[1 Taktische Fragen 4](#_Toc155798870)

[1.1 Matlab/Simulink versus Python 4](#_Toc155798871)

[1.2 Vektorisierung versus klassische Programmierung 4](#_Toc155798872)

[1.3 Editor versus Live Script Editor 4](#_Toc155798873)

[1.4 Word versus Live Script 4](#_Toc155798874)

[1.5 Eigenprodukt versus Fremdprodukt 4](#_Toc155798875)

[1.6 Zertifizierung versus Note reicht aus 4](#_Toc155798876)

[2 Wie funktioniert Wissenschaft 5](#_Toc155798877)

[2.1 Modell 9](#_Toc155798878)

[2.2 Studierende „Was ist numerisch“ 11](#_Toc155798879)

[2.3 Analytische Methoden 11](#_Toc155798880)

[2.4 Numerische Methoden 13](#_Toc155798881)

[2.5 Einfache Beispielaufgabe 13](#_Toc155798882)

[2.5.1 Klassischer Dreisatz 13](#_Toc155798883)

[2.5.2 Ungünstige Art der Auswertung 16](#_Toc155798884)

[2.5.3 Reflektiertere Art der Rechnung 16](#_Toc155798885)

[2.6 Beispiel Sinus 18](#_Toc155798886)

[3 Numerische Differentiation 19](#_Toc155798887)

[3.1 Implementierung von ypl in Matlab Skript C-Style 21](#_Toc155798888)

[3.2 Implementierung von ypl vektorisiert 21](#_Toc155798889)

[3.3 Hausaufgaben zur Vertiefung 23](#_Toc155798890)

[3.4 Präsenzaufgaben 24](#_Toc155798891)

[3.5 Vorgefertigte Matlabbefehle diff und gradient 25](#_Toc155798892)

[3.6 Nicht äquidistante Messdaten 26](#_Toc155798893)

[4 Numerische Integration 30](#_Toc155798894)

[5 Eulerverfahren 34](#_Toc155798895)

[6 (Rück)interpolation 35](#_Toc155798896)

[6.1 Sinc-Interpolation 35](#_Toc155798897)

[6.2 Lineare Interpolation 41](#_Toc155798898)

[7 Simulink Spezialitäten 42](#_Toc155798899)

[7.1 Simulink Debugger 42](#_Toc155798900)

[7.1.1 Einstellungen für Euler 42](#_Toc155798901)

[7.1.2 Breakpoint setzen 43](#_Toc155798902)

[7.1.3 Anzuzeigende Signale auswählen (Show port values) 44](#_Toc155798903)

[7.1.4 Debugsequenz starten 44](#_Toc155798904)

[7.1.5 Debugsequenz mit einem Integrator 45](#_Toc155798905)

[7.2 Simulationsbeispiel: Totzeitglied mit P-Regler 47](#_Toc155798906)

[7.3 Simulationsbeispiel: Reale Spannungsquelle 49](#_Toc155798907)

[7.4 Algebraic Loop 50](#_Toc155798908)

[8 Matlab Spezialitäten 51](#_Toc155798909)

[8.1 Nutzung neuer Funktionen in alten Versionen 51](#_Toc155798910)

[9 Aufstellen von Whiteboxmodellen 54](#_Toc155798911)

[9.1 Mechanische Systeme 54](#_Toc155798912)

[9.1.1 Dynamische Grundgleichungen 54](#_Toc155798913)

[9.1.2 Federn 56](#_Toc155798914)

[9.1.3 Übung Dynamik 3 58](#_Toc155798915)

[9.1.4 Beispiel Druckfeder zwischen Massen 59](#_Toc155798916)

[9.1.5 Dämpfer 60](#_Toc155798917)

[9.1.6 Übung Dynamik 4 71](#_Toc155798918)

[9.1.7 Übung Dynamik 5 71](#_Toc155798919)

[9.2 Elektrische Systeme 72](#_Toc155798920)

[9.2.1 Knotenpunktsatz, 1. Kirchhoffsche Regel 72](#_Toc155798921)

[9.2.2 Maschensatz, 2. Kirchhoffsche Regel 73](#_Toc155798922)

[9.2.3 Bauteile 73](#_Toc155798923)

[9.2.4 Beispiel Filter zweiter Ordnung 79](#_Toc155798924)

[9.3 Hydraulische Systeme 80](#_Toc155798925)

[10 Brute Force 81](#_Toc155798926)

[11 Monte Carlo 82](#_Toc155798927)

[12 Gleichungssysteme 83](#_Toc155798928)

[13 Least Squares 84](#_Toc155798929)

[14 Weitere Quellen 85](#_Toc155798930)

[15 Liste der Matlabbefehle 86](#_Toc155798931)

[15.1 Vektor / Matrixbefehle 86](#_Toc155798932)

[15.2 Logical Indexing 87](#_Toc155798933)

[15.3 Integrieren/Differenzieren 87](#_Toc155798934)

[15.4 Ein-Ausgabe, Textverarbeitung 88](#_Toc155798935)

[15.5 Plotbefehle 88](#_Toc155798936)

[15.6 Sonstige 89](#_Toc155798937)

[15.7 Eigene Funktionen 89](#_Toc155798938)

[16 Programmierkonstrukte in verschiedenen Sprachen 90](#_Toc155798939)

[17 Nachbesprechung der Übungen 92](#_Toc155798940)

[17.1 PM1 Primzahlen 92](#_Toc155798941)

[18 Literatur 93](#_Toc155798942)

# Taktische Fragen

## Matlab/Simulink versus Python

* Suggestivfrage, weil es kein kostenloses Äquivalent zu Simulink in Python gibt  
  Aber viele nutzen Matlab ohne Simulink.
* Ähnliche Veranstaltung (Simulation mit Matlab) schreibt Tool explizit vor.
* Riesen Pluspunkt von Matlab war die Dokumentation und der Support  
   Zugriff über Chat-GPT und andere nun oft bequemer als Dokumentation (fertige Codeschnipsel statt Dokumentation einer bestimmten Funktion)  
  In letzter Zeit bieten die Python Foren oft Lösungen, wo der Matlab-Support keine bietet
* Matlab teuer, aber and der Hochschule umsonst
* Große Professionelle Nutzer und Tool-Anbieter sind noch auf der Matlab-Schiene unterwegs (Opel-Carmaker)
* Python kann in Matlab eingebunden werden (für Fans)

## Vektorisierung versus klassische Programmierung

* Vektorisierung ist der Mehrwert und der eigentliche Charakter von Matlab
* Vektorisierung ist schneller, weil Matlab Funktionen darauf hin optimiert sind
* Was ist klarer?
* Bei klassischer Programmierung schnellerer Wiedererkennungswert

## Editor versus Live Script Editor

## Word versus Live Script

## Eigenprodukt versus Fremdprodukt

* Dozenten erstellt versus erstellt von Studierenden gegen Rohpunkten für Klausur
* MOOC-Videos?
* Matlab onramp?
* Matlab essentials auf Edex?
* Hoch Videos

## Zertifizierung versus Note reicht aus

# Wie funktioniert Wissenschaft

Fakten

Daten

Messwerte

Ereignisse

(Hypo-)thesen

Theorien

Modelle

Gleichungen

Induktion

Deduktion

Deduktion

Deduktion

Induktion

Abbildung ‑: Deduktions-Induktionskreislauf nach (Box, et al., 2005)

**Deduktion**: [lat.; eigtl. = das Abführen, Fortführen]  
Ableitung einer Aussage (These A) aus anderen Aussagen (Hypothesen A1,…An) kraft log. Schlußregeln (symbolisiert: ). Sind die Hypothesen Sätze, also wahre Aussagen (z. B. Axiome), so heißt die D. ein deduktiver Beweis der These. In axiomat. Theorien ( axiomatisches System) bilden D.en das einzige Beweisverfahren. (MEYERS, 1974)

**Induktion**: [lat; eigtl. das Hineinführen, Einführen]  
Sammelbez. für Verfahren zur Gewinnung bzw. für den Beweis allgemeiner Aussagen auf nicht rein logischem Wege. – Die wohl geläufigste Anwendungsform der *induktiven Methode* liegt vor bei dem Schluß von einer Anzahl begründeter Einzelaussagen A(k) über Objekte k aus einem Bereich auf die generelle Aussage d.h. die Behauptung A gelte für jedes Objekt des fragl. Bereichs. – Allgemein spricht man überall dort von I. , wo aus der Gültigkeit einer Aussage oder Theorie für die Elemente einer (Teil-)Klasse auf die Gültigkeit derselben oder einer anderen generellen Aussage oder Theorie für eine andere Klasse (z.B. Oberklasse) geschlossen wird. Der Fall, daß die Gültigkeit der fragl. Aussage bereits für jedes Element der zugrunde gelegten Klasse gezeigt ist, für die erschlossen ist, gilt dabei ebenso wenig als echte Induktion in diesem Sinn wie die „formale“ Induktion (vollständige Induktion) über kalkülmäßig konstruierte Bereiche von Objekten. I.en liefern niemals schlüssige Beweise. Andererseits sind sämtl. Wissenschaften auf induktives Vorgehen angewiesen, wenn sie nicht auf die Formulierung allgemeiner Sätze bzw. Theorien überhaupt verzichten wollen. –Deshalb ist die Untersuchung und Rechtfertigung induktiver Methoden (induktive Logik) und die des Sinns induktiv gewonnener Aussagen eines der Hauptprobleme der modernen Philosophie und Wissenschaftstheorie. – auch unendliche Induktion. (MEYERS, 1974)

**Deduktion** [lat. „Herabführung“] die, die Ableitung von Aussagen mithilfe logischer Schlussregeln aus anderen, allgemeineren Aussagen; Ggs.: Induktion. (Brockh., 2007)

**Induktion**: [lat; „Hinführung“], die, der (nichtlogische) Schluss vom Besonderen auf das Allgemeine. Im Unterschied zur vollständigen I. in der Mathematik spricht man auch von unvollständiger I. Vom kritisch-rationalist. Standpunkt aus kommt den mit I. asl Verfahrensweise aller Erfahrungswiss.en begründeten Aussagen (auch Wahrscheinlichkeitsaussagen) ein bestimmter Grad an Bewährung, aber nie Gewissheit zu. Der Induktivismus versucht, das I.-Verfahren zu präzisieren und zu begründen sowie eine induktive Logik zu entwickeln. Ggs.: Deduktion. (Brockh., 2007)

Fragen:

Das mathematische Verfahren vollständige Induktion ist hier wie einzuordnen?

Bing-Chat:

Die **vollständige Induktion** ist ein mathematisches Beweisverfahren, bei dem eine Aussage für alle natürlichen Zahlen bewiesen wird, die größer oder gleich einem bestimmten Startwert sind. Da es sich um unendlich viele Zahlen handelt, kann eine Herleitung nicht für jede Zahl einzeln erbracht werden. [Die vollständige Induktion ist ein **deduktives Verfahren**, da sie von allgemeinen Aussagen zu spezifischen Schlussfolgerungen schließt](https://de.wikipedia.org/wiki/Vollst%C3%A4ndige_Induktion)

Abbildung 2‑2 Car example (Box, et al., 2005)

Observed

State of

nature

Theory

Induktion

Deduktion

Missing data

true

state of

nature

DOE

Searches

Chance

Expected state

of

nature

Deduktion

**Today**

**is Like**

**any day**

**Car is**

**stolen**

Car in my

Parking

place

Induktion

Deduktion

Missing data

Deduktion

is

not

Car not

in parking lot

Plan to

look

around

Car is

in parking lot

**A thief took**

**it and**

**brought**

**it back**

Car harmed

in parking lot

Plan to

inspect car

Car is unharmed

and unlocked

**My wife**

**took it**

She probably left a note

in parking lot

The note is here

in parking lot

Deduktion

Induktion

Do look around

Do inspect car

Deduktion

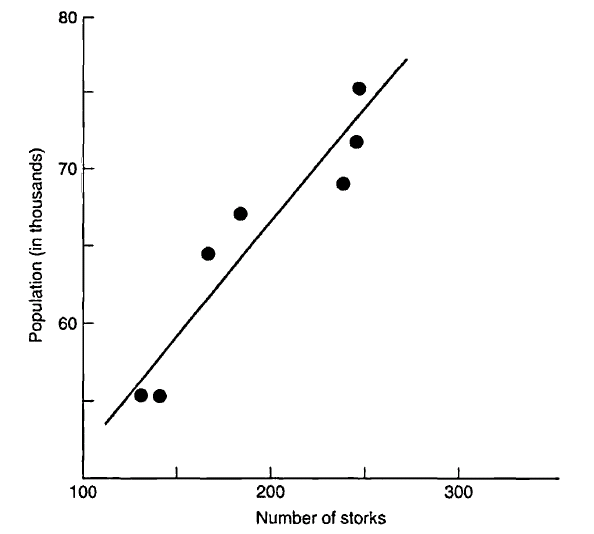


Abbildung 2‑3 Stork example (Box, et al., 2005)

Typische Fragen:

Folgt aus der Korrelation eine Kausalität?

Folgt aus Ordinate Abszisse oder aus Abszisse Ordinate? Geben Sie Interpretationen für das Storch-Bevölkerung Beispiel!

THE ROAD TO WISDOM

The road to wisdom? -- Well, it's plain

and simple to express:

Err

and err

and err again

but less

and less

and less.

(Piet Hein, Universalgelehrter)

## Modell

Ein Modell ist eine vereinfachende Beschreibung eines Teilaspekts der Wirklichkeit. Der eigentliche Trick bei der Vereinfachung ist, dass nicht nach dem Gießkannenprinzip vereinfacht wird. Die Vereinfachung ist bewusst selektiv. Aspekte, die dem Anwender des Modells besonders wichtig sind, werden möglichst genau beschrieben, andere nur ungefähr oder gar nicht.

Das Modell soll weniger komplex sein als die Wirklichkeit, um spezielle Fragestellungen einfacher beantworten zu können.

Jedes Modell stellt also einen Kompromiss zwischen Genauigkeit (Validität) und Aufwand (Komplexität) dar (Zielkonflikt).

Im Rahmen dieser Veranstaltung werden nur quantitative Modelle betrachtet (mindestens die Ausgangsgröße eines Modells soll einen Zahlenanteil beinhalten (nur Zahl [z.B. bei Wahrscheinlichkeiten] oder Zahlenwertanteil einer physikalischen Größe). Kurzer Abriss anderer Modelle

Manipulative Modelle wie „blöder Bayernfan“, „Gutmensch“, „Macho“, „Emanze“ usw. dienen nicht dem Zweck die Wirklichkeit möglichst genau zu beschreiben, sondern dem Zweck die Wahrnehmung der Wirklichkeit zu verzerren. Sie sind nicht Gegenstand dieser Veranstaltung.

Klassische Modelle:

Massenpunkt

Starrer Körper

Homo oeconomicus

Ohm’scher Widerstand

Feder Masse Dämpfer System

Motorkennlinie

Idealer Würfel

**Modellarten**

Sammlung von Gleichungen oder sonstigen Regeln (mathematisches/logisches Modell)

Wirkungsplan, FEM-Modell, Zustandsraummodell,…(Struktur + Parameter)

Physikalisches Modell (Analogrechner, Flugmodell gleicher Reynoldszahl)

**zugehörige Wege zu modelbasierten Vorhersagen**

Berechnen (analytische Methoden) oder

Simulieren (numerische Methoden) oder

Experimentieren (Beeinflussung von unabhängigen Variablen, Messung der abhängigen Variablen)

**Wege zu Daten über die Wirklichkeit**

Experiment (soweit wie möglich an realen Dingen, nicht an Modellen)

Sichtung, Ordnung von bestehenden Daten

## Studierende „Was ist numerisch“

Wenn ich den Taschenrechner brauche

Programmiersprache für Mathematik

Wenn man was mit Zahlen ausdrückt

Was ist da Gegenteil von numerisch?  
 -> Buchstaben

->literarisch

## Analytische Methoden

Eine analytische Methode liefert eine Formel, die den Zusammenhang zwischen abhängigen und unabhängigen Variablen exakt beschreibt, auch wenn die Werte dieser Variablen noch nicht bekannt sind. Dies stellt die größte Allgemeinheit einer Lösung dar.

Beispiel Bewegung mit konstanter Beschleunigung ohne Anfangswerte:

Damit ist das Problem ein für alle Mal gelöst.

Nun ja, eigentlich aber nur dann, wenn man mit der reinen Formel zufrieden ist. Möchte man jetzt s konkret für und berechnen, also einen Zahlenwert bestimmen, so ist das sogenannte Auswerten der Formel ein numerischer Vorgang. Früher hatten Wissenschaftler Mitarbeiter, so genannte Computer (Menschen!) die solche Aufgaben erledigt haben. Heute werden meist elektronische Rechner für den Umgang mit Zahlen benutzt.

Während die frühen Computer (Menschen) auch durchaus für analytische Berechnungen „benutzt“ wurden, wird die überwiegende Zahl der elektronischen Rechner für numerische Berechnungen benutzt. Eine Ausnahme bilden lediglich so genannte CAS (Computer Algebra Systeme). Diese rechnen auch analytisch und das nicht nur im Bereich der Algebra, sondern auch im Bereich der Analysis.

Kurz zur Wortherkunft: analytisch soll nicht bedeuten zur Analysis gehörend (das Teilgebiet der Mathematik, das sich mit Infinitesimalrechnung, also Grenzwertbildung, Integralen und Ableitungen befasst) sondern bezieht sich auf Analyse (ein Problem gedanklich durchdringen).

Die analytische Lösung ist mit viel Hirnschmalz verbunden, die Auswertung der Endformel ist meist vergleichsweise einfach.

Allerdings gibt es in der analytischen Welt auch ein paar Taschenspielertricks:

Diese Formel für die Hangabtriebskraft kann für die meisten Winkel nicht fehlerfrei ausgewertet werden, in dem Sinne, dass eine (auch gerne periodische) Dezimalzahl ausgegeben wird. Denn die meisten Sinuswerte führen zu irrationalen Zahlen, die ja bekanntlich gerade nicht als Bruch oder Dezimalzahl darstellbar sind. (Beispiel: ).

An viele spezielle Funktionen haben sich die Anwender gewöhnt und bedenken nicht mehr, dass diese nicht fehlerfrei ausgewertet werden können (…im Sinne…Dezimalzahl…).

Noch klarer ist die nicht realisierbare Auswertung, wenn innerhalb der Formel unendliche Reihen auftauchen. Denn bei der exakten Berechnung solcher Reihen wird kein Rechner (menschlich oder elektronisch) jemals fertig. (Eine Ausnahme bilden jene unendlichen Reihen, für die es eine analytische Lösungsformel ohne unendliche Reihe gibt.)

Nochmals zurück zu unserer Hangabtriebskraft, der Sinus nun ersetzt durch seine Taylorreihenentwicklung:

macht deutlich, wie kompliziert die Formel in Wirklichkeit ist. (In dieser Variante der Formel MUSS der Winkel im Bogenmaß eingesetzt werden.)

Mehr zur Matlab-Implementierung von sin(x) siehe:

[The Tetragamma Function and Numerical Craftsmanship - MATLAB & Simulink (mathworks.com)](https://de.mathworks.com/company/newsletters/articles/the-tetragamma-function-and-numerical-craftsmanship.html)

## Numerische Methoden

Numerische Methoden liefern Zahlenwerte für die abhängigen Variablen, die zu vorher festgelegten Zahlenwerten der unabhängigen Variablen gehören.

Numerische Methoden sind oft iterativ, das heißt neue Rechendurchgänge bauen auf alten Rechendurchgängen auf.

Numerische Methoden sind im Bereich der reellen Zahlen meist fehlerbehaftet.

Es gibt aber auch numerische Methoden im Bereich der ganzen Zahlen (z. B. GGT-Bestimmung), die fehlerfrei sind.

## Einfache Beispielaufgabe

Rita, eine Studentin in iIng braucht für eine Veranstaltung von 3 CP eine Zeit von 17 Tagen. Wie lange braucht Sie für ein Studium mit 180 CP?

### Klassischer Dreisatz

Hier werden sich die Geister scheiden:

Für analytisch spricht, dass der Endterm fehlerfrei ist.

Gegen analytisch spricht, dass erstens keine Formel vorhanden ist und zweitens auch keine Variablen benutzt werden.

Für analytisch spricht wieder, dass bei geänderten Zahlenwerten in der Aufgabestellung man einfach diese in den Endterm übertragen könnte, es muss nichts bei neuen Zahlen neu gelöst werden.

Analytische modelbasierte Methode:

t Zeit abhängige Variable (laut Aufgabenstellung gefragt)

w Arbeit unabhängige Variable (laut Aufgabenstellung gegeben)

p Leistung Parameter (in Aufgabenstellung versteckt, eine Eigenschaft von Rita)

Die Einteilung in abhängige Variable, unabhängige Variable und Parameter ist keine Eigenschaft der Gleichung. Bei einer Aufgabenstellung:

Welche Leistung p muss Rita erbringen, damit Sie in 3 Jahren fertig ist?

Wäre p die abhängige Variable, die Gleichung aber unverändert.

Darstellung der Gleichung mit Kennzeichnung Eingang/Ausgang: Wirkungsplan



Das prinzipielle Modell wird…



…parametriert und mit Erregung (Eingang) versehen…



…und dann wird der Ausgang berechnet. Man sagt das Modell wird simuliert.

Noch mal zurück zur Rechnung ohne Simulink.

### Ungünstige Art der Auswertung





Dies ist eindeutig eine numerische Rechnung. Das schriftliche Dividieren und Multiplizieren wird kochrezeptmäßig durchgeführt. Die dahinterstehenden Rechenregeln werden nicht mehr reflektiert. Aus Zeit- und Lustgründen wird die Anzahl der Stellen begrenzt.

Frage: Würde die 17 zu 18 geändert, ergeben sich keine Nachkommastellen und somit kein Fehler. Ist das Verfahren dann kein numerisches Verfahren mehr?

Antwort: Doch, ein Fehler kann auch mal zufällig Null sein.

### Reflektiertere Art der Rechnung





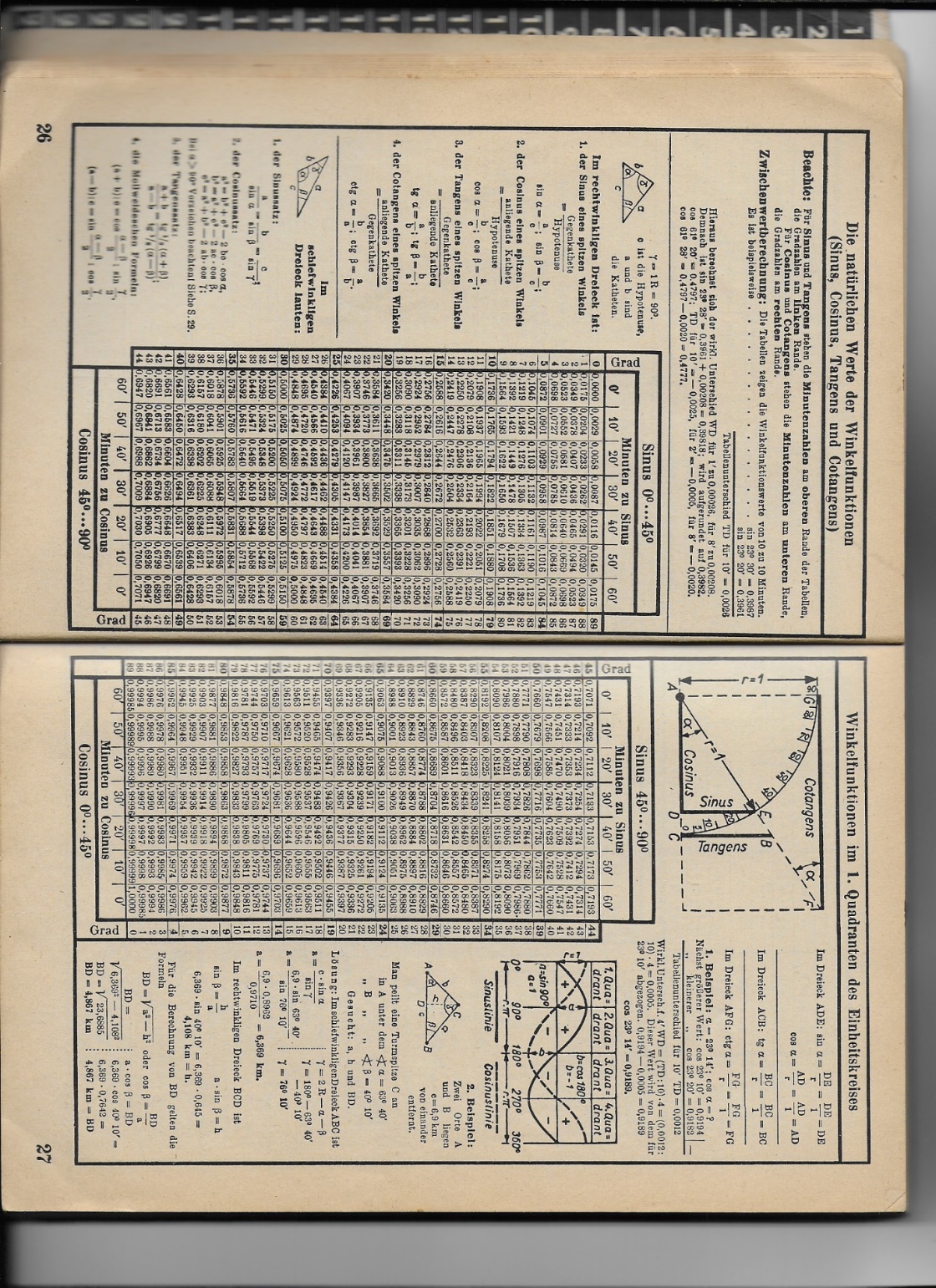
Per „Mustererkennung“ wird richtiges Ergebnis 1020 berechnet (oder erraten). Hier sind wir scharf an der Grenze zwischen numerischem Verfahren und analytischen Verfahren. Da aber die Hauptarbeit mit neuen Zahlen neu anfällt, ist diese Art der Auswertung immer noch numerisch.

Geschicktes Kürzen:

Zwar fehlerfrei, bei neuen Zahlen geht das Spiel aber wieder von vorne los…

## Beispiel Sinus

Die typische Einführung des Sinus in der Schule ist numerischer Art. Für einen Winkel wird der sinus des Winkels zur Verfügung gestellt. Die Beschulten (Genderungetüm) können mit den zur Verfügung gestellten Winkeln rechnen, sie können aber umgekehrt ohne Tabellenbuch oder Taschenrechner nicht den Sinus zu einem beliebigen Winkel ausrechnen.



Exakte Werte des Sinus:

|  |  |
| --- | --- |
| Winkel | Sin(Winkel) |
| 0°=0rad | 0 |
| 30°=30\*pi/180=pi/6 | 1/2 (Aus Aufdopplung zu gleichs. Dreieck) |
| 45°=pi/4 | (Aus gleichseitigem Dreieck u. Pythagoras |
| 60°=pi/3 | (Kongruentes Dreieck zu 30°) |

Selbst die exakten Werte des Sinus sind nicht alle rational! ->Jeder Taschenrechner, jedes Tabellenbuch ist falsch.

# Numerische Differentiation

Tangente

Sekante rechts

Sekante links

z. Sekante

X

Y=f (X)

Xr

Xm

Xlm

Der Differentialquotient ist definiert als die Steigung einer Tangente an die Kurve.

Den x-Wert, an dem die Steigung bestimmt werden soll, nennen wir Xm. Rechts davon folge Xr und links davon soll Xl sein. Nicht dargestellt sind die Werte Yl, Ym und Yr, welche die Funktionswerte an den entsprechenden X-Werten bezeichnen.

Die Steigung der linken Sekante ist:



Die Steigung der rechten Sekante lautet:



Die Steigung der zentrierten Sekante lautet:

Wenn die Abtastpunkte immer die gleiche Schrittweite in x haben (äquidistante Abtastung), entspricht die Steigung der mittleren Sekante auch dem Mittelwert aus linker und rechter Sekante:



Für den Punkt ganz links gibt es keinen linken Nachbarn.

Für den Punkt ganz rechts gibt es keinen rechten Nachbarn.

Für die mittleren x-Werte kann dann die zentrierte Sekante gerechnet werden:

Der letzte Wert muss dann mit der linksseitigen Sekante bestimmt werden.

Der erste Wert muss als rechtsseitige Sekante bestimmt werden

Bei der analytischen Rechnung ergibt sich:



## Implementierung von ypl in Matlab Skript C-Style

%% Quasikontinuierliche Kurven erzeugen

x=ones(1,501) ;% "Deklarieren eines Felds mit 501 Einzelwerten

y=ones(1,501);

yp\_ana=ones(1,501);

dx=0.01;

for i=1:501

x(i)=(i-1)\*dx;

y(i)=sin(x(i));

yp\_ana(i)=cos(x(i));

end

%% Abtastwerte erzeugen

av=10; %Abtastverhältnis, jeder wievielte Wert der Ausgangskurve wird genommen?

Na=floor(501/av)+1;

xa=ones(1,Na);

ya=ones(1,Na);

j=1;

for i=1:av:501

xa(j)=x(i);

ya(j)=y(i);

j=j+1;

end

%% linksseitige Ableitung ausrechnen

ypl=ones(1,Na); %Ersetzt Deklaration, nicht unbedingt notwendig

ypl(1)=nan; %Not A Number, siehe NAN in math.h bei C

for i=2:1:Na

xm=xa(i);

ym=ya(i);

xl=xa(i-1);

yl=ya(i-1);

yp=(ym-yl)/(xm-xl);

ypl(i)=yp;

end

%% Ableitungen darstellen

% hierzu gibt es im nackten C keine Entsprechung

hold off

plot(x,y,'r','DisplayName','sin')

hold on

plot(xa,ya,'Marker','o','LineStyle','none','Color','r','DisplayName','sin\_a');

plot(xa,ypl,'ob','DisplayName','ypl') %plotten der linksseitigen Ableitung

plot(x,yp\_ana,'g','DisplayName','yp\_a\_n\_a')%plotten der analytischen Ableitung

legend

## Implementierung von ypl vektorisiert

%% Quasikontinuierliche Kurven erzeugen

clear all

clc

dx=0.01;

x=1:501;

x=x\*dx;

y=sin(x);

yp\_ana=cos(x);

%% Abtastwerte erzeugen

av=40; %Abtastverhältnis, jeder wievielte Wert der Ausgangskurve wird genommen?

xa=x(1:av:end)

ya=y(1:av:end)

%% linksseitige Ableitung ausrechnen

yl=[nan,ya(1:end-1)]

dxa=xa(2)-xa(1);

ypl=(ya-yl)/dxa

%% Ableitungen darstellen

% hierzu gibt es im nackten C keine Entsprechung

hold off

plot(x,y,'r','DisplayName','sin')

hold on

plot(xa,ya,'Marker','o','LineStyle','none','Color','r','DisplayName','sin\_a');

plot(xa,ypl,'ob','DisplayName','ypl') %plotten der linksseitigen Ableitung

plot(x,yp\_ana,'g','DisplayName','yp\_a\_n\_a')%plotten der analytischen Ableitung

legend

Unterschiede C<-> Matlab

* Deklaration in Matlab nicht notwendig, aber sinnvoll ->ones-Funktion
* Semikolon nicht vorgeschrieben, unterdrückt Ausgabe, erlaubt mehrere Befehle in einer Zeile
* Andere Syntax für for vergl. Abbildung 16‑1
* Andere Syntax für Felder vergl. Abbildung 16‑1
* Nicht im Beispiel oben: andere Syntax für if und function
* Code folding aus visual Studio bekannt?

## Hausaufgaben zur Vertiefung

* Erzeugen Sie einen Zeilenvektor x von 0 bis 20 in Schritten von zwei!
* Erzeugen Sie einen Zeilenvektor x von 0 bis 17 in Schritten von zwei! Lesen Sie den letzten Wert des Vektors mit Hilfe von „end“ aus.
* Erzeugen Sie den Vektor x1, der aus den ersten vier Elementen von x besteht!
* Erzeugen Sie den Vektor x2, der aus den letzten vier Elementen von x besteht!
* Erzeugen Sie den Vektor x3, der aus den Elementen 1bis 3 und 6 bis Ende von x besteht!
* Ändern Sie Liniendicken, Markerarten, Markergrößen, Linienfarben und Markerfarben im Plot ab! Ändern Sie anschließend den m-Code entsprechend ab!

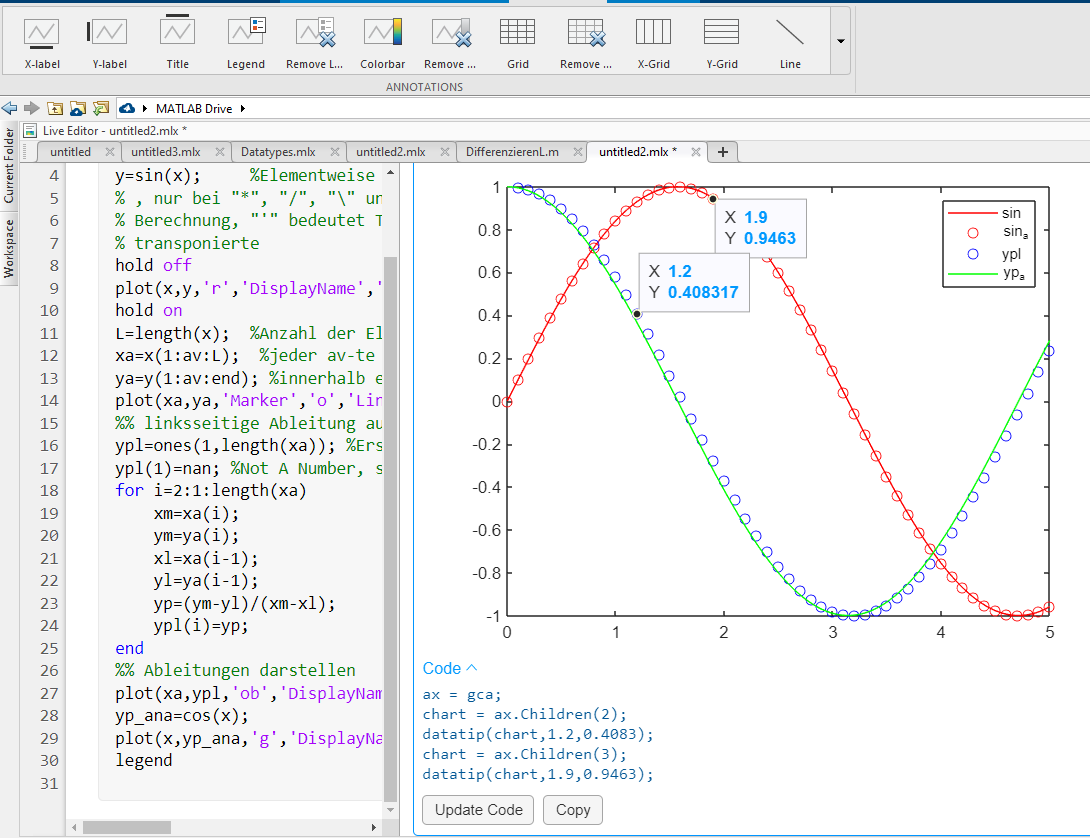
Hinweis:

Tools/Edit Plot aktivieren.

Doppelklick auf Linie oder Marker öffnet den Property Inspector. Die Menünamen dort entsprechen den Modifier Namen im Matlab-plot-commando, die Einträge den Modyfier-Werten.

Unter File/generate Code wird Matlab-Code erzeugt, dem mit etwas Erfahrung auch die einzelnen Befehle entnommen werden können.

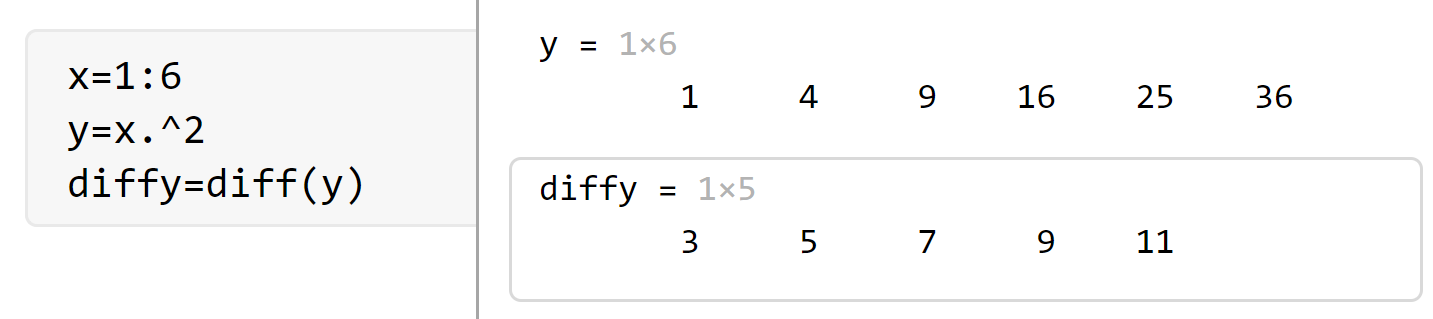
Eine noch intuitivere Nutzerführung steht zur Verfügung, wenn man das Skript in ein Matlab Live Skript umwandelt.



## Präsenzaufgaben

* Wandeln Sie das m-Script in ein Live Script um und ergänzen Sie einen Slidebar für av!
* Was fällt Ihnen als Unterschied zwischen ypl und yp\_ana auf? Wie könnte man diesen Unterschied beheben?
* Ergänzen Sie (im C-Style) die Berechnung ypr und ypc und stellen Sie diese dar!
* Wandeln Sie die Berechnungen in einen Vector-style um!
* Stellen Sie einen Sinus als Stem-Diagramm dar, markieren Sie mittels logical indexing alle Werte, die größer einer Schwelle S sind.  
  

## Vorgefertigte Matlabbefehle diff und gradient



**+**

**-**

**+**

**-**

**+**

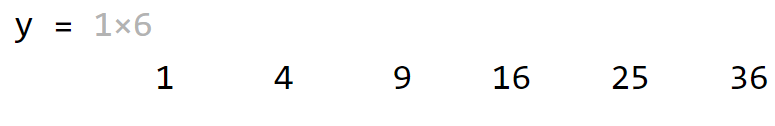
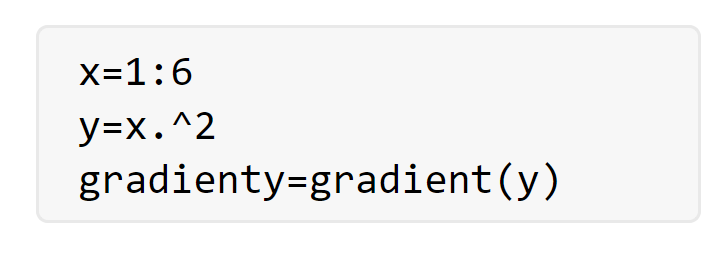
**-**

**+**

**-**

**+**

**-**



**+**

**-**

**/2**

**+**

**-**

**-**

**+**

**+**

**-**

**/2**

**+**

**-**

**/2**

**+**

**-**

**/2**

Beste Variante zum einfachen numerischen Ableiten:

gradient(y) ./ gradient(x)

## **Nicht äquidistante Messdaten**

Tangente

Sekante rechts

Sekante links

x

f (x)

xm

xr

xlm

Ist die Abtastschrittweite nicht konstant, dann schätzt man mit den umgekehrten Hebellängen:





Dieser Ansatz trägt dem Umstand Rechnung, dass der Punkt, der näher zum Tangentenpunkt liegt, stärker zu gewichten ist als einer, der weit weg liegt.

Herleitung über Parabelinterpolation

Xx

f (X)

Xm

Xr

xlm

xXx

Xlm

0

Es wird ein Koordinatensystem eingeführt, in dessen Ursprung sich der Punkt mit (Xm und Ym) befindet. Im xy Koordinatensystem soll nun eine Parabel bestimmt werden, welche durch die drei Punkte (xl, yl) (0,0) und (xr, yr) läuft.



Aus y(0)=0 folgt direkt c=0.

Die beiden anderen Punkte liefern ein Gleichungssystem für a und b:



Da die Ableitung im Punkt (0,0) bestimmt werden soll, wird nur b benötigt. Folglich wird a aus dem Gleichungssystem eliminiert.



Erste Zeile minus zweite Zeile:





 Steigungsergebnis nach Parabelinterpolation

 (umgekehrte Hebelarme)

umgeschrieben in x y System









 Steigungsergebnis aus umgekehrten Hebelarmen

Beide Verfahren für nichtäquidistant abgetastete Daten führen zu identischer Gleichung.

x=linspace(0,8,1000);

y=sin(x);

xa=(0:0.2:2).^3;

ya=sin(xa);

yp=cos(x);

ypgradient=gradient(ya)./gradient(xa);

hold off

plot(x,y)

hold on

plot(xa,ya,'o')

plot(x,yp,'g')

plot(xa,ypgradient,'d')

xl=[nan,xa(1:end-1)]-xa;

xr=[xa(2:end),nan]-xa;

yl=[nan,ya(1:end-1)]-ya;

yr=[ya(2:end),nan]-ya;

ypHA=(-yl.\*xr.^2+xl.^2.\*yr)./(-xl.\*xr.^2+xl.^2.\*xr)

plot(xa,ypHA,'+')

%plot(xl,gradient(yl)./gradient(xl),'+')

# Numerische Integration

Gegeben: Wertepaare t und y als Abtastung einer Funktion y=f(t)

Gesucht: Integralfunktion  an den Abtastpunkten.

Das Integral ist definiert als die Fläche zwischen einer Funktion und der x bzw. t-Achse.

t

a

X2

f(t)

X1

X3

Diese Fläche kann nicht exakt berechnet werden, wenn nur einzelne Abtastwerte bekannt sind, da der Verlauf zwischen den Abtastwerten dann eben nicht bekannt ist.

Es kann dann nur eine Ersatzfläche aus den Abtastpunkten berechnet werden. Mögliche Varianten hierzu sind:

**Rückgerichtete Integration**

Der linke Funktionswert eines Rechtecks bestimmt die Rechteckhöhe

t

a

X2

f(t)

X1

X3

**Vorwärtsgerichtete Integration**

Der rechte Funktionswert bestimmt die Rechteckhöhe

t

a

X2

f(t)

X1

**Zentrale Integration**

Der Mittelwert aus rechtem und linkem Funktionswert bestimmt die Rechteckhöhe.

Dies entspricht auch der der Fläche eines Trapezes.

t

a

X2

f(t)

X1

Die so entstandenen Trapeze enthalten bei positivem f(t) und rechtsgekümmter Kurve zu wenig Flächenanteile und bei positivem und linksgekümmten f(t) zu viel Flächenanteile.

Das numerische Integral ist die Summe aller Trapezflächen zwischen linker und rechter Integralgrenze.



 +

Wir führen folgende Koordinaten zur Beschreibung einer Trapezfläche ein:

xl

xr

yl

yr

In der Bezeichnung steht r für rechts und l für links.

Die Fläche des Trapezes ergibt sich als Summe aus dem grauen Rechteck und dem gelben Dreieck zu:



Bzw. nach Umformung:



Aus Gründen der einfacheren Übertragbarkeit ins Tabellenblatt wollen wir die Abtastpunkt und die Trapeze durchzählen:

t

a

f(t)

1

2

1

2

3

Dabei haben jeweils der linke Punkt eines Trapezes und ein Trapez die gleiche Nummer.

Die Integralfunktion kann nun auch folgenderweise geschrieben werden:

syms x a

int(sin(t), t, 0, x)

# Eulerverfahren

Gegeben sei eine DGL der Form

Mit u als Erregung und v als Antwort.

%% Definition System

a0=1;

a1=1;

b0=1;

b1=10;

b2=0;

%% Definition Zeitschritte

t=linspace(0,10,1000);

dt=t(2)-t(1);

%%Definition Erregung

u=t\*0;

u(10:end)=1; %Sprungerregung

%% Deklarationen und Anfangswerte

v=u\*0;

vp=u\*0;

vpp=u\*0;

v(1)=0;

vp(1)=0;

vpp(1)=0;

%% Ableitungen u

up=gradient(u)./gradient(t);

upp=gradient(up)./gradient(t);

%% Iterationen

for i=2:length(t)

v(i)=v(i-1)+vp(i-1)\*dt;

vp(i)=vp(i-1)+vpp(i-1)\*dt;

vpp(i)=b0\*u(i)+b1\*up(i)+b2\*upp(i)-a0\*v(i)-a1\*vp(i);

end

%% Darstellung

hold off

plot(t,v,'r')

hold on

plot(t,u,'b')

hold off

# (Rück)interpolation

Gegeben:

Abgetastete Werte xa und ya einer kontinuierlichen Funktion y(x). Gesucht y\_inter (xinter), das möglichst genau mit y übereinstimmt.

## Sinc-Interpolation

Wenn beim Erzeugen von xa und ya das Abtasttherorem exakt eingehalten wurde, gilt:

Bei Funktionen über der Zeit t schreibt sich die Formel zu:

Mit als Abtastzeit.

Da der zweite Summand innerhalb der sinc-Funktion bei äquidistanter Abtastung immer eine ganze Zahl ist, schreibt man oft auch nur den Index in die Berechnungsvorschrift:

Wenn der erste Index mit den Index 0 hat. Bzw.

Wenn der erste Index den Wert 1 hat, wie dies bei Matlab der Fall ist.

Beachte:



Abbildung ‑ Sinc-Interpolation mit Ein- und Auslaufverhalten

%% Quasikontinuierliche Kurven erzeugen

clear all

clc

dx=0.0125;

x=0:2000;

x=x\*dx;

y=cos(x);

yp\_ana=cos(x);

%% Abtastwerte erzeugen

av=100;

0; %Abtastverhältnis, jeder wievielte Wert der Ausgangskurve wird genommen?

xa=x(1:av:end);

ya=y(1:av:end);

%% sinc-Interpolation ausrechnen

deltax=xa(2)-xa(1);

xinter=x; % der Einfachheit halber und der besseren Vergleichbarkeit halber

yinter=zeros(1,length(x));

for j=1:length(xinter)

for i=1:length(xa)

yinter(j)=yinter(j)+ya(i)\*sinc(xinter(j)/deltax-(i-1));

end

end

%% Kurven darstellen

% hierzu gibt es im nackten C keine Entsprechung

hold off

plot(x,y,'r','DisplayName','sin')

hold on

plot(xa,ya,'Marker','o','LineStyle','none','Color','r','DisplayName','sin\_a');

plot(x,yinter,'.b','linewidth',3,'DisplayName','sincinterp.')

legend

%% sinc als Funktion

function [out]=sinc(in)

out=sin(pi\*in)/(pi\*in);

index=isnan(out);

out(index)=1;

end



Abbildung ‑ Lin-Interpolation

%% Quasikontinuierliche Kurven erzeugen

clear all

clc

dx=0.0125;

x=0:2000;

x=x\*dx;

y=cos(x);

yp\_ana=cos(x);

%% Abtastwerte erzeugen

av=100;

0; %Abtastverhältnis, jeder wievielte Wert der Ausgangskurve wird genommen?

xa=x(1:av:end);

ya=y(1:av:end);

%% lin-Interpolation ausrechnen

deltax=xa(2)-xa(1);

Na=length(xa);

Ni=av %nur weil einfach und vergleichbar

xinter=linspace(xa(1),xa(end),(length(xa)-1)\*(Ni)+1);

yinter=xinter\*0;

k=1;

for i=1:Na-1

for j=1:Ni

yinter(k)=ya(i)+(ya(i+1)-ya(i))\*(j-1)/(Ni);

k=k+1;

end

end

%% Kurven darstellen

% hierzu gibt es im nackten C keine Entsprechung

hold off

plot(x,y,'r','DisplayName','sin')

hold on

plot(xa,ya,'Marker','o','LineStyle','none','Color','r','DisplayName','sin\_a');

plot(xinter,yinter,'.b','linewidth',3,'DisplayName','Lininterp.')

legend

Vektorisierter Code

Statt

k=1;

for i=1:Na-1

for j=1:Ni

yinter(k)=ya(i)+(ya(i+1)-ya(i))\*(j-1)/(Ni);

k=k+1;

end

end

einfach nur:

yinter=interp1(xa,ya,xinter);

statt

yinter=zeros(1,length(x));

for j=1:length(xinter)

for i=1:length(xa)

yinter(j)=yinter(j)+ya(i)\*sinc(xinter(j)/deltax-(i-1));

end

end

nur

%% sinc-Interpolation ausrechnen 1

deltax=xa(2)-xa(1);

xinter=x; % der Einfachheit halber und der besseren Vergleichbarkeit halber

yinter=x\*0;

for i = 1:length(xinter)

yinter(i) = sum(ya .\* sinc((xinter(i) - xa) / (xa(2) - xa(1)))); %eingebauter Vektorf.

end

%% sinc-Interpolation ausrechnen 2

deltax=xa(2)-xa(1);

xinter=x; % der Einfachheit halber und der besseren Vergleichbarkeit halber

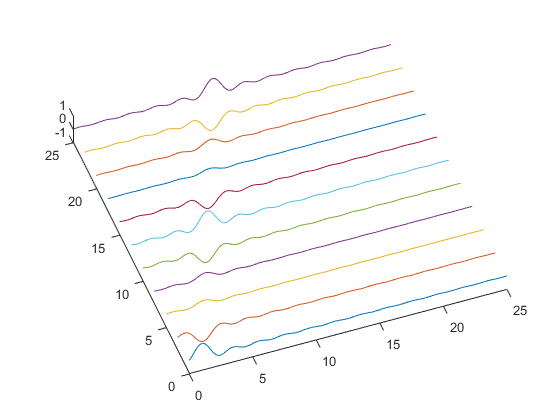
yinter=x\*0;

[Xinter,Xa] = ndgrid(xinter,xa);

yinter = sinc((Xinter - Xa)/deltax)\*ya'

Einfaches Beispiel weniger Werte





statt:



Abbildung ‑ sinc-Interpolation einer Rechteckschwingung

Spezialfrage: warum sind die positiven und die negativen Halbwellen der Interpolation nicht identisch?

## Lineare Interpolation

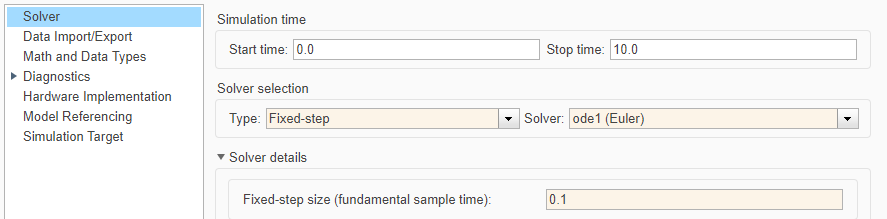
Präsenzaufgabe: ergänzen Sie selber den m-Code für die lineare Interpolation!

Abbildung ‑ lineare Interpolation einer Rechteckschwingung

# Simulink Spezialitäten

## Simulink Debugger

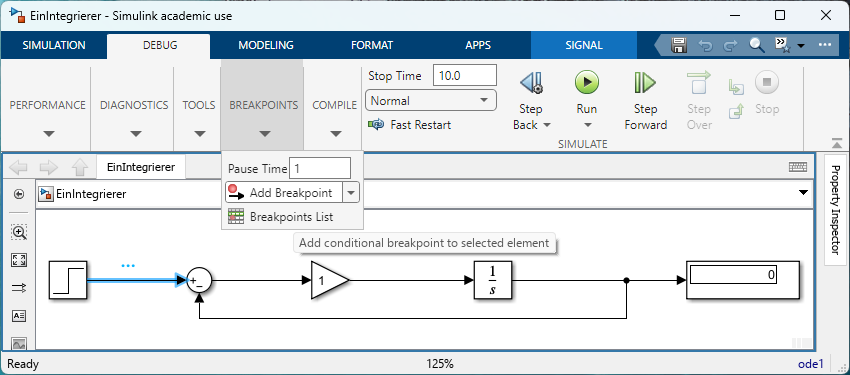
### Einstellungen für Euler



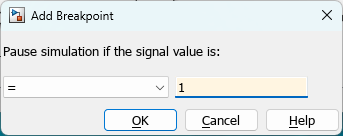
### Breakpoint setzen

Wirkungslinie (Signal) auswählen

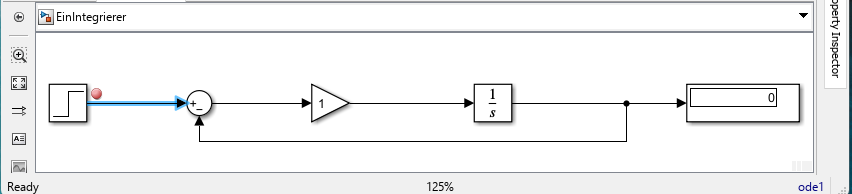
Debug/Breakpoints/Add Breakpoint



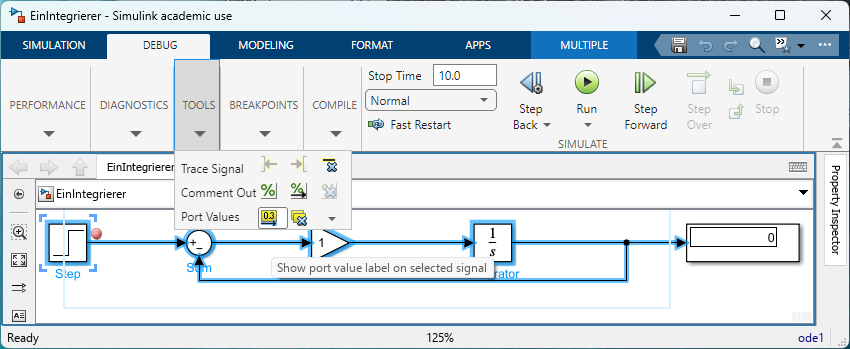
Haltebedingung eingeben

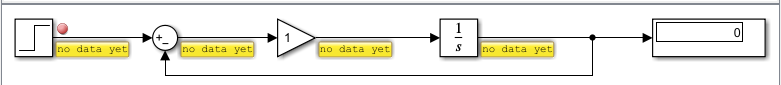


Breakpoint ist gesetzt

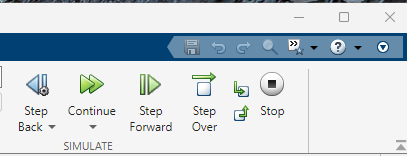


### Anzuzeigende Signale auswählen (Show port values)





### Debugsequenz starten

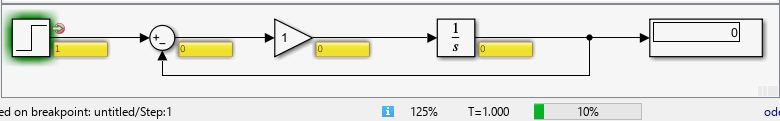


Nur, wenn die Simulation vor einem Haltepunkt pausiert, sind die Schaltflächen „Step Over“, „Step into“ und „Step out“ aktiv. Mit Ihnen kann ein einzelner Simulink-Block übersprungen werden. Bzw. Es kann in Ihn hinein gesprungen werden oder wieder heraus gesprungen.

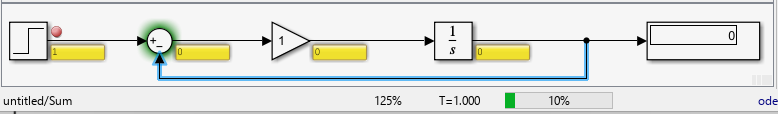
Nach einem einzelnen Blockschritt bleiben die Schaltflächen aktiv. War dies jedoch der letzte Block innerhalb eines Zeitschritts, werden sie wieder inaktiv, bis der Zeitschritt ausgeführt ist.

### Debugsequenz mit einem Integrator

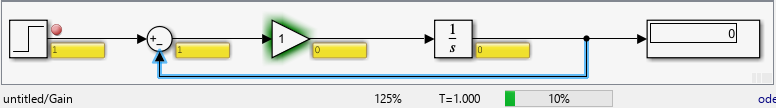
Wie bei textbasierten Entwicklungsumgebungen (IDE), stoppt die Simulation vor einem Haltepunkt. Hier hat die Sprungquelle bereits eine 1 ausgegeben, der Summierer hat darauf aber nicht reagiert.



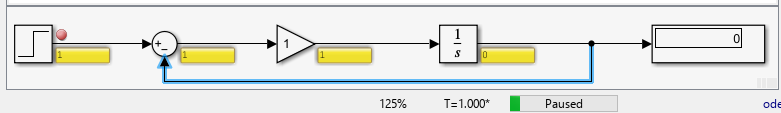
Ein „Step over“ weiter rechnet der Summierer nun mit der neuen Eingangsgröße…



Und wiederum ein „Step over“ weiter rechnet der Gain-Baustein

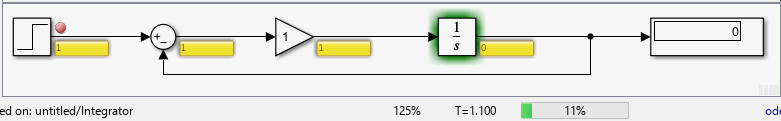


Noch ein „Step over“ liegt das neue Eingangssignal am Integrierer an. Damit kann nun ein neuer Zeitschritt starten. „Step over“ ist inaktiv, bevor der neue Zeitschritt mit „Step Forward“ gestartet wird. Der Stern bei T=1.000\* deutet an, dass der Eingangs des Integrieres nun schon den Wert für den nächsten Zeitschritt hat..

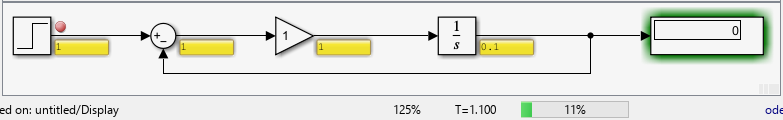


Nun erfolgt der Zeitschritt mittels „Step Forward“

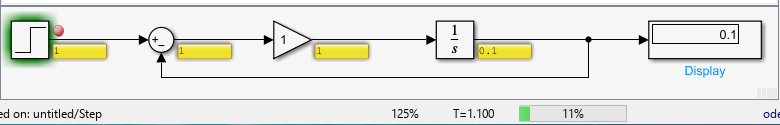
Update startet beim Integierer



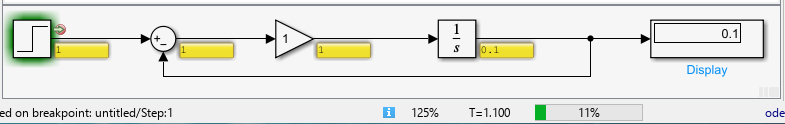
Nachfolgeblöcke mit nur einem Eingang werden aktualisiert



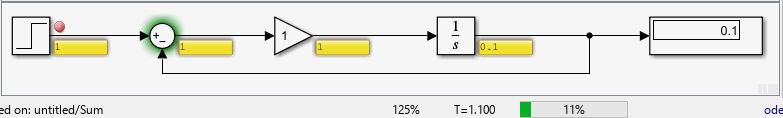
Blöcke die für weitere Eingänge der Nachfolgeblöcke wichtig sind

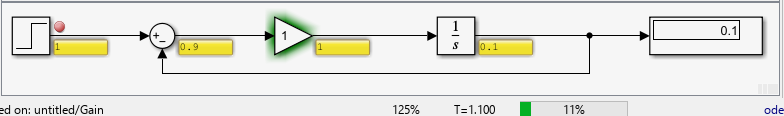


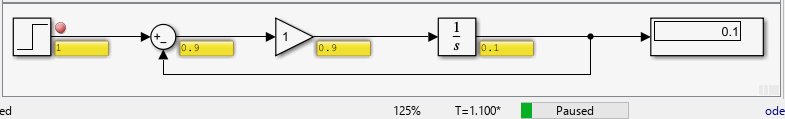
Stopp am Haltepunkt (Bereakpoint)



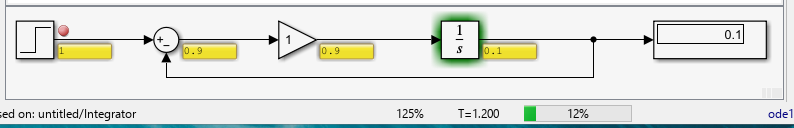
Weiter in Signalflussrichtung..







Zeitschritt ist fertig, nun geht der nächste los



## Simulationsbeispiel: Totzeitglied mit P-Regler

Bunker

Schieber

Transportband

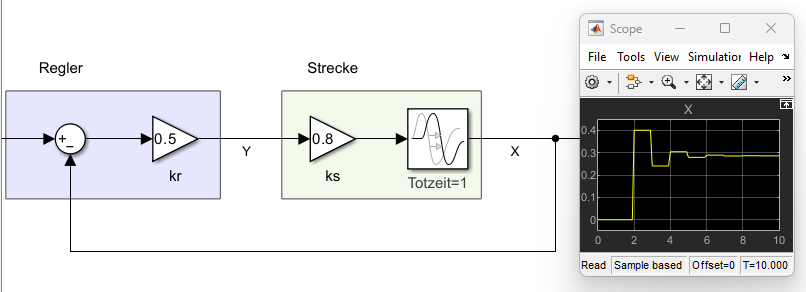
Abgabestelle

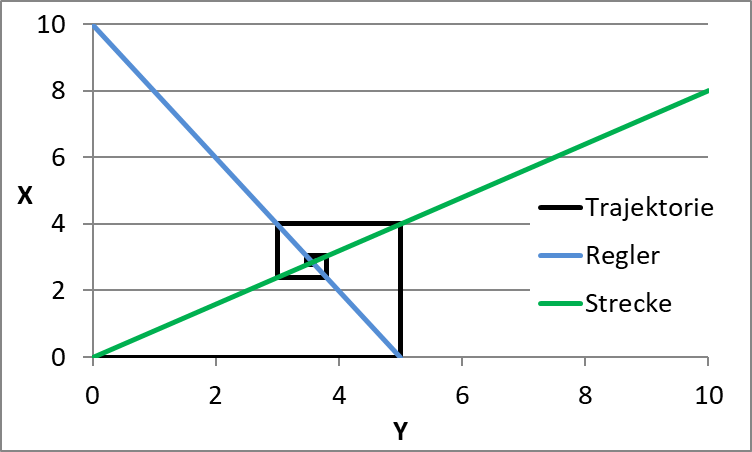
X

Y

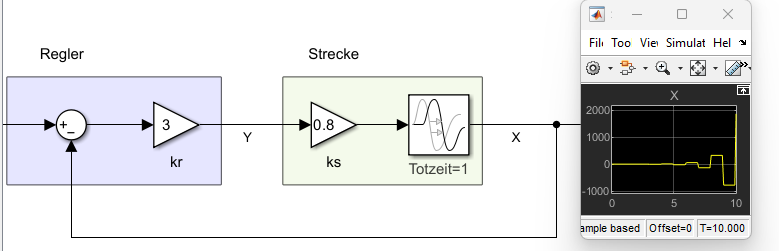
Regelstrecke:

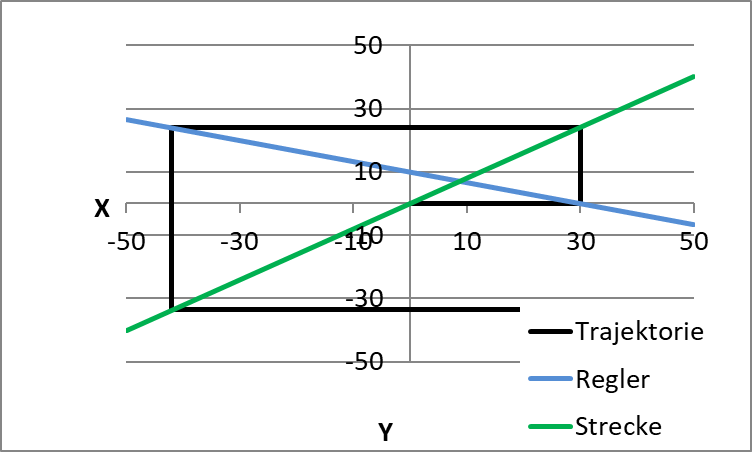
Regler:





Trajektorie bei kr=0,5 und ks=0,8



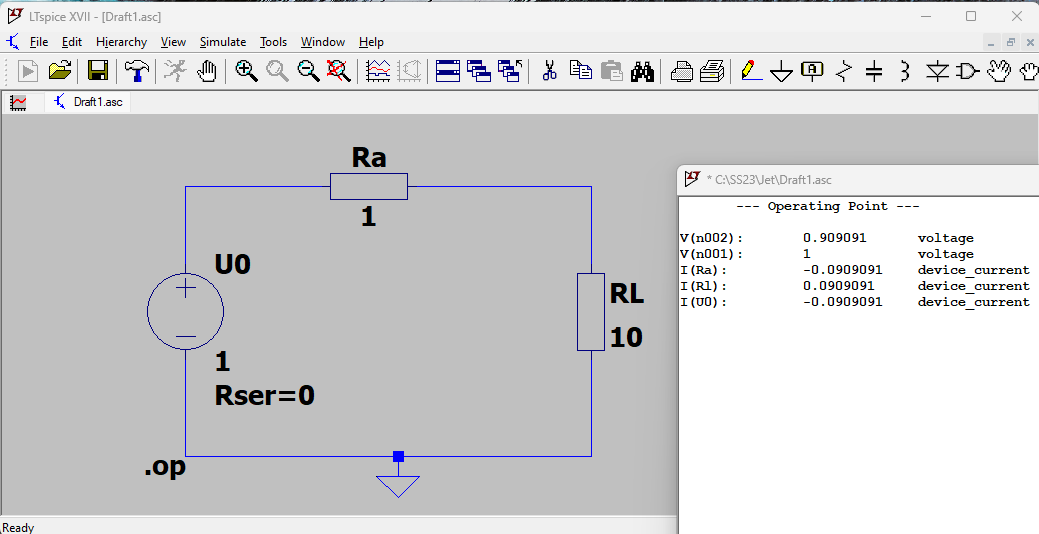


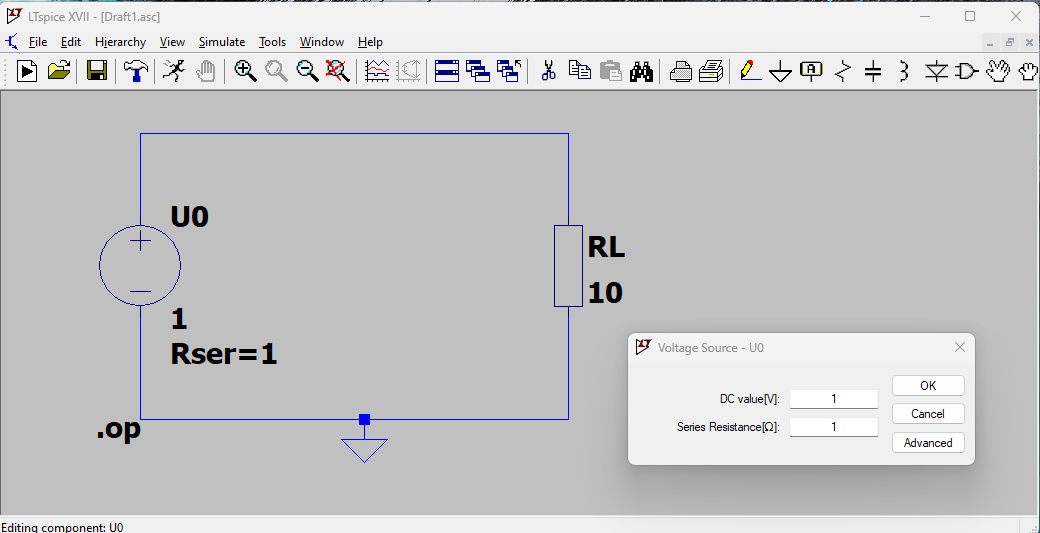
Trajektorie bei kr=3 und ks=0,8

## Simulationsbeispiel: Reale Spannungsquelle

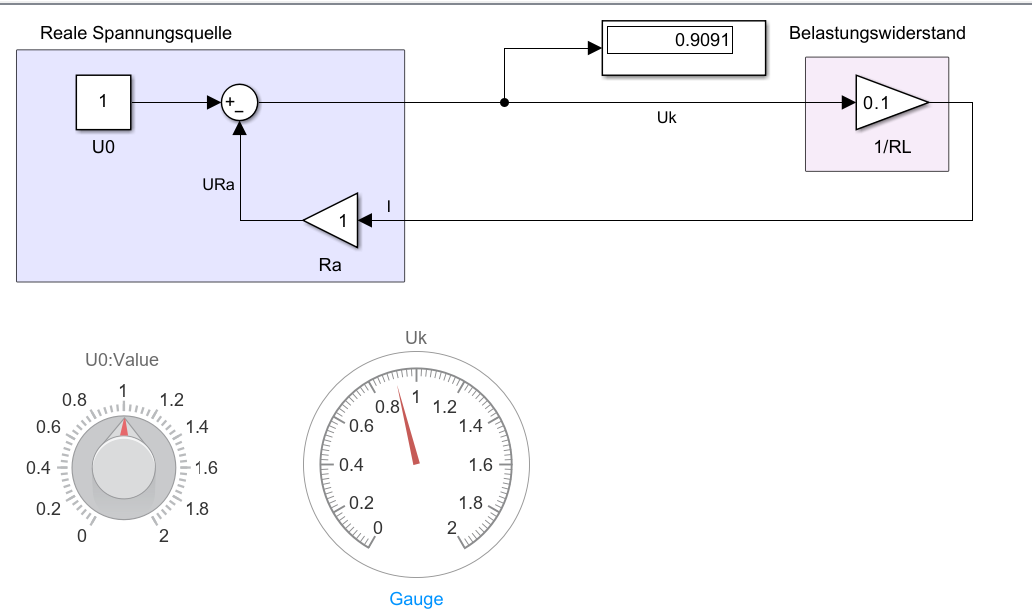
Simulink betont die











## Algebraic Loop

# Matlab Spezialitäten

## Nutzung neuer Funktionen in alten Versionen

Ab dem Release 2023 a gibt es zum Beispiel die Funktion combinations, die viele Brute Force Aufgaben vereinfacht.

Muss auf diese Funktion in alten Matlab-Versionen verzichtet werden?

Nein, wenn man einmal Zugriff auf Release 2023 a oder neuer hat, kann man die Funktion editieren mit dem Kommando >>edit combinations

function T = combinations(varargin)

%COMBINATIONS Generate all element combinations of arrays

%

% T = COMBINATIONS(A1, A2, ..., An) returns all element combinations.

% Each row in output table T is a combination with the first element

% coming from A1, the second element coming from A2, and so on. The

% number of rows in T is the product of the number of elements in each

% input, and the number of variables in T is the number of inputs.

%

% The inputs can be vectors, matrices, or multidimensional arrays. If an

% input Ak is not a vector, then COMBINATIONS treats it as a single

% column vector, Ak(:).

%

% Examples:

% % All element combinations of two vectors

% T = combinations([2 1], [5 4 6])

%

% % All element combinations of inputs of mixed datatypes

% A1 = [0 1; 2 3];

% A2 = ["a" "b"];

% A3 = categorical(["x" "y"]);

% T = combinations(A1, A2, A3)

%

% See also REPELEM, REPMAT, PERMS, NCHOOSEK, NDGRID, MESHGRID

% Copyright 2022-2023 The MathWorks, Inc.

narginchk(1,Inf);

numVars = nargin;

varNumel = zeros(1,numVars);

input = varargin;

for k = 1:numVars

varNumel(k) = numel(input{k});

end

m = prod(varNumel);

% Gracefully error like in perms()

[~,maxsize] = computer;

if m\*numVars > maxsize

error(message('MATLAB:pmaxsize'))

end

vars = cell(1,numVars);

varNames = cell(1,numVars);

noName = false(1,numVars);

for k = 1:numVars

try

var = input{k}(:);

catch

error(message('MATLAB:combinations:InvalidType',class(input{k})));

end

r = repelem(var,prod(varNumel(k+1:end)),1);

vars{:,k} = repmat(r,prod(varNumel(1:k-1)),1);

varNames{k} = inputname(k);

noName(k) = isempty(varNames{k});

end

% Construct output table containing the combinations

% Use default table variable names for inputs with no name

numNoName = sum(noName);

if numNoName > 0

vdim = matlab.internal.tabular.private.varNamesDim;

varNames(noName) = vdim.dfltLabels(find(noName));

end

% Disambiguate reserved names for table variables

if numNoName < numVars

mdim = matlab.internal.tabular.private.metaDim;

rlabels = mdim.labels;

rnames = matlab.internal.tabular.private.varNamesDim.reservedNames;

r = [rlabels rnames];

varNames = matlab.lang.makeUniqueStrings(varNames,r,namelengthmax);

end

T = table.init(vars,m,{},numVars,varNames);

end

Die Funktion kann nun wie eine eigene Funktion abgespeichert und unter älteren Matlab-Versionen benutzt werden. Ob dies immer und überall legal ist, weiß ich nicht. Aber zumindest solange auch die neuere Matlab-Version lizensiert ist, sie aber noch nicht auf z. B. Clusterrechnern installiert ist, sollte das in Ordnung sein.

Wie weit kann man auf diese Art in den Versionen zurück gehen?

Das hängt davon ab, welche anderen neuen Matlab-Funktionen innerhalb von combinations benutzt werden.

|  |  |
| --- | --- |
| Funktion / Datentyp / Konstrukt | Erstmals in Matlab- |
| narginchk | **R2011b** |
| nargin | before R2006a |
| varargin | before R2006a, improved R2023a |
| prod | before R2006a, improved R2023a, R2018b |
| repelem | R2015a |
| repmat | before 2006a, improved R 2019b |
| table | R2013b, several improvements |

Im Beispiel scheint es möglich, bis einschließlich R2015a die Funktion „combinations“ unverändert zu benutzen. Die Releasdaten der einzelnen Funktionen stehen auf der entsprechenden Hilfeseite. Bei späteren „Improvements“ muss man dann ausprobieren oder nachlesen, aus was konkret diese bestanden haben.

# Aufstellen von Whiteboxmodellen

## Mechanische Systeme

### Dynamische Grundgleichungen

|  |  |  |
| --- | --- | --- |
|  |  |  |

Mehrere Kräfte/Momente -> Summierglied

Geschwindigkeit und Position

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

m

F

v

s

M

f

w

Abbildung ‑ Zählpfeile Translation und Rotation

**Übung Dynamik I**

Auf einen Körper der Masse m=2kg wirke eine Kraft F(t) gemäß



Die Anfangsgeschwindigkeit zum Zeitpunkt t=0 beträgt -2m/s

Die Anfangsposition zum Zeitpunkt t=0 beträgt 0 m.

Stellen Sie die Position des Körpers über t mittels Simulink dar!

**Übung Dynamik II**

Auf den Körper der vorhergehenden Übung wirke eine Kraft mit dem unten eingezeichneten Verlauf.

Ermitteln Sie den Zeitverlauf der Position des Körpers!

Zum Zeitpunkt Null habe der Köper die Position Null und die Geschwindigkeit Null.

F

t

1s

5s

3s

1N

-1N

### Federn

Eine Feder als Maschinenelement kann über das Verhalten des Modellelements Feder hinaus auch Verhalten des Modellelements Dämpfer zeigen.

Bemerkung: Dargestellt ist die Kraft, mit der an der Feder gezogen wird. Die Kraft, mit der die Feder auf den Ziehenden rückwirkt (action/reactio) hat das umgekehrte Vorzeichen.

X2

F

F

X1

s

#### Translatorische Feder

(VZS)

(EZS)

U

I

VZS

U

I

EZS

Abbildung ‑ Zählpfeile translatorische Feder

**Einfachste** **Hook’sche Feder:**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Berücksichtigung einer Länge L0 bei entspannter Feder**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Progressive Feder**

z. B. Gummifeder, Gasfeder, speziell geformte Feder

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Degressive Feder**

|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Torsionsfeder

M



M



Abbildung ‑ Zählpfeile Torsionsfeder

|  |  |  |
| --- | --- | --- |
|  |  |  |

### Übung Dynamik 3

Berechnen Sie den Verlauf von x über der Zeit, wenn die Masse 2 kg beträgt und die Kraft sich sprungförmig (zum Zeitpunkt t=1s) von 0 auf 1 N ändert!

X

F

m

C

Die Federkonstante C betrage 3 N/m.

Zu Beginn befindet sich die Masse bei x=0 in Ruhe.

### Beispiel Druckfeder zwischen Massen

m2

m1

C, L0

a, v, s



### Dämpfer

Bewegungen werden durch Reibung gebremst. Die Bremskraft kann in unterschiedlicher Art von der Relativgeschwindigkeit zwischen Körper und Umgebungsmedium abhängen:

#### Statische Haftreibung

**1**

**2**

v1=v2=0

FN

FN

Abbildung ‑ Zählpfeile Haftreibung



Es wird vorausgesetzt, dass FN größer als Null ist, ohne Normalkraft gibt es keine Haftreibung.

Übersteigt FS die maximale Haftreibung, erfolgt ein Übergang zu einer anderen Reibungsart.

Bleibt die Geschwindigkeit sehr klein, kann die Reibkraft oft auf den Bereich ohne Haftung ausgedehnt werden:



|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Dynamische Reibung

##### Dynamische Haftreibung (kleine Geschwindigkeiten)

**1**

**2**

v2

v1

Fd

FN

Abbildung ‑ Zählpfeile Gleitreibung

|  |  |
| --- | --- |
|  |  |

##### Gleitreibung (große Geschwindigkeiten)

**1**

**2**

v2

v1

Fd

FN

Abbildung ‑ Zählpfeile Gleitreibung

|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

#### Viskose Reibung

v

Fd

Abbildung ‑ Zählpfeile viskose Reibung



|  |  |  |
| --- | --- | --- |
|  |  |  |



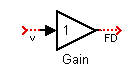


Abbildung ‑ WP viskose Reibung

#### Windwiderstandsreibung

Zählpfeile siehe Abbildung 9‑7.



|  |  |  |
| --- | --- | --- |
|  |  |  |





Abbildung ‑ WP Windwiderstand

#### Beliebiger Exponent



Abbildung ‑ WP beliebiger Exponent

****

Abbildung ‑ Zeitverlauf und Kennlinie für Kappa=2

#### Übergang Haftreibung <-> Gleitreibung (Stick Slip)

Ein abrupter Wechsel von Haft- auf Gleitreibung findet nicht statt. Vielmehr ist in einem bestimmten Geschwindigkeitsbereich beides wirksam. Die einfachste Modellierung benutzt dazu eine gegenläufige Wichtung:

|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

, die exponentiell von der Geschwindigkeit abhängt:

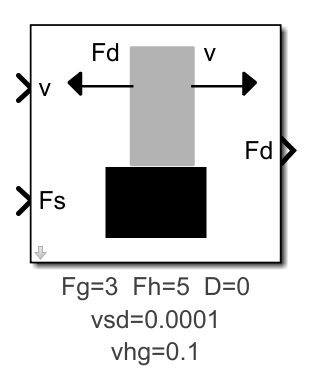
|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

dabei ist vu =vhg die Geschwindigkeit, bei der die Haftreibung auf exp(-1)=36,8% abgefallen ist.

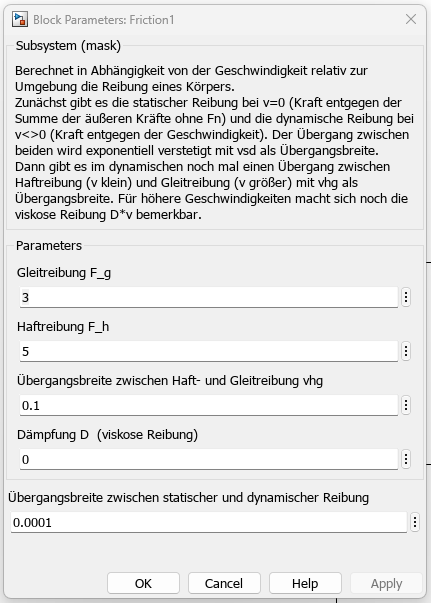


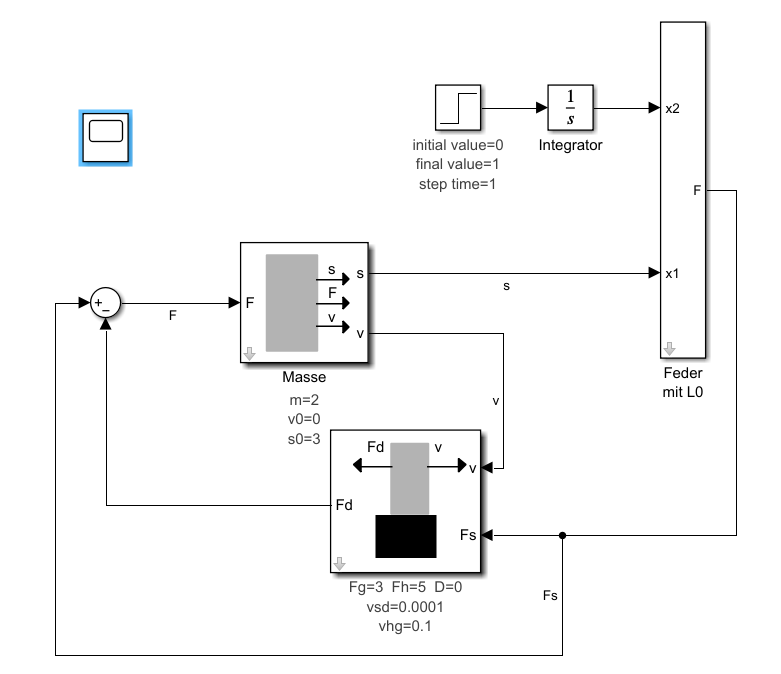
Abbildung ‑ Gewichtung von Haftreibung und Gleitreibung

**Reibblock aus der Blocksammlung -Übergang der Reibungsarten**

****







#### Kennlinie des Reibungsbausteins:



Abbildung ‑ WP zur Kennlinienmessung des Reibungsbausteins



Abbildung ‑ Kennlinie des Reibungsbausteins

Der Punkt bei v=0 -sprich Fd=Fs (statische Haftreibung)- ist nur dann als Markierung im Diagramm sichtbar, wenn die Schrittweite so gewählt wurde, dass v=0 exakt getroffen wird.

#### Stick Slip

V2=const

Hand

Tisch

Feder

µh bzw. µg

L0=3

x

X2,0=0

X1,0=-3

Rucksack m

Abbildung ‑ Stick Slip Anordnung



Abbildung ‑ WP der Stick Slip Anordnung



Abbildung ‑ Signalverläufe bei Stick Slip

### **Übung Dynamik 4**

F

s

C

D

m

### Übung Dynamik 5

m2

m1

F

C2

C1

D2

D1

x1

x2

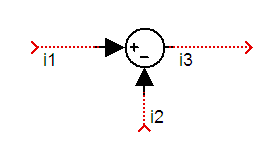
## Elektrische Systeme

Annahme:

Konzentrierte Elemente  
(elektrisches Feld existiert nur *innerhalb* von Widerstand, Kondensator oder Spannungsquelle, magnetisches Feld existiert nur innerhalb von Spule)

### Knotenpunktsatz, 1. Kirchhoffsche Regel

|  |  |  |
| --- | --- | --- |
|  |  |  |



i1

i2

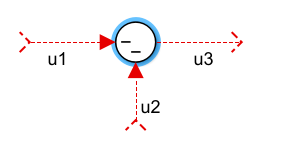
i3

Abbildung ‑ Knotenpunktsatz

Im Wirkungsplan muss genau einer der Ströme als Ausgangstrom gewählt werden.

### Maschensatz, 2. Kirchhoffsche Regel

|  |  |  |
| --- | --- | --- |
|  |  |  |



U1

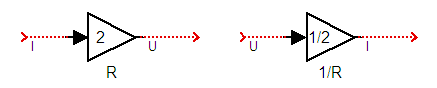
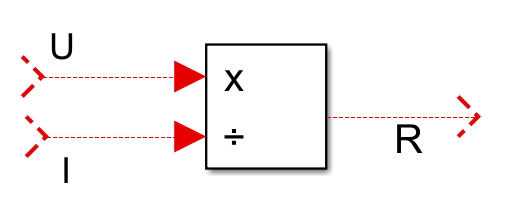
U2

U3

Abbildung ‑ Maschenumlauf

### Bauteile

#### Widerstand



U

I

Abbildung ‑ Ohm'scher Widerstand

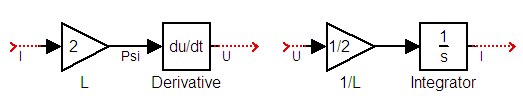
|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

#### Spule

|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

|  |  |  |
| --- | --- | --- |
|  |  | Gl. |

|  |  |  |
| --- | --- | --- |
|  |  | Gl. |



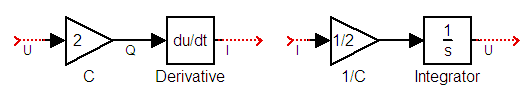
U

I

Abbildung ‑ Spule

#### Kondensator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  |  | |
|  | |  |  | |
|  |  | | |  | |

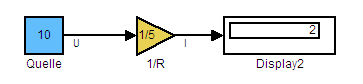


U

I

Abbildung ‑ Kondensator

#### Ideale Spannungsquelle:



*Wirkungsplan*

U=10V

R=

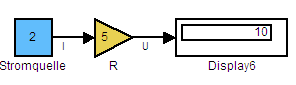
*Schaltplan*

IU

Abbildung ‑ Ideale Spannungsquelle

#### Ideale Stromquelle:

*Wirkungsplan*



U

R=5

*Schaltplan*

I=2AU

Abbildung ‑ Ideale Stromquelle

#### Reale Spannungsquelle:

U0

Ra

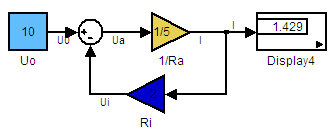
*Schaltplan*

IU

Ri

Ua

Ui



*Wirkungsplan*

Abbildung ‑ Reale Spannungsquelle



|  |  |  |
| --- | --- | --- |
|  |  |  |

Variante mit Ua als Ausgangsgröße

|  |  |  |
| --- | --- | --- |
|  |  |  |

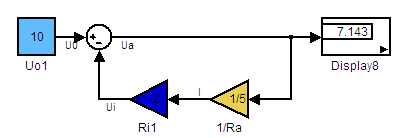


Abbildung ‑ Reale Spannungsquelle mit Ua als Ausgang

Darstellung mit Superblock

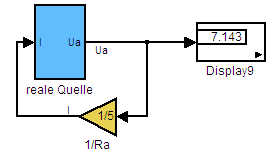
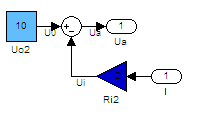
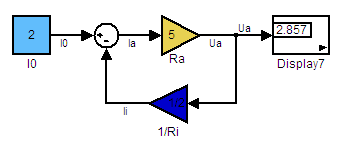


Abbildung ‑ Reale Spannungsquelle als Superblock

**Stromquelle**



Ua

Ra

*Schaltplan*

I0=2A

Ri

Ii

Ia

*Wirkungsplan*

Abbildung ‑ Stromquelle

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

### Beispiel Filter zweiter Ordnung

Uein

Uaus

R1

R2

C1

C2

## Hydraulische Systeme

Folgt noch…

# Brute Force

Siehe separates Powerpoint.

# Monte Carlo

Siehe separates Powerpoint

# Gleichungssysteme

Siehe separates Powerpoint

# Least Squares

Siehe separates Powerpoint

# Weitere Quellen

[Beyond the Code: Showcasing Your MATLAB Results Like a Pro - YouTube](https://www.youtube.com/watch?v=IVOebVU3QVY)

# Liste der Matlabbefehle

## Vektor / Matrixbefehle

M=[] Blanko Deklaration

M=[1,2;3,4] M ist die Matrix

A=[s1,s2,s3] Baut aus Spalten s1, s2 und s3 eine Matrix zusammen.   
s1, s2, s3 müssen die gleiche Anzahl an Zeilen haben.

[d1, d2]= size (M) Zeilenzahl und Spaltenzahl einer Matrix.  
(bzw. alle array-Abmessungen)

length(x) Länge des Vektors x (Zeilenzahl bei Spaltenvektoren, Spaltenzahl bei Zeilenvektoren, bzw. größte array-Abmessung

ones(z,s) Matrix aus z Zeilen und s Spalten, die aus lauter Einsen besteht

zeros(z,s) Matrix aus z Zeilen und s Spalten, die aus lauter Nullen besteht

randn(z,s) Matrix aus z Zeilen und s Spalten, die aus lauter standard-normalverteilten Zufallszahlen besteht.

rand(z,s) Matrix aus z Zeilen und s Spalten, die aus lauter gleichverteilten Zufallszahlen besteht ( 0 bis 1).

x=M(:,n) n-te Spalte von M

x=M(n,:) n-te Zeile von M

x=5:2:17 Zeilenvektor, erster Wert ist 5. Letzter ist 17, Schreitweite ist 2

x=(5:2:17)‘ Spaltenvektor, erster Wert ist 5. Letzter ist 17, Schreitweite ist 2

x([3,7,9]) Vektor aus dem 3. , 7. Und 9. Element von x (Das heißt x muss mindestens 9 Elemente haben) Das Ergebnis hat 3 Elemente.

x=linspace(1,13,5) Zeilenvektor aus Werten gleichen Abstands von 1 bis 13 mit   
 insgesamt 5 Elementen

x=x(10:end) Die ersten 9 Elemente werden weggeschnitten. Entspricht x=x(10:length(x)).

x(4:17) vierter bis siebzehnter Wert von x

x(ia:ie)=[] Löschen aller Elemente von x mit Index zwischen ia und ie

p=A\y Berechnet die Lösung p des (auch überbestimmten) Gleichungssystem A\*p=y

.\* ./ .^ elementweise Ausführung der entsprechenden Operationen

## Logical Indexing

Siehe Aufgabenblatt

## Integrieren/Differenzieren

xp=gradient(x)./gradient(t) Ableitung des Vektors xnach dem Zeitvektor t. Berechnet zu jedem Punkt x(n),t(n) die Steigung des Sekante zu den Nachbarpunkten. xp(n)= (x(n+1)-x(n-1))/(t(n+1)-t(n-1)). An den Randpunkten wird der nicht vorhandene Nachbarpunkt durch durch x(n),t(n) ersetzt. Weitere Details siehe Extrakapitel.

xi=cumtrapz(t,x) (genähert über Trapezregel)

## Ein-Ausgabe, Textverarbeitung

strcat(Text1,Text2) Verketten von zwei Texten zu einem Gesamttext nachfolgende Leerzeichen werden ignoriert.

horzcat(Text1,Text2) Hier: verketten von zwei Texten zu einem Gesamttext nachfolgende Leerzeichen werden beachtet.

num2str(Zahl) Umwandlung einer Zahl zu einem Text

disp(Text) Anzeigen eines Textes im CommandWindow

input(Text) Bediener wird mit Text (Eingabeprompt) aufgefordert eine Zahl einzugeben. Rückgabewert ist die Zahl als Text. -> Einlesen einer Zahl als Zahl: num2str(input(Text))

load(Dateiname.mat) Lädt einen Matlab Workspace (Satz von Variablen)

## Plotbefehle

plot(t,x) Stellt x über der Zeit t dar

axis equal erzwingt winkeltreue Darstellung

'DisplayName','xp\_n' Vorgabe des Kürzels, das später in Legende steht

'MarkerSize',20

'Marker','none' oder: ‘\*’, ‘.’, ‘+’, ‘x’, ‘s’=’square’, ‘d’=’diamonds’, ‘^’, ‘v’, ‘>’, ‘<’, ‘p’=’pentagram’, ‘h’=hexagram

'LineStyle','none' oder: ‘-‘, ‘--‘, ‘:’,’ -.’

‘LineWidth’, 0.5

'Color',[0 0 1] oder: ‘y’, ‘m’, ‘c’, ‘r’, ‘g’, ‘b’, ‘w’, ‘k’

legend('show') Zeigt Legende in aktuellem Graphen an

ylim([-0.1,1.1]) Grenzen für y-Achse

[ax,h1,h2]=plotyy(t,U,t,Y) Sequenz für zwei Kurven mit Sekundärachse  
set(ax(1),'YColor','b')   
set(ax(2),'YColor','k')   
set(h1,'Color','b','LineWidth',3)   
set(h2,'Color','k','LineWidth',3)  
set(ax,'FontSize',20)

## Sonstige

fix(x) Schneidet Nachkommastellen von x ab (z. B. für Indexvariable)

Mean(x) Mittelwert(x)

## Eigene Funktionen

function[out1,out2,…] = funktionsname(in1,in2,…)

# Programmierkonstrukte in verschiedenen Sprachen

|  |  |  |  |
| --- | --- | --- | --- |
| **C++** | G (Labview) | Matlab | VBA |
| **Deklaration und Datentypen** | | | |
| **bool a, b;**  **long c, d;**  **float e, f;**  **double g, h;**  **char\* text;** | TF=Boolean=grün  Rechtsklick/Darstellung  I32=integer=blau  Sgl=Real=orange  Dbl=Real=*orange*  abc=String=violett | a=logical(1); b=logical(1)  c=uint32(1);d=uint32(1)  e=single(pi); f=single(pi)  g=double(pi); h=double(pi)  text=num2str(17); | Dim a,b as boolean  Dim c,d as long;  Dim e,f as float  Dim g,h as double |
| **For Schleife und Kommentare** | | | |
| **for (n = 1; n <= k; n++)**  **{**  **a = a + n;**  **}**  **//Dies ist ein Kommentar** |  | for n=1:1:k  a=a+n  end  % Dies ist ein Kommentar | For n=1 to k step 1  a=a+n  Next  ‘Dies ist ein Kommentar |
| **Abweisende Schleife** | | | |
| **while (n <= k)**  **{**  **a = a + n;**  **n = n + 1;**  **}** | *Keine abweisende Schleife in Labiew* | while n<=k  a=a+n  n=n+1  end | While n<=k  a=a+n  n=n+1  Wend |
| **Einladende Schleife** | | | |
| **do**  **{**  **a=a+n;**  **n=n+1:**  **}**  **while (n<=k)** | *Zuerst wird iteriert, dann erst Schleifenkern ausgeführt.* | *Keine einladende Schleife in Matlab* | Do  a = a + n  n = n + 1  Loop While n <= k |
| **Verzweigung** | | | |
| **int main(void)**  **{**  **int n, A;**  **cout << "Bitte n eingeben";**  **cin >> n;**  **if (n<0)**  **{**  **cout << "n muss >0 sein";**  **}**  **else if (n % 2 == 0)**  **{**  **A = n / 2;**  **}**  **else**  **{ A = (n + 1) / 2;**  **}}** |  | n=input('geben Sie n ein\n');  n=int16(n)  if n<0  disp('n muss >0 sein');  elseif rem(n,2)==0  A=n/2  else  A=(n+1)/2;  end | Sub test()  Dim n As Integer  n = InputBox("Gebe n ein")  If n < 0 Then  MsgBox ("n muß >0 sein")  ElseIf n Mod 2 = 0 Then  A = n / 2  Else  A = (n + 1) / 2  End If  End Sub |
| **Funktionnen und Felder** | | | |
| **int findmax(int a[]);**  **int \_main(void)**  **{ int a[4];**  **a[0] = 4; a[1] = 17;**  **a[2] = 42; a[3] = 21;**  **int b = findmax(a);**  **return 0;**  **}**  **int findmax(int a[])**  **{int max = a[0];**  **for (int i = 0; i <= sizeof(a); i++)**  **{ if (a[i] > max)**  **{max = a[i];}}**  **return max;}** |  | function Haupt  x=int16 (ones(4,1));  x(1)=4; x(2)=17;  x(3)=42; x(4)=21;  b=findmax(a)  end    function =findmax(a)  c=a(1);  for i=1 : length(a)  if a(i)>c  c=a(i);  end  end  end | Sub Haupt()  Dim a() As Variant  a = Array(4, 17, 42, 21)  b = findmax(a)  End Sub  Function findmax(Feld() As Variant)  Max = Feld(1)  For i = LBound(Feld) To UBound(Feld)  If Feld(i) > Max Then Max = Feld(i)  Next  findmax = Max  End Function |

Abbildung ‑ Syntax vierer Programmiersprachen

# Nachbesprechung der Übungen

## **PM1 Primzahlen**

Sie können sich die Primfaktorenzerlegung der Zahlen in AlmostPrime mit dem Befehl  
y = arrayfun(@factor, AlmostPrimes, 'UniformOutput', false)

anschauen.

“arrayfun” stellt im Prinzip das für Funktionen dar, was der „.“ Operator für Operatoren ist. Im ersten Argument wird ein Funktionshandle (@factor) der Funktion (factor) übergeben, die elementweise berechnet werden soll. Letzten Endes wird auf jedes Element von AlmostPrimes die Funktion factor angewendet.

Würde factor nur einen skalaren Wert zurück liefern, wäre die Ergänzung 'UniformOutput', false nicht notwendig. Wenn nicht alle Rückgabewerte der Funktion skalar sind, muss die Ergänzung dazu. Die Ergebenisse werden dann in einem Cell-Array (array mit Elementen unterschiedlichen Datentyps) abgelegt.

# Literatur

**Arndt, Bröder. 2011.** *Versuchsplanung und experimentelle Praktikum.* Göttingen : Hogrefe Verlag GmbH, 2011.

**Box, George E. P., Hunter, J. Stuart und Hunter, William G. 2005.** *Statistics for Experimenters.* Hoboken, New Jersey : J. Wiley, 2005. ISBN 978-0-471-71813-0.

**Brockh. 2007.** *Der Brockhaus, Universallexikon.* Leipzig : F. A. Brockhaus GmbH, 2007.

**Dubben, Hans-Herrmann und Becl-Bornhold, Hans-Peter. 2010.** *Der Hund, der Eier legt.* Hambuerg : Rowohlt Taschenbuch Verlag, 2010. ISBN 978 3 499 62196 3.

**Field, Andy und Hole, Graham. 2010.** *How to Design and Report Experiments.* London : Sage Publications LTD, 2010.

**Kleppmann, Wilhelm. 2009.** *Taschenbuch Versuchsplanung.* München : Carl Hanser, 2009. ISBN 978-3-446-42033-5.

**Lunze, Jan. 2010.** *Regelungstechnik 2.* Heildelberg : Springer, 2010. e-ISBN 978-3-642-10198-4 (für Studenten der HS RM kostenlos verfügbar).

**MEYERS. 1974.** *MEYERS Enzyklopädisches Lexikon.* Mannheim : Bibliographisches Institut AG, 1974.

**Siebertz, Karl, van Bebber, David und Hochkirchen, Thomas. 2010.** *Statistische Versuchsplanung.* Heidelberg : Springer, 2010. e-ISBN 978-3-642-05493-8 (für Studenten der HS RM kostenlos verfügbar).