

In den Platzhalter trägt man eine Datenmatrix ein.

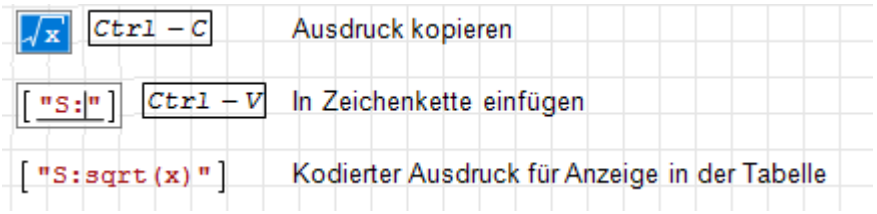


Die Elemente der Datenmatrix können verschiedene Typen haben, auch gemischt:

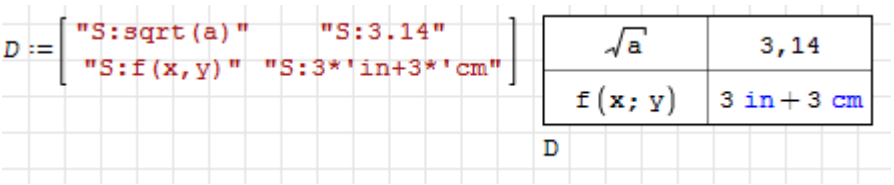
- reelle oder komplexe numerische Werte
- symbolische Ausdrücke (wenn der numerische Wert nicht definiert ist)
- Text (Zeichenketten)
- Text mit kodierten symbolischen Ausdrücken. Diese Ausdrücke werden nicht ausgewertet, nur angezeigt. Sie haben die Form `"S:..."`



Anzeige von Ausdrücken: Die Kodierung der symbolischen Ausdrücke entspricht Darstellung von Ausdrücken in der Zwischenablage. Wenn diese unbekannt ist, kopiere man einen Ausdruck in die Zwischenablage und füge den Inhalt in die Zeichenkette `"S:"` nach dem Doppelpunkt ein.



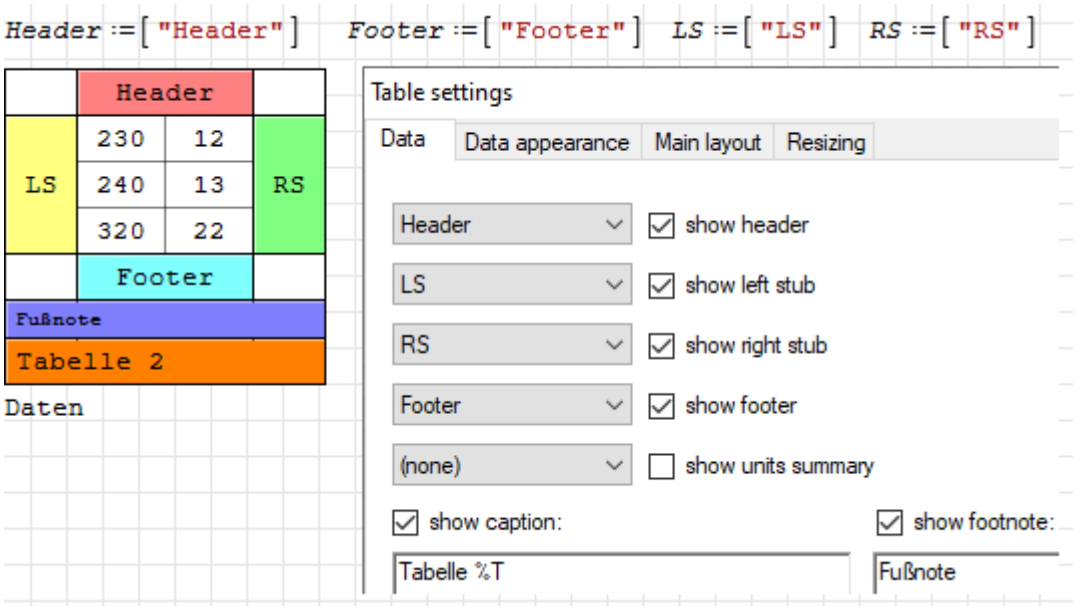
Die Zwischenablage kümmert sich nicht um die Voreinstellungen zu Dezimaltrennzeichen und Argumenttrennzeichen. Bei der Anzeige in der Tabelle wird das aber wieder beachtet. Hier einige Beispiele:



6.2 Zeilen- und Spaltentitel

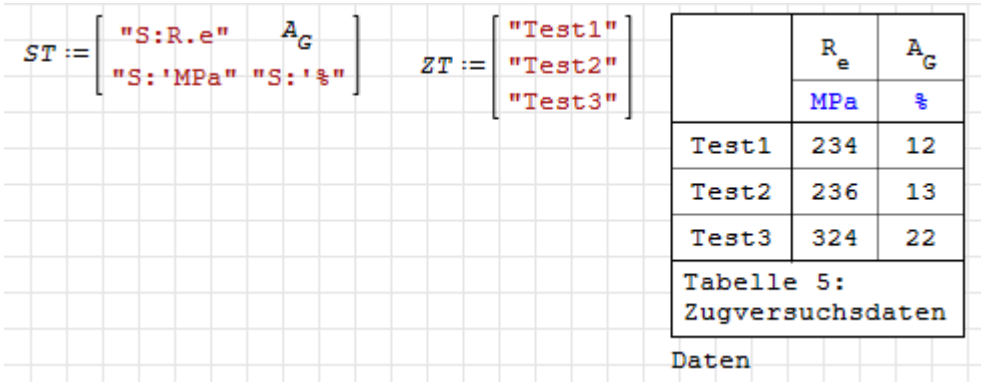
Rundherum um die Datentabelle (*body*) kann man Zeilen- oder Spaltentitel ergänzen. Das sind im einfachsten Fall 1×1-Matrizen. Die Zuordnung erfolgt im Einstellungsmenü (Table settings). Dies ist zu erreichen unter Kontextmenü (rechte Maustaste) oder durch Doppelklick. Die ansprechbaren Elemente sind:

- *header* (oben, für Spaltentitel)
- *left stub* (links, für Zeilentitel)
- *right stub* (rechts, für Zeilentitel)
- *footer* (unten, für Spaltentitel)
- *note* (Fußnote)
- *caption* (Unterschrift)



Die Farben wurden in der Einstellungsseite „Data appearance" gesetzt.

Hier die gleiche Tabelle mit sinnvollen Zeilen- und Spaltentiteln:



Die Zeilentitel-Matrix muss die gleiche Zeilenzahl wie die Datenmatrix haben (oder nur eine Zeile).

Die Spaltentitel-Matrix muss die gleiche Spaltenzahl wie die Datenmatrix haben (oder nur eine Spalte).

6.3 Maßeinheiten in Tabellen

Daten mit Maßeinheiten werden immer in Basiseinheiten gespeichert und erst bei der Anzeige in die gewünschte Einheit umgerechnet.

In Tabellen wird zunächst erst mal die rohe gespeicherte Form angezeigt:

$F := \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ N $\quad h := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ cm $\quad M := \overrightarrow{F \cdot h} = \begin{bmatrix} 2 \\ 6 \\ 12 \end{bmatrix}$ N cm

2 (kg m)/(s^2)	0.01 m	0.02 (kg m^2)/(s^2)
3 (kg m)/(s^2)	0.02 m	0.06 (kg m^2)/(s^2)
4 (kg m)/(s^2)	0.03 m	0.12 (kg m^2)/(s^2)
Table 6		

D := augment(F; h; M)

Das ist nicht gut leserlich. Daher haben Tabellen in SMath einen Mechanismus, die Einheiten zeilen- oder spaltenweise vorzugeben.

Man definiert einen Vektor mit einer Maßeinheit für jede Spalte und ordnet sie im Einstellungsdialog dem „unit summary" zu.

Units := [N cm N cm] Units ☑ show units summary

1 (kg m)/(s^2)	0.01 m	0.01 (kg m^2)/(s^2)
2	1	2
3	2	6
4	3	12
Table 7		

D

Die Zahlenwerte sind nun schon angenehmer, aber die Anzeige der Maßeinheiten funktioniert nicht, weil auch der Vektor *Units* beim Speichern in Basiseinheiten umgerechnet wird.

Da hilft nur ein Spaltentitel mit passenden Einträgen. Hier wurde gleich noch eine weitere Zeile für die Formelzeichen definiert.

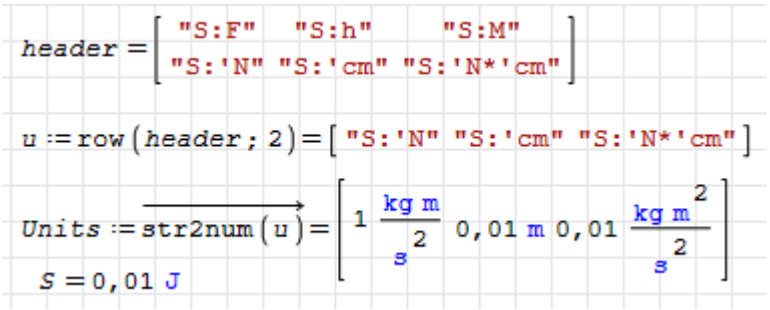
Units := [N cm N cm] *header* := $\begin{bmatrix} \text{"S:F"} & \text{"S:h"} & \text{"S:M"} \\ \text{"S:'N"} & \text{"S:'cm"} & \text{"S:'N*'cm"} \end{bmatrix}$

header ☑ show header
(none) ☐ show left stub
(none) ☐ show right stub
(none) ☐ show footer
Units ☐ show units summary

F	h	M
N	cm	N cm
2	1	2
3	2	6
4	3	12
Table 8		

D

Hinweis: Die Einheiten im Spaltentitel und im Einheitenvektor (*unit summary*) werden unabhängig voneinander definiert. Der Anwender ist dafür verantwortlich, dass die angezeigten und die tatsächlich verwendeten Einheiten zusammenpassen. Man kann den Einheitenvektor aber auch aus den Einheiten im Spaltentitel gewinnen. Dabei macht man sich zunutze, dass `"S:..."` eine ausführbare Anweisung ist, nämlich die Definition der Variable S mit dem folgenden Ausdruck.



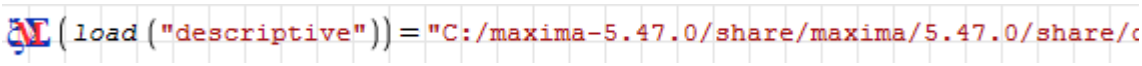
7 Datenanalyse

Inhalt

7.1 Deskriptive Statistik 53
7.2 Quantile 54
7.3 Boxplots 55
7.4 Histogramme 56

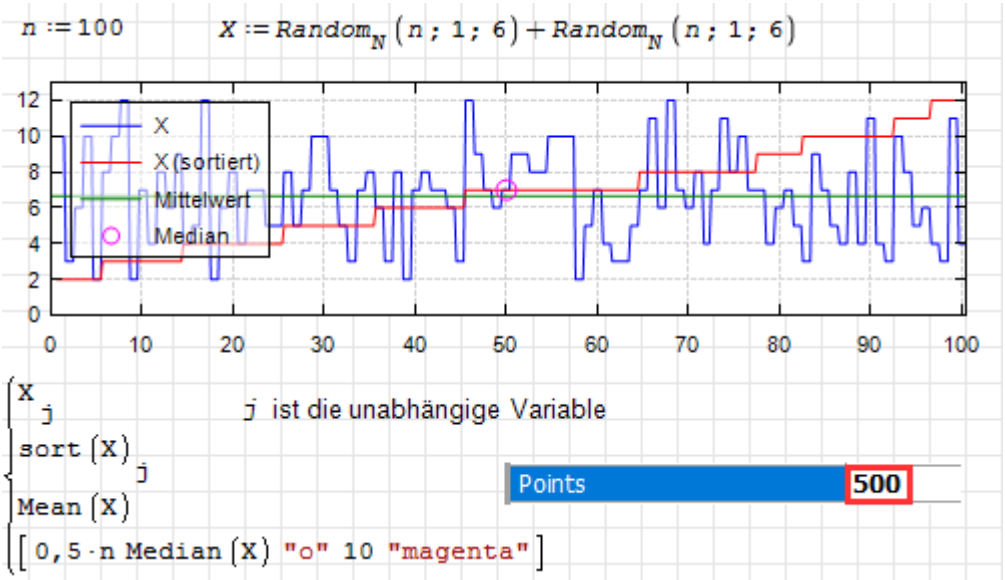
Hier werden einige Möglichkeiten zur Datenanalyse in SMath vorgestellt. SMath bringt von Haus aus nur ganz elementare Funktionen mit, so dass man am besten gleich die Plugins *Statistical Tools* und *MaximaPlugin* verwendet.

Das Maxima-Paket *descriptive* wird wie in Maxima selbst mit der Funktion`load()` geladen. Das bunte Symbol aus M, Σ und ξ steht für die Funktion `Maxima()`, welche einen Ausdruck an Maxima zur Ausführung übergibt. Das Ergebnis ist der Pfadname zur Paketdatei.



Im Diagramm unten sehen wir z.B. die Funktionen `Mean()` und `Median()` aus dem Plugin *Statistical Tools*. Welche Funktionen noch definiert sind, schaut man sich am besten unter **Einfügen> Funktion** an.

Als Beispiel dient eine Datenreihe X mit n Werten, die aus der Summe zweier ganzer Zufallszahlen zwischen 1 und 6 generiert wird. Das entspricht der Augensumme zweier Würfel (wie man es beim Monopoly oder Siedler von Catan kennt).



Im Diagramm wurde die Zahl der Auswertepunkte für Funktionen (Points) auf 500 gestellt, damit man den Stufenverlauf der Funktion $f(j) = X_j$ sieht.

7.1 Deskriptive Statistik

Die am Kapitelanfang definierten Daten werden jetzt analysiert.

Der Mittelwert ist die Summe der Datenwerte geteilt durch deren Anzahl. Der Median ist der Wert, den 50% der Datenpunkte nicht überschreiten. Er wird berechnet, indem die Daten sortiert und beim Mittelwert von 1 und n interpoliert werden.

Mittelwert $\texttt{Mean}(X) = 6,65$ $\quad \mu := \frac{1}{n} \cdot \sum_{j=1}^{n} X_j = 6,65$

Median $\texttt{Median}(X) = 7$ $\quad \texttt{linterp}\left([1..n]; \texttt{sort}(X); \frac{n+1}{2}\right) = 7$

Bei Varianz, Standardabweichung und Schiefe muss man zwischen den Werten für die Grundgesamtheit (Mittelwert ist fest) und Stichprobe (Mittelwert ist abhängig von der Stichprobe) unterscheiden. Wir zeigen im Folgenden die Berechnung mit Hilfe der Definitionsgleichung, die Berechnung mit dem Paket *descriptive* im *Maxima Plugin* und mit dem Plugin *Statistical Tools.*

$\texttt{load}("descriptive") = \texttt{"C:/maxima-5.47.0/share/maxima/5.47.0/share/c}$

Varianz (Grundgesamtheit)

$$\frac{1}{n} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^2 = 6,9275$$

$\texttt{var}(X) = 6,9275$

Varianz (Stichprobe)

$$\frac{1}{n-1} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^2 = 6,9975$$

$\texttt{var1}(X) = 6,9975$

$\texttt{Variance}(X) = 6,9975$

Standardabweichung (Grundgesamtheit)

$$\sigma := \sqrt{\frac{1}{n} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^2} = 2,632$$

$\texttt{std}(X) = 2,632$

Standardabweichung (Stichprobe)

$$\sigma_1 := \sqrt{\frac{1}{n-1} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^2} = 2,6453$$

$\texttt{std1}(X) = 2,6453$

$\texttt{StdDev}(X) = 2,6453$

Schiefe (Grundgesamtheit)

$$\frac{1}{\sigma^3} \cdot \frac{1}{n} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^3 = 0,1221$$

$\texttt{skewness}(X) = 0,1221$

$\texttt{Skewness}(X) = 0,1221$

Schiefe (Stichprobe)

$$\frac{1}{\sigma_1^3} \cdot \frac{1}{n-2} \cdot \sum_{j=1}^{n} \left(X_j - \mu\right)^3 = 0,1228$$

Man sieht, dass die Funktionen `Variance()` und `StdDev()` sich auf Stichproben beziehen, während `Skewness()` sich auf die Grundgesamtheit bezieht.

7.2 Quantile

Eine Verallgemeinerung von Minimum, Maximum und Median ist die Quantil-Funktion (mitunter auch Perzentil-Funktion genannt.). Sie gibt den Wert an, den ein vorgegebener Anteil zwischen 0 und 1 (0 und 100%) nicht überschreitet.

Solch eine Funktion steht nur im Maxima-Plugin zur Verfügung. Man kann sie aber in SMath nachbauen:

```
Q(X#; p#) := | n := length(X#)
             | i := (n − 1)·p# + 1
             | linterp([1..n]; sort(X#); i)
```

Hier wurden Programmierfunktionen verwendet:

- Die vertikale Linie kennzeichnet einen Anweisungsblock. Er wird mit der `line()`-Funktion erzeugt. Einfügen und Löschen von Einträgen funktioniert genau wie bei Vektoren und Listen.
- Rückgabewert einer mit Anweisungsblock definierten Funktion ist der letzte Eintrag.
- X#, p# sind Übergabeparameter. Es ist üblich, diese so zu schreiben, dass es keine Überlappungen mit gleichnamigen globalen Variablen gibt.
- n und i sind lokale Variablen.

Die Quantile für 0%, 25%, 50%, 75% und 100% entsprechen den Werten Minimum, unteres Quartil, Median, oberes Quartil und Maximum. Das folgende Bild zeigt die Übereinstimmung der obigen SMath-Implementierung mit der Maxima-Funktion `quantile()`.

```
load("descriptive") = "C:/maxima-5.47.0/share/maxima/5.47.0/share/c
```

Minimum	$Q(X; 0) = 2$	quantile(X; 0) = 2
	min(X) = 2	
Unteres Quartil	$Q_1 := Q(X; 0{,}25) = 4{,}75$	quantile(X; 0,25) = 4,75
Median	$Q(X; 0{,}5) = 7$	quantile(X; 0,5) = 7
	Median(X) = 7	
Oberes Quartil	$Q_3 := Q(X; 0{,}75) = 8$	quantile(X; 0,75) = 8
Maximum	$Q(X; 1) = 12$	quantile(X; 1) = 12
	max(X) = 12	
Interquartil-abstand	$Q_3 - Q_1 = 3{,}25$	qrange(X) = 3,25

Werte, die vom oberen oder unteren Quartil einen Abstand größer als ein bestimmtes Vielfaches des Interquartilabstands haben, werden als Ausreißer bezeichnet. Typischerweise ist dieser Faktor für die Ausreißergrenze gleich 1,5.

7.3 Boxplots

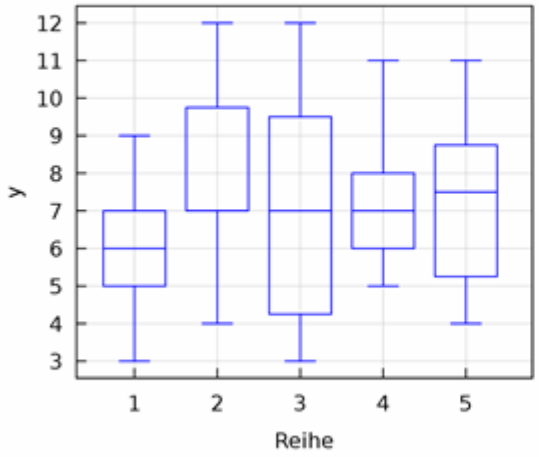
Es ist üblich, Minimum, unteres Quartil, Median, oberes Quartil und Maximum in Form von Boxplots darzustellen. Die Box wird von oberem und unterem Quartil begrenzt. Ausreißer extra eingetragen und bei den Minimum-Maximum-Marken nicht berücksichtigt.

Boxplots werden im Maxima-Paket *descriptive* definiert.

```
load("descriptive") = "C:/maxima-5.47.0/share/maxima/5.47.0/share/c
```

Die Funktion `boxplot_description()` erzeugt den Boxplot als Grafikobjekt für die `Draw2D()`-Funktion des *Maxima* Plugins. Hier ein minimales Beispiel mit 5 Versuchsreihen zu je 10 Würfen mit zwei Würfeln. Bei so wenigen Würfen sind die Verteilungskennzahlen der verschiedenen Reihen noch sehr unterschiedlich, was man im Boxplot sehr anschaulich sieht.

```
Reihen := 5   Werte := 10
M := Random_N(Werte; Reihen; 1; 6) + Random_N(Werte; Reihen; 1; 6)
```



1	2	3	4	5
3	10	12	7	6
7	9	4	8	5
6	12	8	6	4
6	7	10	5	8
5	7	5	10	9
7	7	10	5	11
5	11	8	6	5
5	4	6	7	7
7	7	3	8	10
9	6	3	11	8

M

```
Draw2D({boxplot_description(M; range = 1,5); "pdf"; {300
       {xlabel = "Reihe"                     {240
```

`boxplot_description(`*M*`, option1,...)` Optionen:

`box_width` (Voreinstellung 3/4): Relative Breite der Boxen. Der Wert muss im Bereich [0,1] sein.

`box_orientation` (Voreinstellung `vertical`): Mögliche Werte: `vertical` und `horizontal`.

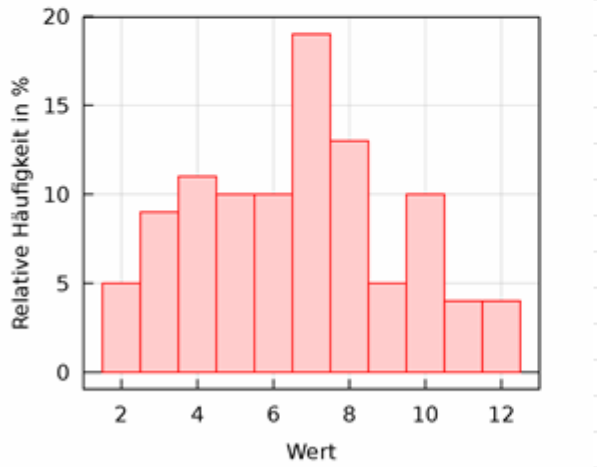
`range` (Voreinstellung ∞): Positiver Faktor bezogen auf den Interquartilabstand für die Ausreißergrenze.

`outliers_size` (Voreinstellung 1): Kreisgröße für Ausreißer.

7.4 Histogramme

```
H_args := { X
          { nclasses = { 1,5
          {            { 12,5
          {            { 11
          { frequency = percent
          { fill_density = 0,2

cmd := { apply(histogram_description; H_args)
       { xlabel = "Wert"
       { ylabel = "Relative Häufigkeit in %"
```

Argumente für *histogram_description*




```
Draw2D(cmd; "png"; { 300
                   { 240 )
```

`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` (Voreinstellung `10`) Klassenzahl des Histogramms oder eine Liste mit den Bereichsgrenzen und optional deren Anzahl:

$$\left\{ \begin{array}{c} x_{\max} \\ x_{\max} \\ n_k \end{array} \right. \quad \text{oder} \quad \left\{ \begin{array}{c} x_{\max} \\ x_{\max} \end{array} \right.$$

`frequency` (Voreinstellung `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` (Voreinstellung `auto`) , `endpoints`, `intervals` oder eine Liste mit Marken

8 Lineare Regression

Inhalt

8.1 Beispiel 58

Bei der linearen Regression besteht die Aufgabe besteht darin, eine Modellfunktion

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_n f_n(x)$$

so zu bestimmen, dass ein Datensatz aus m zugeordneten x- und y-Werten möglichst gut dargestellt wird. Die Regression heißt linear, weil die Modellfunktion linear in den freien Parametern c_i ist.

Eine Möglichkeit dafür ist die Methode der kleinsten Fehlerquadrate, die den mittleren quadratischen (vertikalen) Abstand zwischen den Funktionswerten $f(x_i)$ und den Datenwerten y_i minimiert. Die Funktion kann als Skalarprodukt des Koeffizientenvektors $\underline{c}$ mit 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 durch Anwendung der Funktionen f_i auf den ganzen Datenvektor $\underline{x}$ gebildet werden.

$$\underline{\underline{X}} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \cdots & f_n(x_m) \end{bmatrix}$$

Die Koeffizienten des Modells berechnen sich dann mit der Pseudoinversen von $\underline{\underline{X}}$

$$\underline{c} = \left(\underline{\underline{X}}^\top \underline{\underline{X}}\right)^{-1} \underline{\underline{X}}^\top \underline{y}$$

8.1 Beispiel

Diese Berechnung soll nun in SMath ausgeführt werden. Zunächst die Definition der Daten:

$$x_D := \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix} \quad y_D := \begin{bmatrix} 0,9 \\ 2,4 \\ 7,0 \\ 17,4 \\ 30,5 \\ 48,3 \end{bmatrix} \quad n_D := \text{rows}\left(x_D\right) \quad j := \left[1..n_D\right]$$

Die Modellfunktion (Ansatz) sei $f(x) = c_1 + c_2 x + c_3 x^3$[1]. Damit ist $f_1(x) = 1$, $f_2(x) = x$ und $f_3(x) = x^3$. Dann geben wir den Vektor $\underline{f}$ vor und berechnen die Matrix $\underline{\underline{X}}$:

$$f(x) := \begin{bmatrix} 1 \\ x \\ x^3 \end{bmatrix} \quad k := \left[1..\text{rows}\left(f\left(x_D\right)\right)\right]$$

$$X_{j,k} := f\left(x_{D_j}\right)_k \quad X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 8 \\ 1 & 3 & 27 \\ 1 & 4 & 64 \\ 1 & 5 & 125 \\ 1 & 6 & 216 \end{bmatrix}$$

Nun berechnen wir die Koeffizienten und bilden daraus die Modellfunktion $f(x)$ als Skalarprodukt aus dem Koeffizientenvektor und dem Funktionsvektor. Alternativ kann man auch die entsprechende Funktion aus dem Plugin *DotNumerics* verwenden[2]:

$$c := \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot y_D = \begin{bmatrix} -1,36 \\ 1,40 \\ 0,193 \end{bmatrix} \quad f(x) := c \cdot f(x)$$

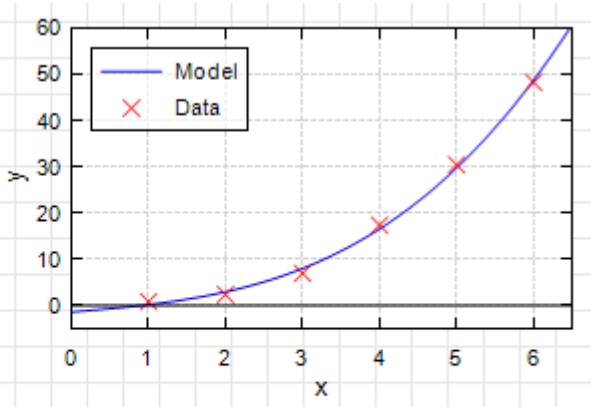
$$\text{dn\_LinAlgLLS\_SVD}\left(X; y_D\right) = \begin{bmatrix} -1,36 \\ 1,4 \\ 0,193 \end{bmatrix}$$

Nun soll noch die so bestimmte Funktion mit den Datenpunkten verglichen werden.

[1] Mit $f(x) = c_1 + c_2 x$ würde das der klassischen linearen Regression entsprechen, bei der eine Gerade an die Daten angepasst wird.

[2] SVD heißt *singular value decomposition* (Singulärwertzerlegung)

$P := \text{augment}\left(x_D, y_D\right)$



$$\begin{cases} f(x) \\ P \\ [0\ 0\ "+"\ 500] \end{cases}$$

Die Qualität der Modellanpassung kann man beurteilen anhand der Differenzen zwischen Modell und Daten an den einzelnen Punkten (E im folgenden Rechenblatt). Ein gängiges Maß zur Abweichungsbewertung ist auch der mittlere quadratische Fehler (mean square error, MSE), der hier einmal elementar mit Matrixfunktionen und einmal mit der Funktion `MSE()` aus dem Maxima-Plugin berechnet wird. Diese Funktion wird im Zusammenhang mit der nichtlinearen Regression noch genauer erläutert.

Abweichung zwischen Modell und Daten:

$$\text{MSE}\left(P; \begin{cases} x \\ y \end{cases}; y = f(x)\right) = 0,5436 \qquad E_j := y_{D_j} - f\left(x_{D_j}\right) = \begin{bmatrix} 0,661 \\ -0,593 \\ -1,07 \\ 0,785 \\ 0,698 \\ -0,484 \end{bmatrix}$$

$$\frac{1}{n_D} \cdot \sum_{i=1}^{\text{rows}\left(x_D\right)} \left(y_{D_i} - f\left(x_{D_i}\right)\right)^2 = 0,544$$

9 Nichtlineare Regression

Inhalt

9.1 Beispiel 1: Kubische Parabel durch 5 Zufallspunkte 61
9.2 Beispiel 2: Kreis durch 4 Punkte 62
9.3 Beispiel 3: Anpassung eines Modells an eingelesene Daten . 63

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; tol)`

`data` (Matrix) gegebene Daten. Jede Zeile ist ein Datensatz, jede Spalte eine Variable.

`vars` (Liste) Variablennamen (für jede Spalte in `data` ein Eintrag).

`eqn` Gleichung, verknüpft die in `vars` aufgelisteten Größen und die in `pars` aufgelisteten Ausgleichsparameter.

`pars` (Liste) Namen der freien Parameter in der Gleichung `eqn`.

`init` (Liste) der Startwerte für die Suche nach den optimalen Parametern.

`tol` (optional, Zahl) Toleranz für die Genauigkeit der Parameterwerte.

Rückgabewert ist eine Liste mit Gleichungen der Form `Parameter = Wert`. Diese kann mit `Assign()` für die Zuweisung der Werte an die Parameter benutzt werden, z.B. wenn man die Modellfunktion plotten will.

Zur Beurteilung der Qualität des Abgleichs gibt es die Funktionen `MSE()` und `Residuals()`, die den mittleren quadratischen Fehler bzw. die Abweichung an den Datenpunkten liefern.

Man beachte, dass im Allgemeinen das Ergebnis von den Anfangsschätzungen für die Parameter abhängen kann. Es handelt sich schließlich um ein numerisches Lösungsverfahren

Die Funktionen `Assign()`, `MSE()` und `Residuals()` sind wie `Fit()` im *Maxima*-Plugin definiert.

9.1 Beispiel 1: Kubische Parabel durch 5 Zufallspunkte

Eine kubische Parabel $f(x) = ax^3 + bx^2 + cx + d$ soll an fünf Punkte mit $x_i = 1 \ldots 5$ und Zufallswerten zwischen 0 und 1 angepasst werden.

Zunächst definieren wir die Datenmatrix durch Nebeneinanderstellen der x-Wertefolge und der y-Zufallszahlen. Dann wird die Modellfunktion $f(x)$ definiert, wobei sichergestellt wird, dass die Parameter freie Variablen sind. Wenn das auch so sichergestellt ist, kann man die `Clear()`-Anweisung weglassen.

$$D := \text{augment}([1..5]; \text{Random}(5))$$

$$\text{Clear}(a; b; c; d) = 1$$

$$f(x) := a \cdot x^3 + b \cdot x^2 + c \cdot x + d$$

$$D = \begin{bmatrix} 1 & 0{,}483 \\ 2 & 0{,}752 \\ 3 & 0{,}419 \\ 4 & 0{,}847 \\ 5 & 0{,}263 \end{bmatrix}$$

Nun wird die Fit-Funktion angewendet. Im Kontextmenü muss man Optimierung> Abgeschaltet einstellen, sonst kann SMath die Ergebnisgleichungen nicht anzeigen.

Daten und Spaltennamen — Modell — Parameter mit Anfangsschätzungen — gefundene Parameterwerte

$$F := \text{Fit}\left(D; \begin{pmatrix} x \\ y \end{pmatrix}; y = f(x); \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}; \begin{pmatrix} 1, \\ 3, \\ 1, \\ 1, \end{pmatrix}\right) = \begin{pmatrix} a = -0{,}0341621399154106 \\ b = 0{,}2400871958062992 \\ c = -0{,}4365135025377997 \\ d = 0{,}7587808396908096 \end{pmatrix}$$

Kontextmenü> Optimierung> Abgeschaltet

Mit `Assign()` werden die Ergebnisgleichungen von `Fit()` benutzt, um die gefundenen Werte den Parametern zuzuweisen. Somit kann die definierte Modellfunktion direkt im Diagramm verwendet werden.

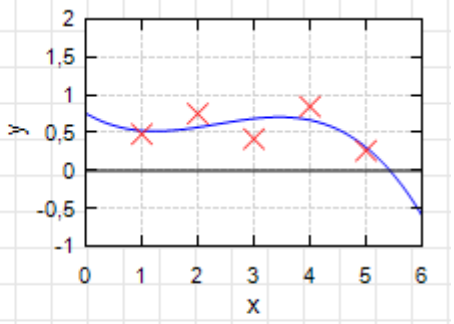
Die Anpassungsqualität wird dann noch anhand der Abweichungen (Residuen) und anhand des mittleren quadratischen Fehlers (MSE) bewertet. Diese Funktionen erhalten die gleichen Argumente wie `Fit()`.

$$\text{Assign}(F) = \begin{pmatrix} -0{,}0342 \\ 0{,}2401 \\ -0{,}4365 \\ 0{,}7588 \end{pmatrix}$$

Anpassungsqualität

$$\text{Residuals}\left(D; \begin{pmatrix} x \\ y \end{pmatrix}; y = f(x)\right) = \begin{bmatrix} -0{,}0449 \\ 0{,}1794 \\ -0{,}2688 \\ 0{,}1791 \\ -0{,}0448 \end{bmatrix}$$

$$\text{MSE}\left(D; \begin{pmatrix} x \\ y \end{pmatrix}; y = f(x)\right) = 0{,}0281$$



$$\begin{pmatrix} f(x) \\ \text{augment}(D; \text{"x"}; 10; \text{"red"}) \\ [0\ 0\ \text{"+"}\ 1000] \end{pmatrix}$$

9.2 Beispiel 2: Kreis durch 4 Punkte

Mittelpunktkoordinaten x_x, y_c und Radius R eines Kreises sollen so gewählt werden, dass er möglichst genau durch vier gegebene Punkte geht.

Als Modellfunktion verwenden wir die implizite Kreisgleichung $f(x, y) = 0$. Vorsichtshalber werden die verwendeten Variablennamen gelöscht.

$$D := \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 1 \\ 2 & 3 \end{bmatrix} \quad \text{Clear}(x_c; y_c; R; x; y) = 1$$

$$f(x; y) := (x - x_c)^2 + (y - y_c)^2 - R^2$$

Die Punkte liegen leider nicht auf einem Kreis, daher ist nur eine näherungsweise Lösung möglich. Mit der Funktion `Fit()` bestimmen wir die Kreisparameter so, dass der mittlere quadratische Fehler des Wert von $f(x, y)$ an den Datenpunkten minimal wird.

$$F := \text{Fit}\left(D; \begin{pmatrix} x \\ y \end{pmatrix}; f(x; y) = 0; \begin{pmatrix} x_c \\ y_c \\ R \end{pmatrix}; \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\right) = \begin{cases} x_c = 0{,}29924286819971646 \\ y_c = 1{,}9337810843605383 \\ R = 1{,}990876602322017 \end{cases}$$

Kontextmenü> Optimierung> Abgeschaltet

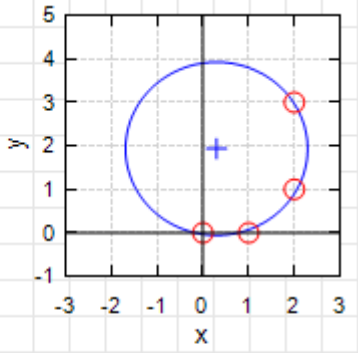
Die Ergebnisgleichungen werden wieder in Zuweisungen umgewandelt und für das Plotten benutzt. Der X-Y-Plot kann die implizite Kreisgleichung direkt darstellen. Mit Textmarken ergänzen wir die Datenpunkte, den Kreismittelpunkt und das Achsenkreuz.

$$\text{Assign}(F) = \begin{cases} 0{,}299 \\ 1{,}93 \\ 1{,}99 \end{cases}$$

$x_c = 0{,}299$

$y_c = 1{,}93$

$R = 1{,}99$




```
f(x; y)
augment(D; "o"; 10; "red")
[x_c y_c "+" 10 "blue"]
[0 0 "+" 1000]
```

Bei einer exakten Lösung müsste die Kreisfunktion an den Datenpunkten gleich Null sein. `Residuals()` zeigt die Abweichung. Das ist nicht der Abstand zwischen Kreislinie und Datenpunkt, ist diesem aber ungefähr proportional.

$$\text{Residuals}\left(D; \begin{pmatrix} x \\ y \end{pmatrix}; f(x; y) = 0\right) = \begin{bmatrix} -0{,}1345 \\ 0{,}267 \\ -0{,}1991 \\ 0{,}0658 \end{bmatrix} \quad f(D_{1\,1}; D_{1\,2}) = -0{,}1345$$

9.3 Beispiel 3: Anpassung eines Modells an eingelesene Daten

Aus der Werkstofftechnik, Buchmayr (2002): Abhängig von der Abkühlgeschwindigkeit t_{85} von 800°C auf 500°C kommt es in der Wärmeeinflusszone einer Schweißnaht zu einer Aufhärtung. In der Datei *HV_ WEZ.prn* stehen Messwerte für t_{85} (Spalte 1, in s) und für die Vickers-Härte HV (Spalte 2).

```
1       440
2       410
3       370
```

...

An diese Daten soll das folgende Modell angepasst werden:

$$HV(t_{85}) = a_1 + a_2 \arctan\left(a_3 \ln t_{85} + a_4\right).$$

Zunächst werden die Daten eingelesen und die Modellfunktion definiert (wieder mit zur Sicherheit gelöschten Variablenbelegungen):

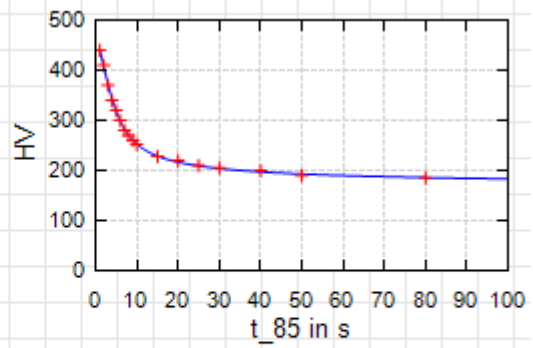
```
dir := CurrentDirectory(DocumentDirectory(""))

D := importData("HV_WEZ.prn")                    D = [1 440; 2 410; 3 370; ⋮]   rows(D) = 17

Clear(a1; a2; a3; a4; t; h) = 1

HV(t) := a1 + a2 · arctg(a3 · ln(t) + a4)
```

Nun wird der Abgleich durchgeführt. Auch hier ist wieder zu beachten, dass das Ergebnis von den Anfangsschätzungen der Parameter abhängen kann.

$$F := \text{Fit}\left(D; \begin{pmatrix} t \\ h \end{pmatrix}; h = HV(t); \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}; \begin{pmatrix} 300 \\ 0 \\ 4 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} a_1 = 328{,}9859053123413 \\ a_2 = -116{,}90652821023383 \\ a_3 = 0{,}9559051697601506 \\ a_4 = -1{,}438624548160154 \end{pmatrix}$$

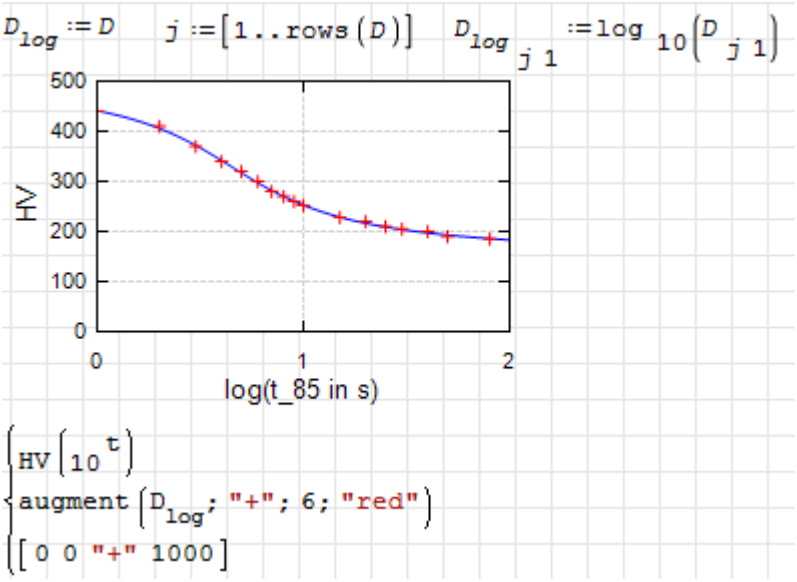
Die Qualität des Abgleichs wird mit einem Diagramm überprüft:

$$\text{Assign}(F) = \begin{pmatrix} 329 \\ -117 \\ 0{,}956 \\ -1{,}44 \end{pmatrix}$$

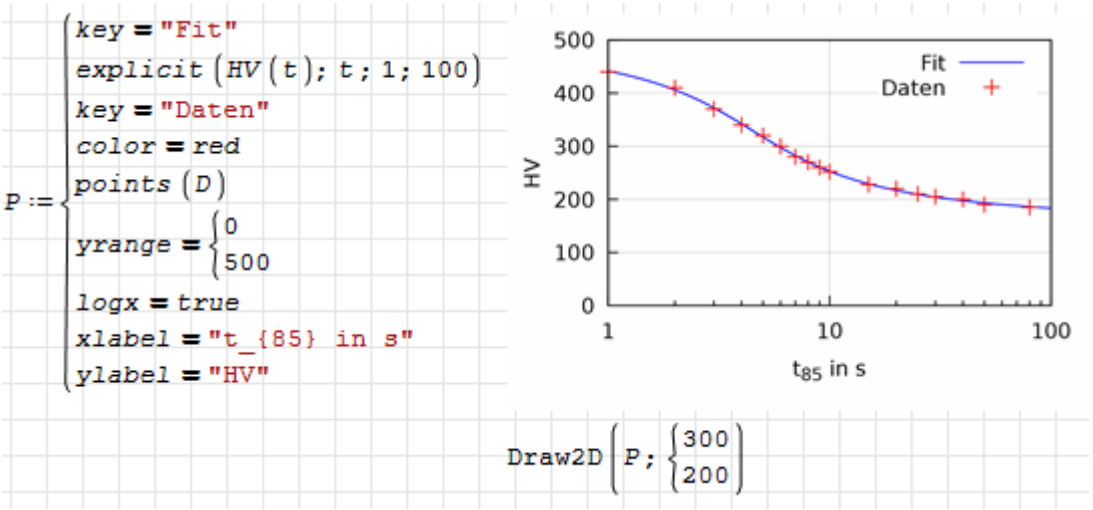



```
HV(t)
augment(D; "+"; 6; "red")
[0 0 "+" 1000]
```

Besser kann man den Verlauf bei einer logarithmischen x-Achse einschätzen. Das geht im X-Y Plot nur indirekt durch Transformation der Daten.



Alternativ kann man die Diagrammfunktionen des Maxima-Plugins benutzen (siehe Abschnitt 5.11).



Literatur

B. Buchmayr. *Werkstoff- und Produktionstechnik mit Mathcad. Modellierung und Simulation in Anwendungsbeispielen.* Springer, 2002.

M. Kraska. *SMath Studio mit Maxima. Einführung und Referenz für Version 0.99.7561.* Technische Hochschule Brandenburg. 2020. DOI: 10.25933/opus4-2946.

M. Kraska. *Technisches Rechnen mit SMath. Erste Schritte.* Technische Hochschule Brandenburg. 2023. DOI: 10.25933/opus4-2947.