Einführung in MATLAB + MATLAB Simulink

Dipl.-Inf. Markus Appel

mappel@informatik.hu-berlin.de

27.10.2017

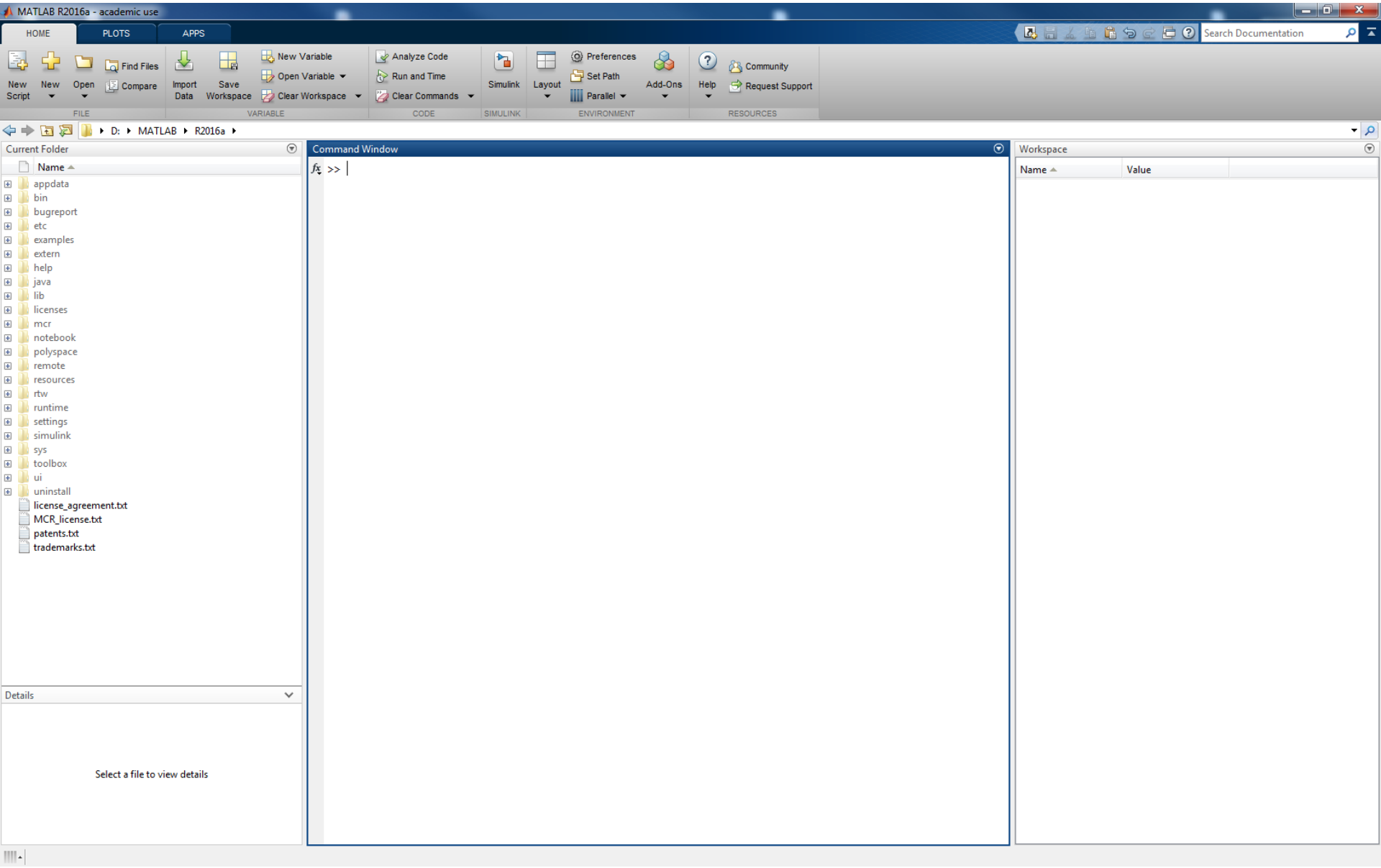
Was ist MATLAB?

- ein universelles Algebra-Programm
  - zur Lösung mathematischer Probleme
  - grafische Darstellung der Ergebnisse
- es wurde in den 70er Jahren am der University of New Mexico und der Stanford University entwickelt
- ist in erster Linie für numerische Berechnungen mit Hilfe von Matrizen ausgelegt
- Name: „**MAT**RIX **LAB**ORATORY“
- kann durch zahlreiche „Toolboxes“ erweitert werden

Institutsrechner

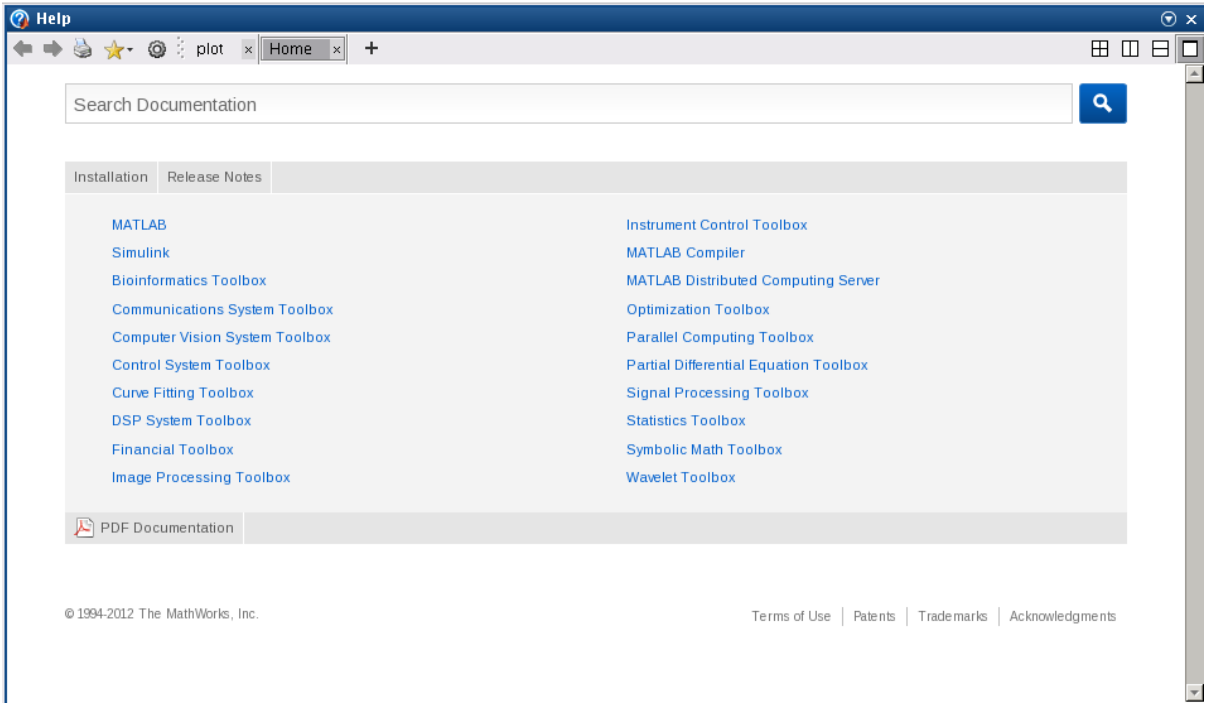
- installiert auf den Linux-Rechnern, auf den Windows-Rechnern und auf den Sun-Desktops
- je nach System sind verschiedene Versionen nutzbar
- Zentraler Server verwaltet Lizenzen
- ! X-Weiterleitung kann Probleme machen !
- >> **matlab-R2016a**

MATLAB R2016a



Hilfe

```
>> help
>> help plot
>> doc
>> doc plot
```

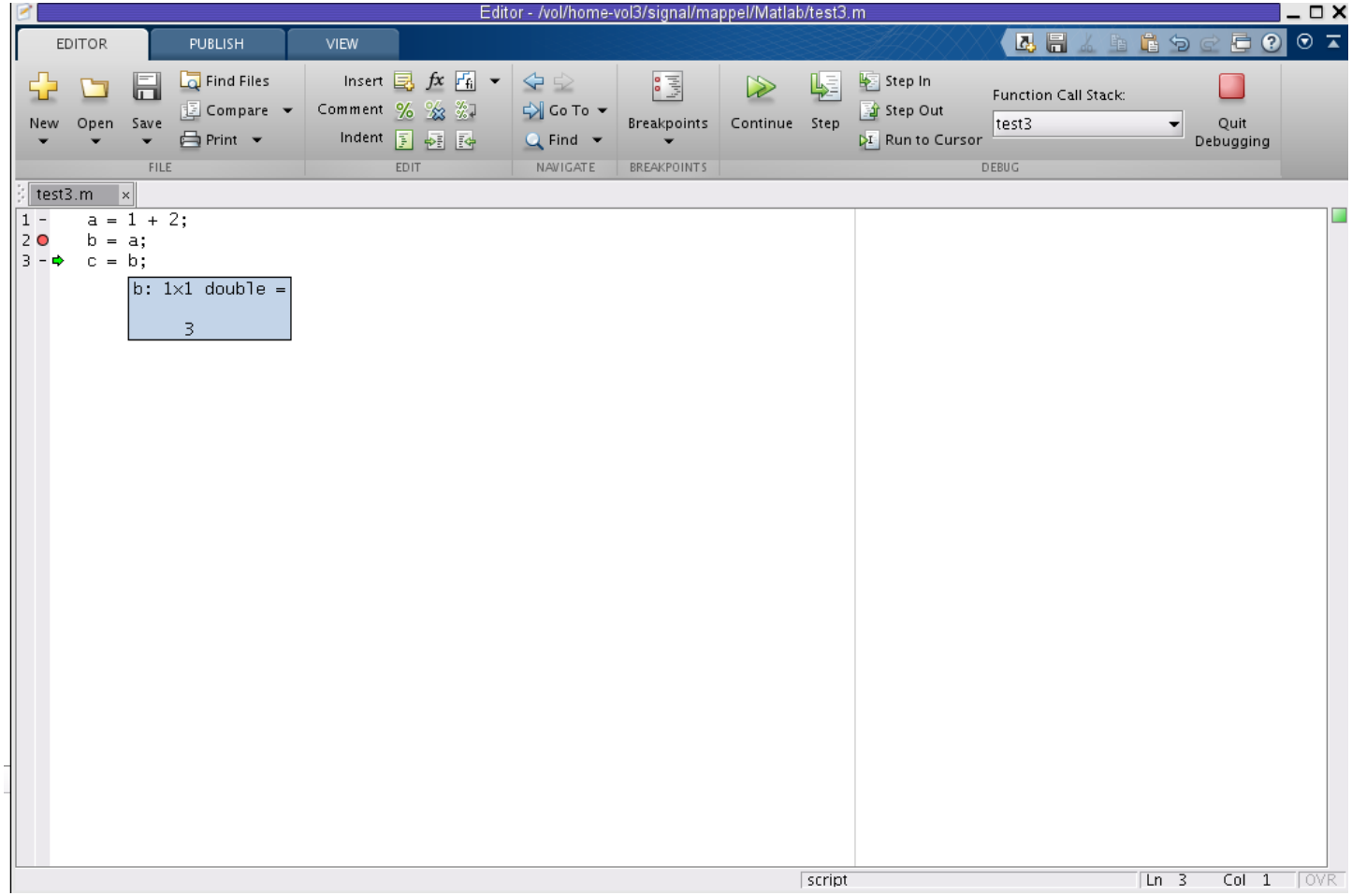


Code kommentieren

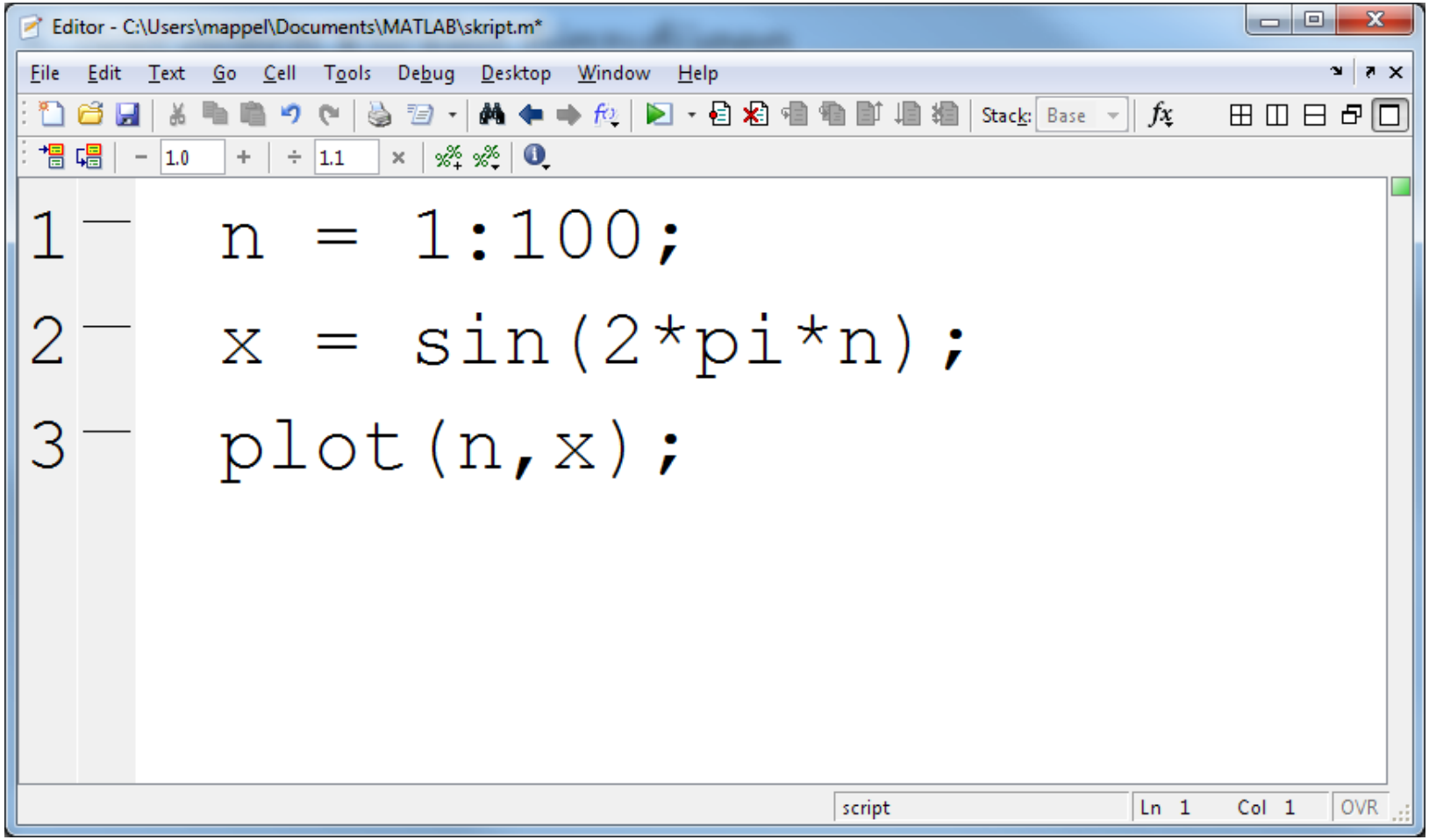
- Kommentar: %
- Besonderheit:

  Kommentar direkt nach Funktions-Definition erscheint bei Aufruf von *help functionname*

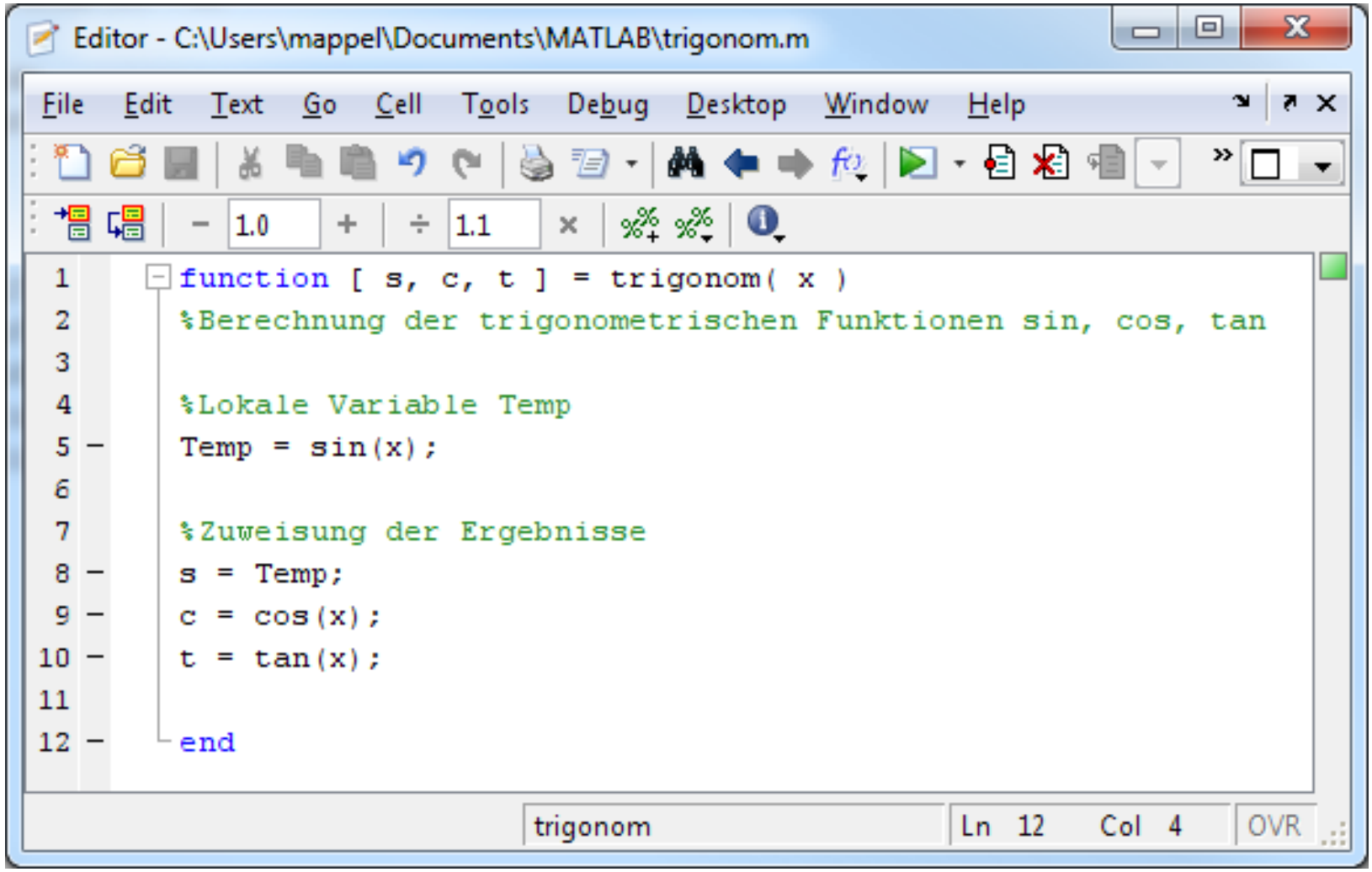
Editor


Editor - /vol/home-vol3/signal/mappel/Matlab/test3.m
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
FILE
Insert
Comment
Indent
EDIT
Go To
Find
NAVIGATE
Breakpoints
BREAKPOINTS
Continue
Step
Step In
Step Out
Run to Cursor
Function Call Stack:
test3
Quit Debugging
DEBUG
test3.m
1 - a = 1 + 2;
2 b = a;
3 - c = b;
b: 1x1 double =
3
script
Ln 3
Col 1
OVR

Skripte

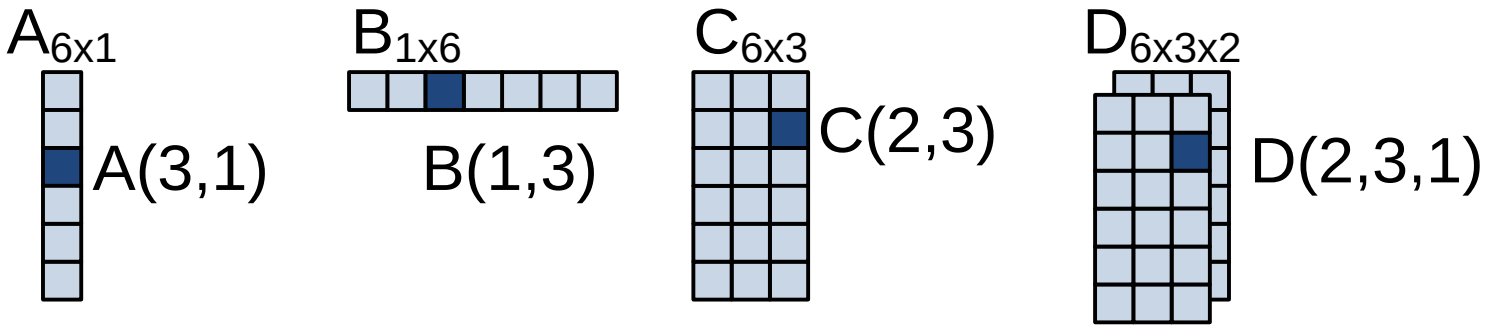

Editor - C:\Users\mappel\Documents\MATLAB\skript.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
1 — n = 1:100;
2 — x = sin(2*pi*n);
3 — plot(n,x);
script
Ln 1 Col 1
OVR

Funktionen


Editor - C:\Users\mappel\Documents\MATLAB\trigonom.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
1 function [s, c, t] = trigonom(x)
2 %Berechnung der trigonometrischen Funktionen sin, cos, tan
3
4 %Lokale Variable Temp
5 Temp = sin(x);
6
7 %Zuweisung der Ergebnisse
8 s = Temp;
9 c = cos(x);
10 t = tan(x);
11
12 end
trigonom
Ln 12 Col 4
OVR

Datentypen (1)

- Zentrale Datentypen:
  Skalare, Vektoren und Matrizen
  - indiziert über Zeilen und Spalten
  - Indizes starten immer mit 1



Datentypen (2)

- einfache Regeln für die Verwendung von Variablen
  - jede Variable ist eine Matrix
  - es gibt keine Variablendeklaration
  - Variablen werden durch Wertzuweisung dimensioniert
  - Unterscheidung von Groß- und Kleinschreibung
  - Namen von Variablen und Konstanten beginnen mit einem Buchstaben
  - Achtung: vordefinierte Konstanten, von besonderer Bedeutung sind die imaginäre Einheit „i“ bzw. „j“ und die Zahl „pi“
- es gibt auch komplexe Strukturen wie *struct* und *cell*

Schleifen

```
x=1;
for k=1:10
   x=x*k;
end
```

```
x=1;
k=1;
while k<10
   x=x*k;
   k=k+1;
end
```

Verzweigungen

```
if x>0
   y=x;
elseif x<0
   y=-x;
else
   y=0;
end
```

```
method='Bilinear';
switch lower(method)
   case{'linear','bilinear'}
      disp('Method islinear')
   case 'cubic'
      disp('Method is cubic')
   otherwise
      disp('Unknown method')
end
```

Semikolon

- schließt das Semikolon „;“ eine Kommandozeile ab, so wird die Anzeige des Ergebnisses unterdrückt

```
>> x = 5                    >> y = 5;
>> ans = 5                  >>
```

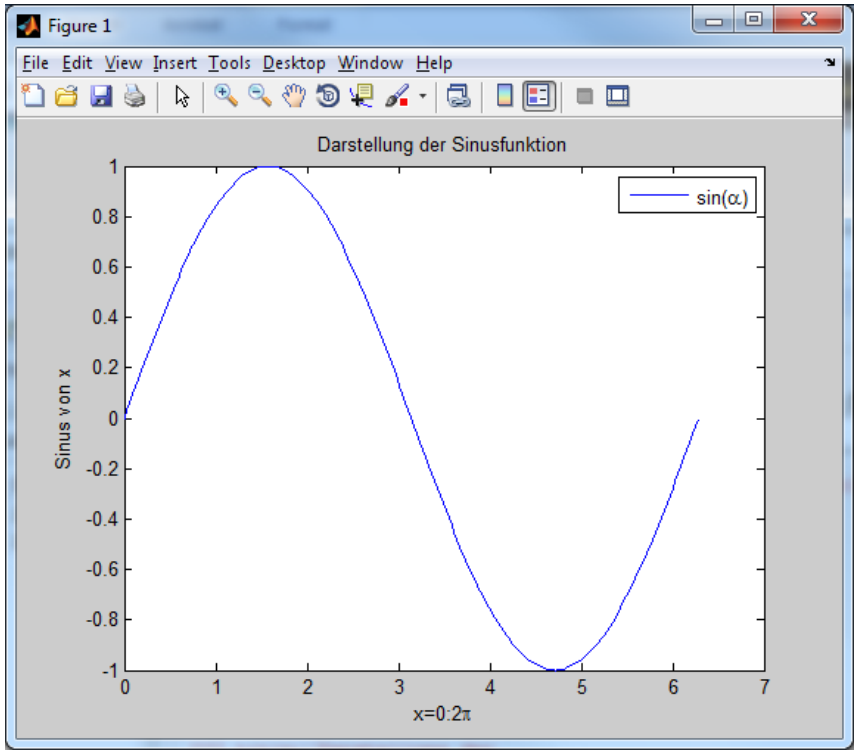
Doppelpunkt

- mit dem Doppel-Punkt-Operator lassen sich Datenfelder mit Elementen gleichen Abstands erzeugen

```
>> t = 1:10     =>  t = [1 2 3 ... 9 10]
>> t = 1:2:10   =>  t = [1 3 ... 9]
>> t = 0:2:10   =>  t = [0 2 ... 10]
```

Grafische Darstellung

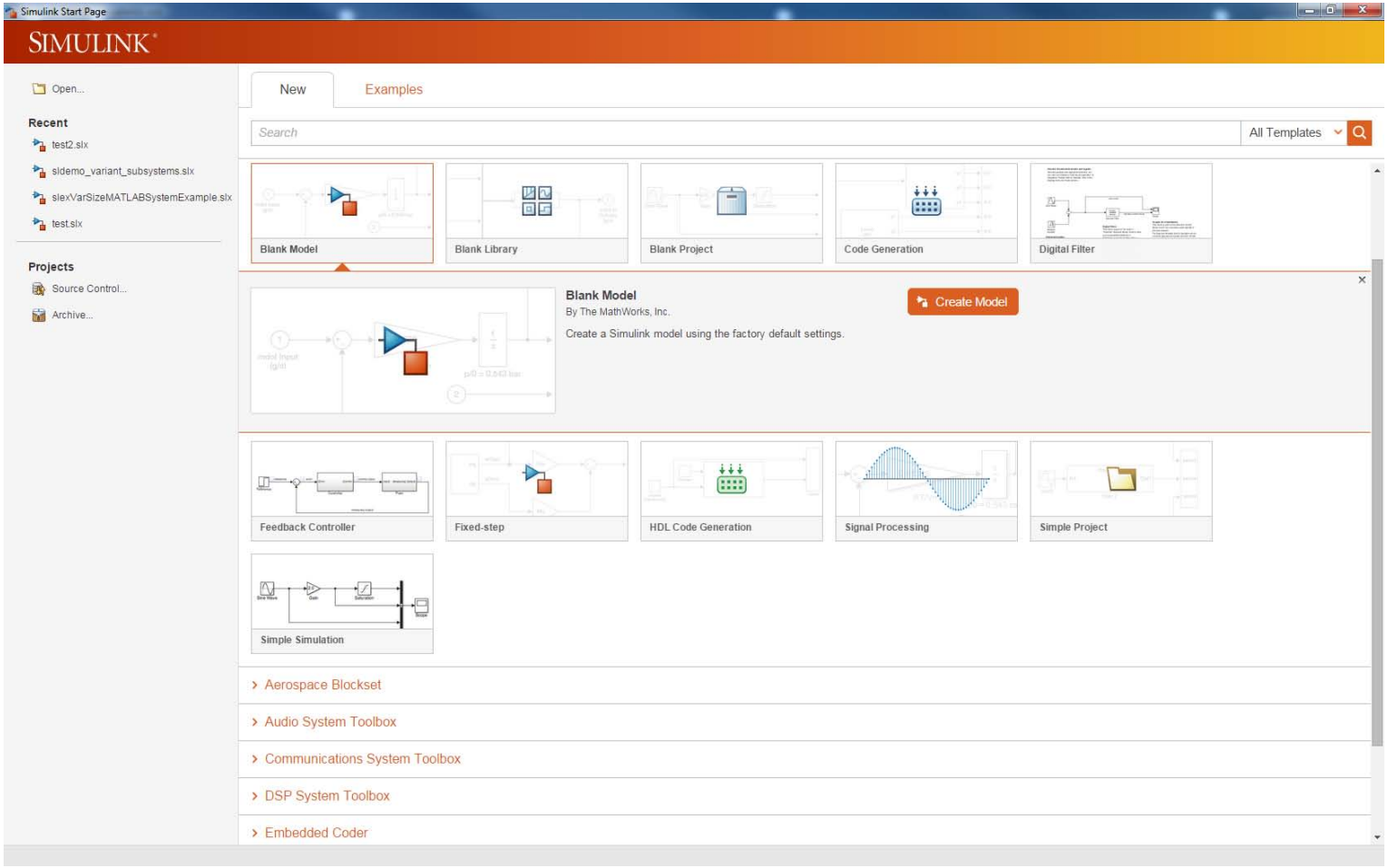
```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);
xlabel('x=0:2\pi');
ylabel('Sinus von x');
title('Darstellung der
    Sinusfunktion');
legend('sin(\alpha)');
```



Simulink

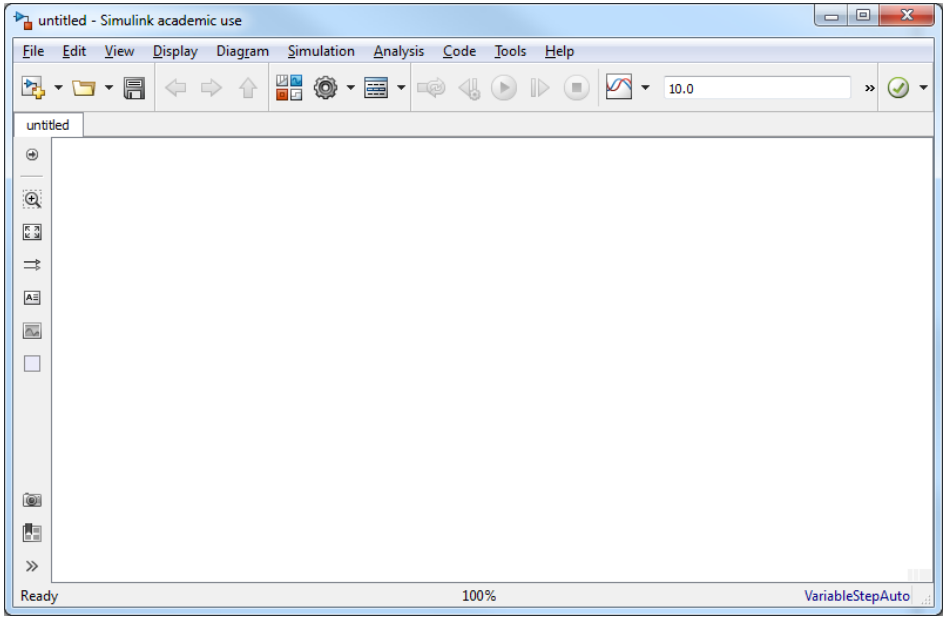
- Zusatzprodukt zu MATLAB
- Simulation von verschiedenen Systemen
- Blockbasierte Modellierung
- Datenfluss zwischen den Blöcken wird mit Verbindungslinien realisiert
- Kann durch Toolboxes erweitert werden
- >> simulink (im MATLAB Commando Fenster)

Simulink


Simulink Start Page
SIMULINK
Open...
Recent
test2.slx
sldemo_variant_subsystems.slx
slexVarSizeMATLABSystemExample.slx
test.slx
Projects
Source Control...
Archive...
New
Examples
Search
All Templates
Blank Model
Blank Library
Blank Project
Code Generation
Digital Filter
Blank Model
By The MathWorks, Inc.
Create a Simulink model using the factory default settings.
Create Model
Feedback Controller
Fixed-step
HDL Code Generation
Signal Processing
Simple Project
Simple Simulation
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
DSP System Toolbox
Embedded Coder

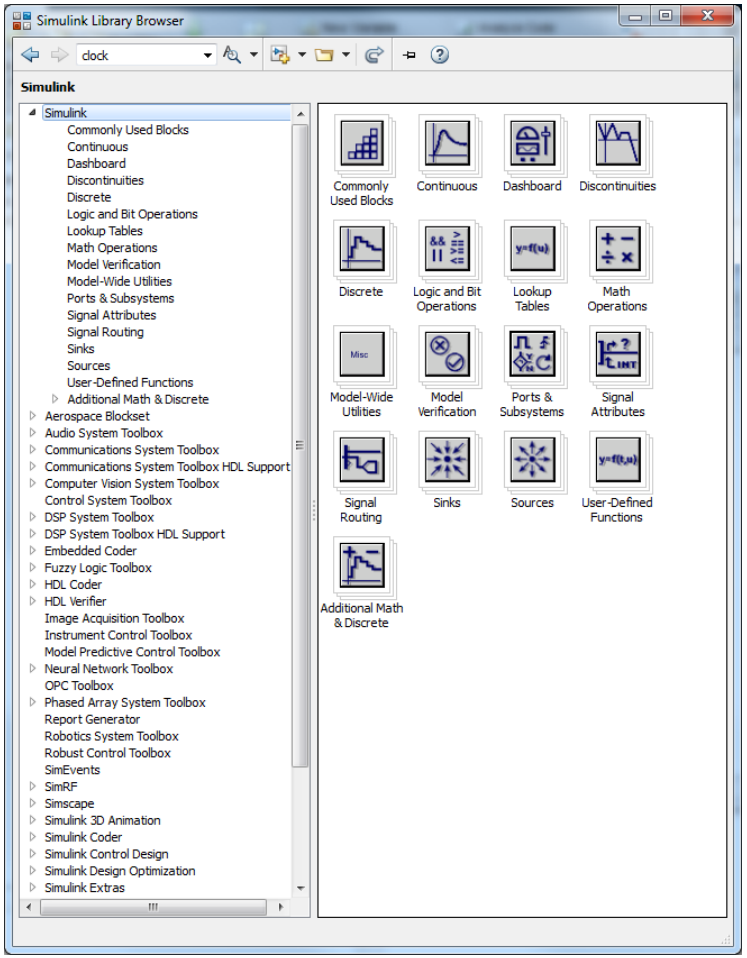
Simulink – Erste Schritte

- Neues Projekt anlegen
  - Blank Project
- **Neues Modell anlegen**
  - **Blank Model → Create Model**



Simulink – Library Browser

- Library Browser enthält die vorgefertigten Blöcke



Simulink – Beispiel Chirp Signal (1)

- Beispielaufgabe: Linearer Chirp

  $y(t) = \sin(2\pi f_0 t + \pi k t^2)$

  $k = (f_1 - f_0) / T$

  mit

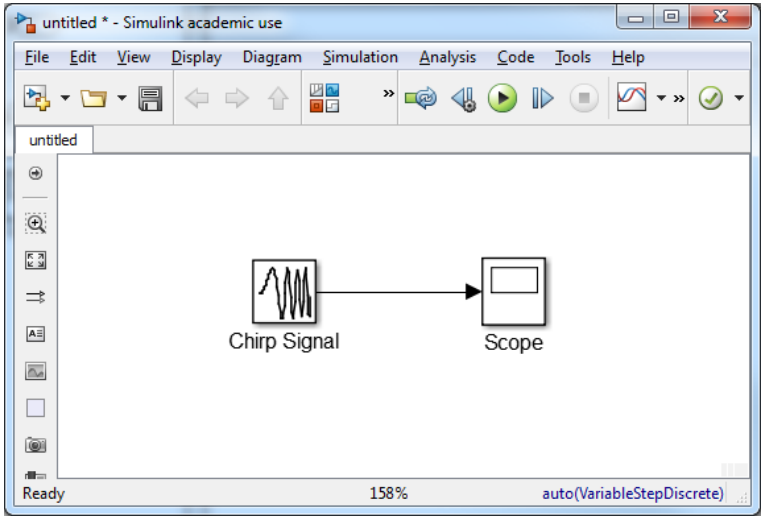
  Startfrequenz f_0

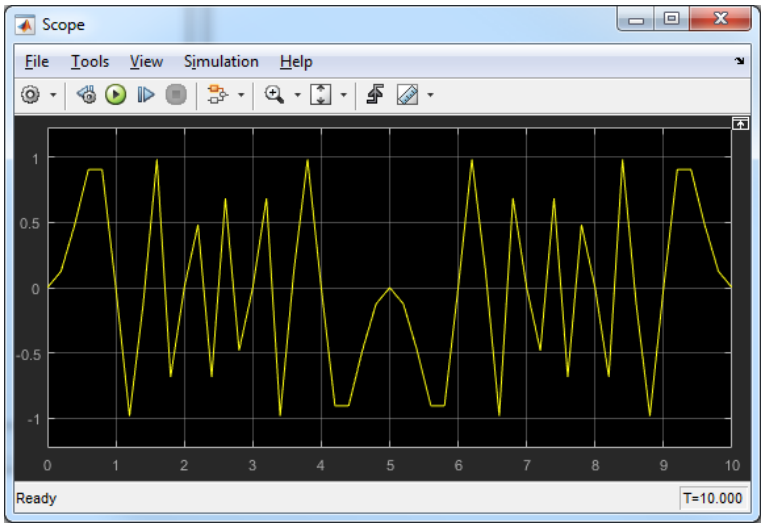
  Endfrequenz f_1

  Dauer T

Simulink – Beispiel Chirp Signal (2)

- **Simulink→Sources → Chirp Signal** auswählen und in das Model ziehen
- Startfrequenz = 0, Endfrequenz = 10, Dauer = 10 einstellen durch Doppelklick auf Chirp Signal Block
- **Simulink→Sinks → Scope** auswählen und in das Modell ziehen
- Quelle und Senke verbinden
- Doppelklick auf Scope
- Simulation starten





Simulink - Simulation

- Simulationsresultat sieht recht merkwürdig aus → zu wenige Punkte, um Chirp korrekt darzustellen
- **Simulation → Model Configuration Parameters** öffnen und Max step size auf 0.01 setzen
- Chirp Signal sieht nun deutlich besser aus

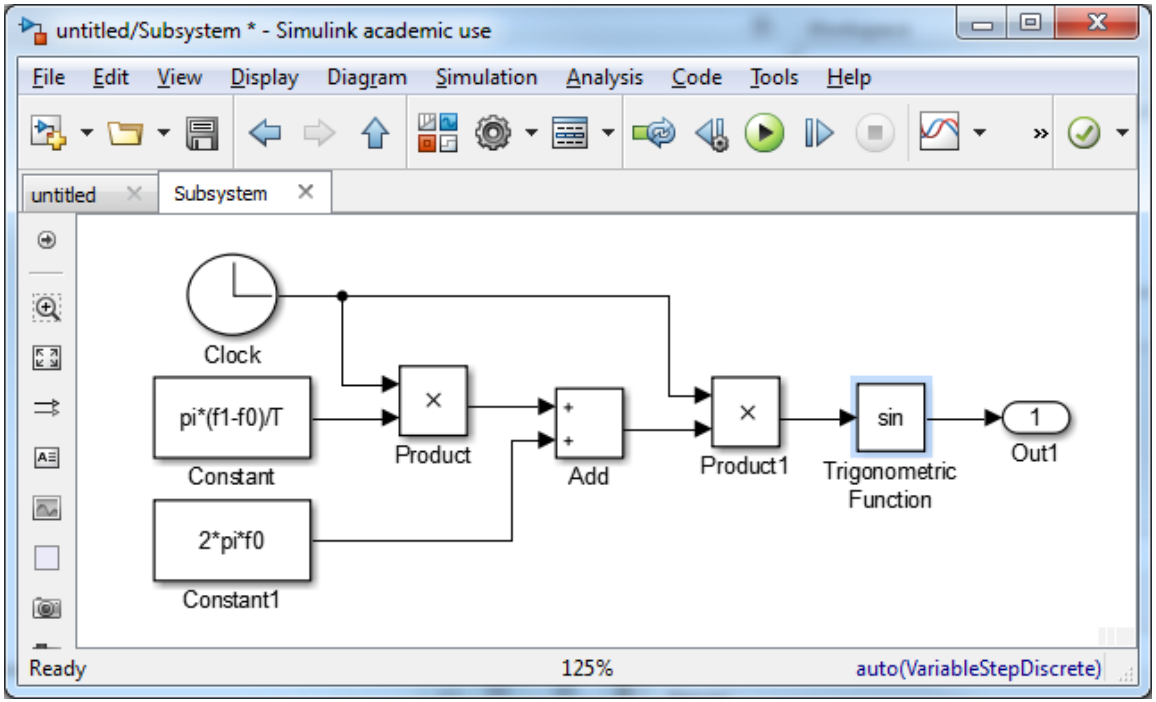
Simulink - Solver

- Solver Optionen sind in Simulink wichtige Parameter
- Simulation mit variablen oder festen Abständen
- Solver sind entweder diskrete Lösungsverfahren oder verschiedene Verfahren für differential Gleichungen

Simulink - Beispiel Chirp Signal (3)

- Aufbau des Chirp Signal Generators aus diskreten Blöcken
- Hierachie nutzen, in dem aus **Simulink → Ports & Subsystems → Subsystem** Block hinzugefügt wird, welcher per Doppelklick geöffnet wird

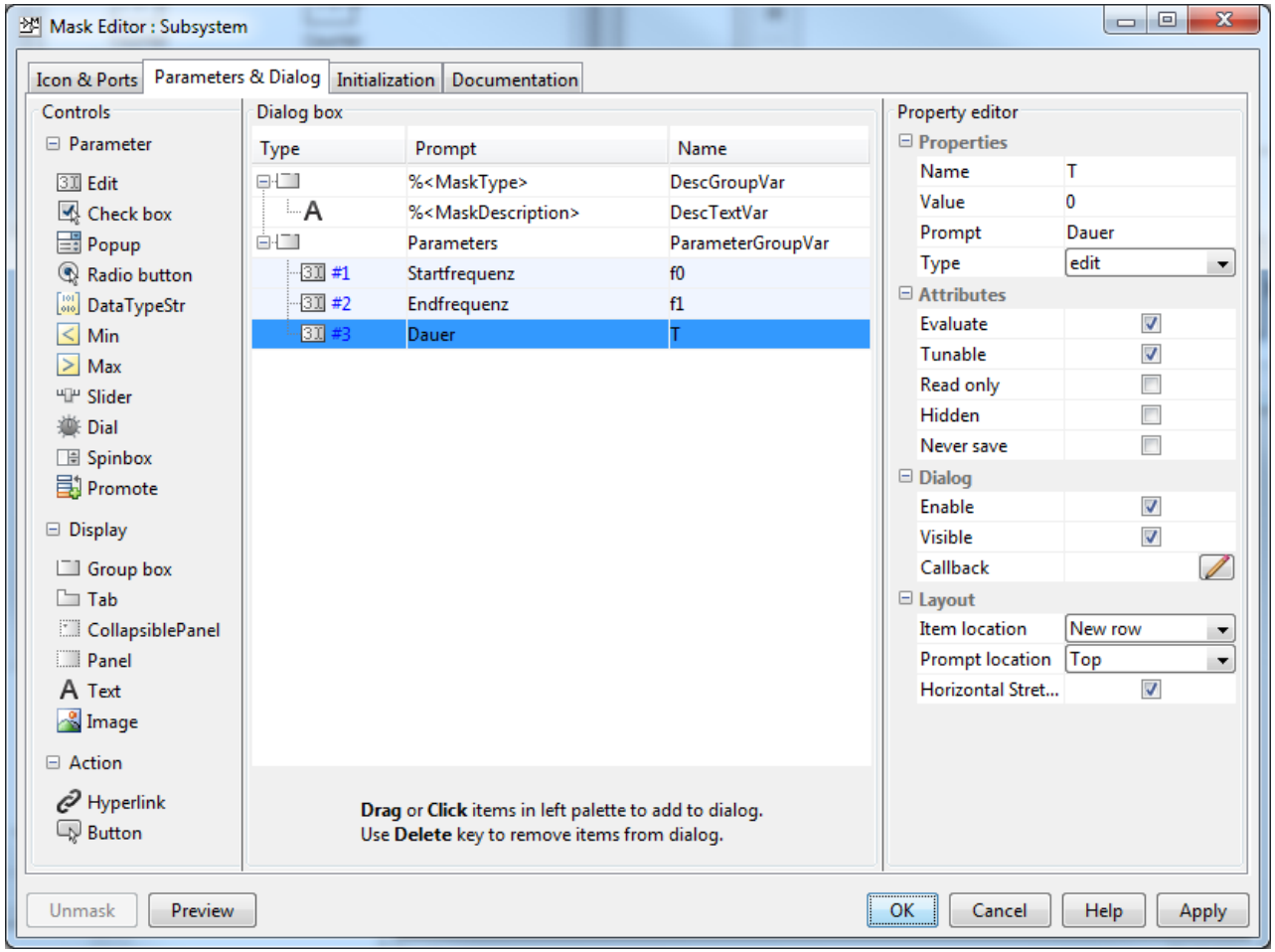
Simulink - Beispiel Chirp Signal (4)



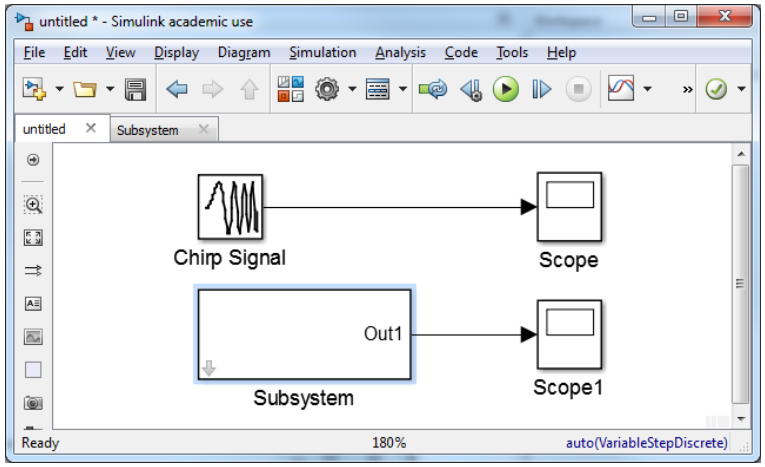
Simulink - Beispiel Chirp Signal (5)

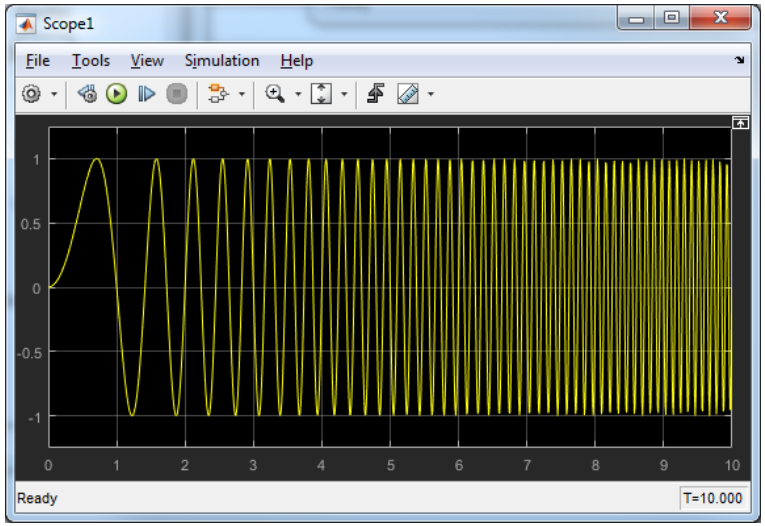
- Modell enthält
  - den Takt für die jeweilige Werte der Simulation
  - Startfrequenz f_0, Endfrequenz f_1, Dauer T als Variablen
- Variablen müssen noch einstellbar gemacht werden: **Diagram → Mask → Create Mask**
  - Unter **Parameters** müssen dann noch die fehlenden Variablen f_0, f_1 und T eingetragen werden, so dass diese beim Klicken auf das Subsystem-Symbol eingestellt werden können

Simulink - Beispiel Chirp Signal (6)



Simulink - Beispiel Chirp Signal (7)

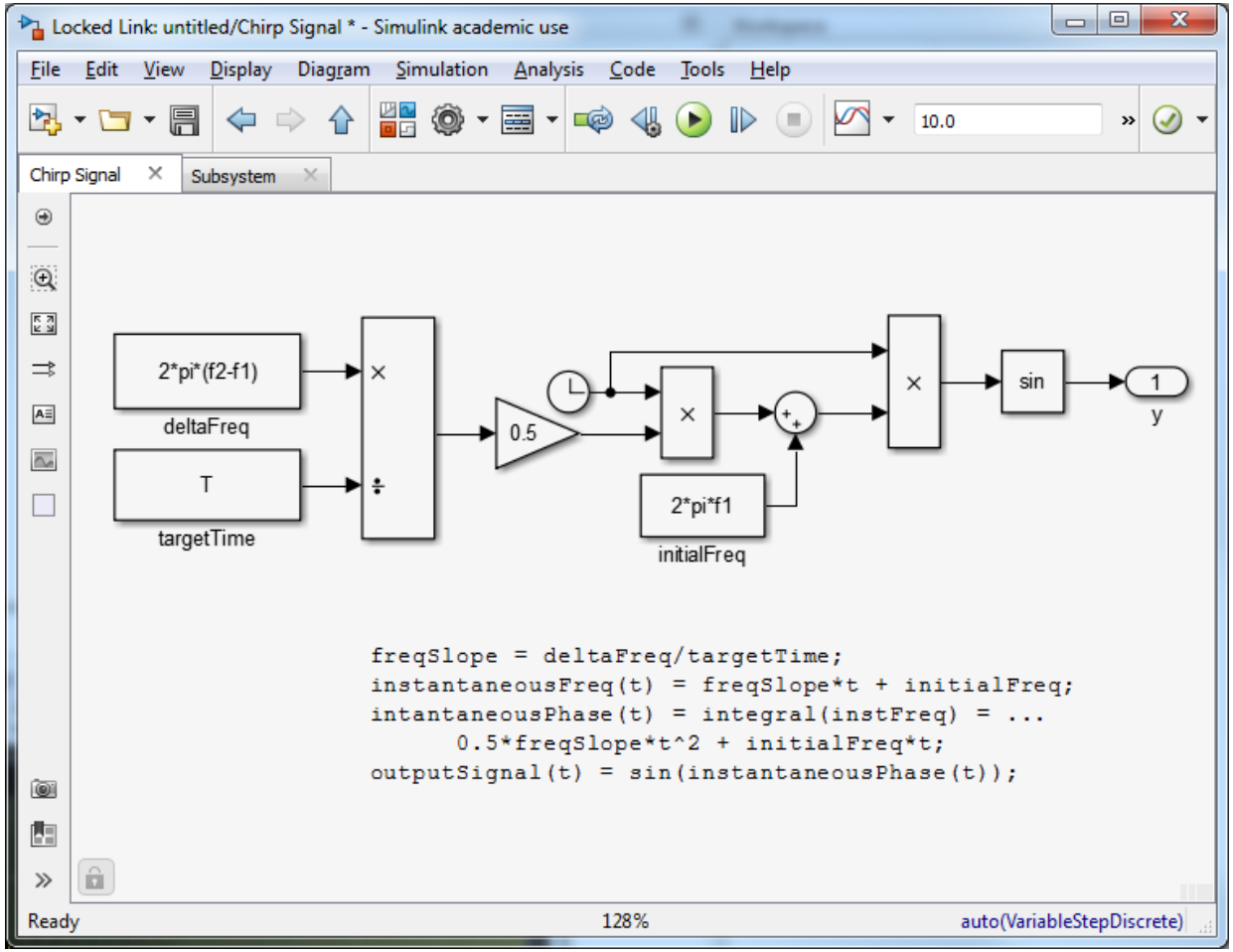




Simulink – Fertige Blöcke anschauen

- **Diagram → Mask → View Mask**
  - Masken von fertigen Blöcken anschauen, wenn man eigene Blöcke baut
- **Diagram → Mask → Look Under Mask**
  - Liefert Einblick in das Innenleben der Blöcke

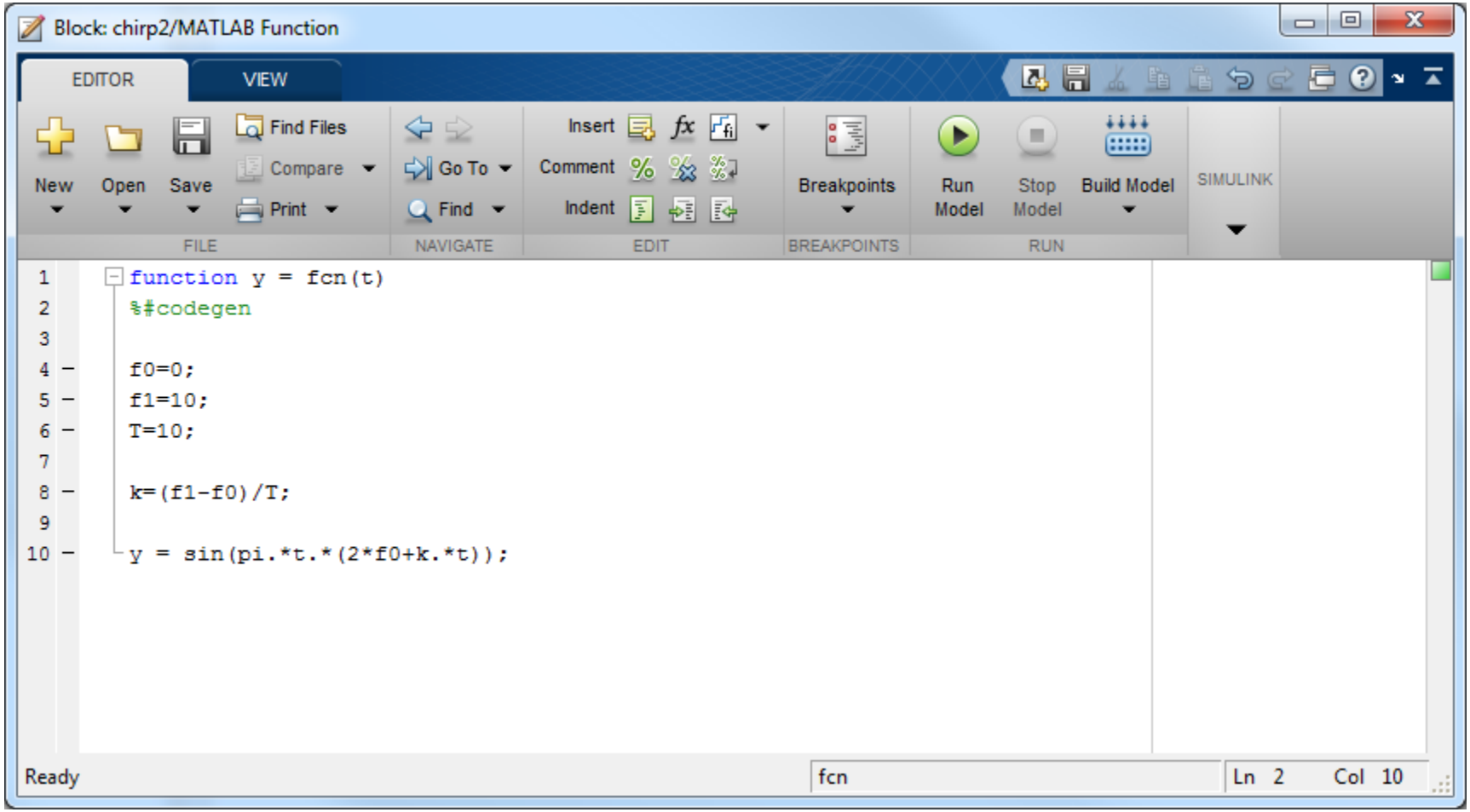
Simulink – Fertige Blöcke anschauen



Simulink - Beispiel Chirp Signal (8)

- Aufbau des Chirp Signal Generators mit Hilfe von MATLAB Code
- **Simulink → User-Defined Functions → MATLAB Function**
- Alternative: **S-Function**

Simulink - Beispiel Chirp Signal (9)



Simulink - Beispiel Chirp Signal (10)

- Takt Signal muss als Funktions-Parameter vorhanden sein, damit zu den einzelnen Simulations-Schritten jeweils ein neuer Wert berechnet werden kann

Simulink – Scope Block

- dient zum Anzeigen der Signale
- Unter File → Number of Input Ports kann festgelegt werden, wie viele Signale in einem Scope angezeigt werden
- Unter View → Layout kann festgelegt werden, ob alle Signale in einer Zeichenfläche dargestellt werden oder ob verschiedene Zeichenflächen zur Verfügung stehen

Simulink - Beispiel Chirp Signal (11)