

# Einführung in die Programmierung (MA8003)

## Theorie 3.1: Visualisierung von Ergebnissen

Dr. Laura Scarabosio

Technische Universität München  
Fakultät Mathematik, Lehrstuhl für Numerische Mathematik M2

10.10.2019

<b>Theorie 1.1+1.2</b>	Mo (07.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 1.1	Mo (07.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 1.2	Mo (07.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
<b>Theorie 2.1+2.2</b>	Mi (09.10.2019)	13:00 - 14:30 Uhr	HS BC2 0.01.17
Praxis 2.1	Mi (09.10.2019)	15:00 - 16:30 Uhr	HS BC2 0.01.17/16
Praxis 2.2	Mi (09.10.2019)	16:30 - 18:00 Uhr	HS BC2 0.01.17/16
<b>Theorie 3.1+3.2</b>	Do (10.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 3.1	Do (10.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 3.2	Do (10.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
<b>Theorie 4.1+4.2</b>	Fr (11.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 4.1	Fr (11.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 4.2	Fr (11.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
<b>Klausur</b>	Fr (25.10.2019)	xxx Uhr	xxx
Nachholklausur	Fr (22.11.2019)	xxx Uhr	xxx

Kurswebseite mit Infos, Folien und Übungsblättern:

<https://www-m2.ma.tum.de/bin/view/Allgemeines/...>  
...MA8003WS19

**Bitte melden Sie sich über TUM-Online für die Klausur an!**

## 1 2D-Plots

- Plotten von Funktionen mit einer Veränderlichen
- Konfigurieren der Plots

## 2 3D-Plots

- Kurven in 3D
- Plots bivariater Funktionen in 3D

## 3 Plots erstellen mit Hilfe der GUI

## 1 2D-Plots

- Plotten von Funktionen mit einer Veränderlichen
- Konfigurieren der Plots

## 2 3D-Plots

- Kurven in 3D
- Plots bivariater Funktionen in 3D

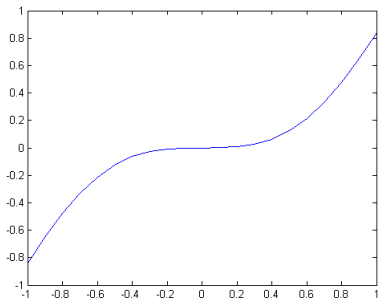
## 3 Plots erstellen mit Hilfe der GUI

Verwende `plot` zum Plotten von Punkten im  $\mathbb{R}^2$  und Kurven  $\omega: \mathbb{R} \rightarrow \mathbb{R}^2$ .

## Syntax

- `plot(x, y)` mit `x` und `y` Vektoren gleicher Länge:  
Plottet die Punkte  $(x(i), y(i))$  und verbindet sie mit Strecken.
- `plot(X, Y)` mit `X` und `Y` Matrizen gleicher Dimension:  
Plottet jeweils die **Spalten** von `X` und `Y` in verschiedenen Farben, also jeweils für ein `j` die Punkte  $(X(i,j), Y(i,j))$  für alle `i`.
- `plot(x1, y1, x2, y2, ...)` == `plot([x1, x2, ...], [y1, y2, ...])` falls `xi` und `yi` Spaltenvektoren sind.
- `plot(x, Y)` == `plot([x, x, x, ...], Y)`, wenn `x` Spaltenvektor und `Y` Matrix ist.

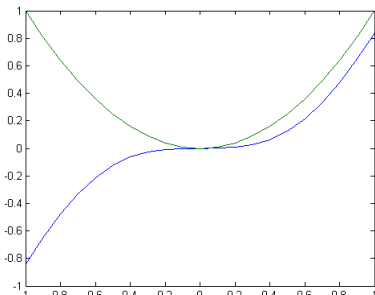
```
>> x = -1:0.1:1;  
>> y = x.*sin(x.^2);  
>> plot(x,y)
```



```
>> y2 = x.^2;  
>> plot(x, y, x, y2)
```

```
>> y2 = x.^2;  
>> plot([x', x'], [y', y2'])
```

```
>> y2 = x.^2;  
>> plot(x, [y', y2'])
```



Das Aussehen der Plots kann durch mannigfaltige Optionen verändert werden.

Farbe	
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Marker	
o	Kreis
*	Stern
.	Punkt
+	Plus
x	Kreuz
s	Quadrat
d	Diamant
^	Dreieck nach oben
v	Dreieck nach unten
>	Dreieck nach rechts
<	Dreieck nach links

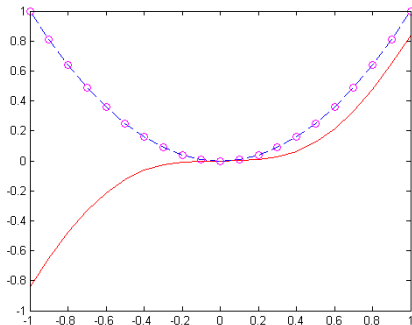
Linienart	
-	durchgezogene Linie
--	gestrichelte Linie
:	gepunktete Linie
-.	gepunktet und gestrichelt

Die Parameter werden als String hinter die zu plottenden Daten geschrieben, z. B. `plot(x,y,'r',x2,y2,'-b')`.

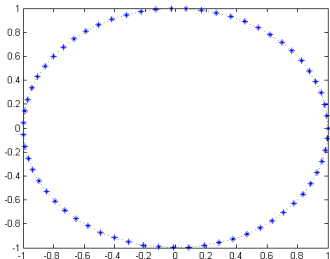
Standard sind die Optionen `plot(...,'-', 'color', [0 0.45 0.74])`.

# Plotten IV

```
>> plot(x,y, 'r', x,y2,'--b', x,y2, 'om')
```



```
>> t=0:0.1:2*pi;  
>> x=cos(t);  
>> y=sin(t);  
>> plot(x,y, '*', x,y, ':')  
>> axis equal
```





Um die Skalierung der Achsen manuell anzupassen verwende die Befehle `axis`, `xlim` oder `ylim`.

## Syntax

- `axis([xmin, xmax, ymin, ymax])` skaliert den Bereich, so dass  $[xmin, ymin] \times [xmax, ymax]$  sichtbar ist.
- `ylim([ymin ymax])` bzw. `xlim([xmin xmax])` skaliert nur die x- bzw. y-Achse.

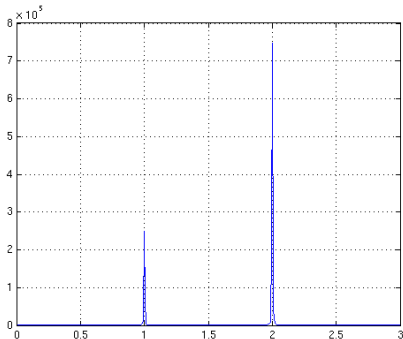
Hinweis: Für 3-D Plots gibt es auch `zlim`.

# Plotten VI

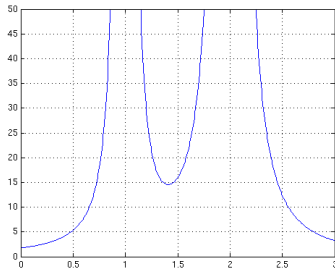
Wir betrachten die Funktion

$$x \mapsto \frac{1}{(x-1)^2} + \frac{3}{(x-2)^2}.$$

```
>> x = linspace(0,3,500);  
>> plot(x, 1./(x-1).^2 + 3./(x-2).^2)  
>> grid on
```

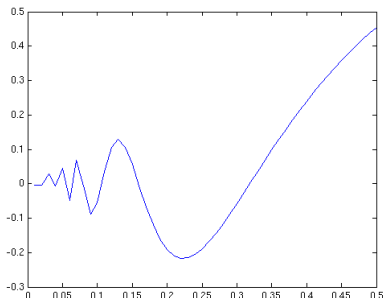


```
>> ylim([0 50])
```

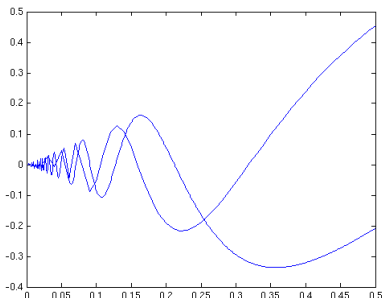


Normalerweise überschreibt Matlab das aktuelle Fenster bei einem neuen Plot-Befehl. Um einen neuen Plot in dem aktuellen zusätzlich zu erzeugen, verwende den Befehl `hold on`.

```
>> x = 0:0.01:0.5;  
>> plot(x, x.*sin(1./x))
```



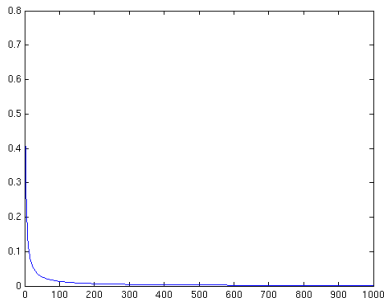
```
>> x = 0:0.001:0.5;  
>> hold on;  
>> plot(x, x.*cos(1./x))
```



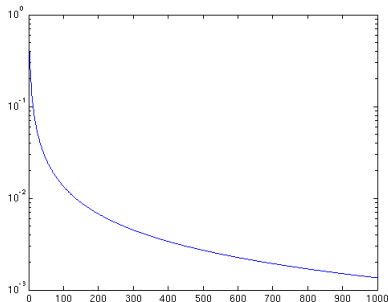
Verwende `figure` um ein neues Plot-Fenster zu öffnen.

Mit `semilogx`, `semilogy` und `loglog` werden Plots mit logarithmischer Achsenskalierung erstellt. Diese werden häufig verwendet um Konvergenzgeschwindigkeiten von Algorithmen besser abzulesen zu können.

```
>> n = 1:1000;  
>> y = exp(1) - (1+1./n).^n;  
>> plot(n,y)
```



```
>> n = 1:1000;  
>> y = exp(1) - (1+1./n).^n;  
>> semilogy(n,y)
```

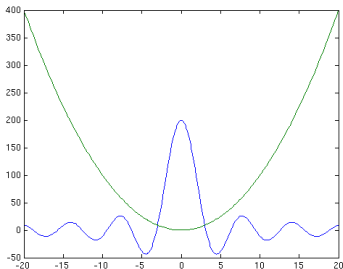


Mit `fplot` können schnell algebraische Funktionen geplottet werden. Der User muss sich im Gegensatz zum `plot`-Befehl nicht um die Diskretisierung kümmern.

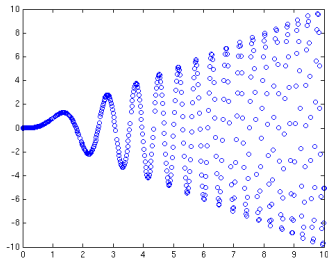
## Syntax

`fplot(f, [xmin, xmax])` mit `f` Funktionshandle oder Funktionsstring:  
Plottet die Funktion `f` im Intervall `[xmin, xmax]`.

```
>> f = @(x) [200*sin(x(:))./x(:), ...  
x(:).^2];  
>> fplot(f, [-20, 20])
```



```
>> fplot('w.*sin(w.^2)', ...  
[0, 10], 'o');
```



Mit dem Befehl `subplot` können mehrere Plots in ein Fenster in verschiedene Koordinatensysteme gezeichnet werden.

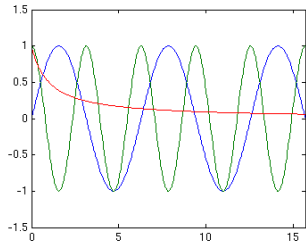
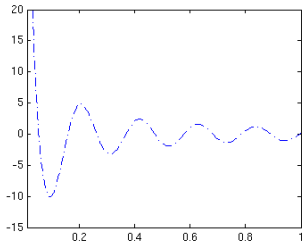
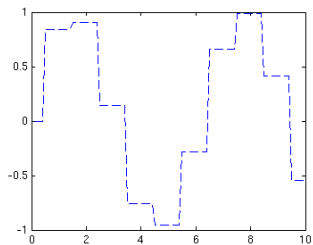
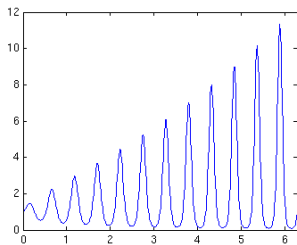
## Syntax

`subplot(n,m,p)`: Erzeugt beim ersten Aufruf ein Fenster mit  $n \cdot m$  Koordinatensystemen in  $n$  Zeilen und  $m$  Spalten.

Setzt das  $p$ -te Koordinatensystem aktiv. Der nächste `plot`-Befehl zeichnet in dieses Koordinatensystem.

```
>> subplot(2,2,1), fplot('exp(sqrt(x)*sin(12*x))',[0 2*pi])
>> subplot(2,2,2), plot(0:0.1:10, sin(round(0:0.1:10)), '--')
>> subplot(2,2,3), fplot('cos(30*x)/x',[0.01 1 -15 20], '-. ')
>> subplot(2,2,4), fplot('[sin(x), cos(2*x), 1/(1+x)]',[0 5*pi -1.5 1.5])
```

# subplot II



## 1 2D-Plots

- Plotten von Funktionen mit einer Veränderlichen
- Konfigurieren der Plots

## 2 3D-Plots

- Kurven in 3D
- Plots bivariater Funktionen in 3D

## 3 Plots erstellen mit Hilfe der GUI

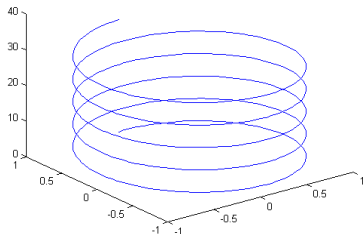


Mit dem Befehl `plot3` können Kurven  $\omega: \mathbb{R} \rightarrow \mathbb{R}^3$  geplottet werden.

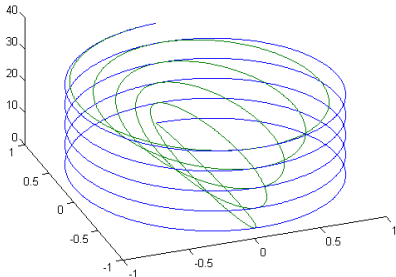
### Syntax

- `plot3(x,y,z)` mit Vektoren `x`, `y`, `z`: Plottet die Punkte  $(x(i), y(i), z(i))$  für alle  $i \in 1, \dots, \text{length}(x)$  und verbindet sie gegebenenfalls.  
**Wichtig:** `length(x) == length(y) == length(z)`.
- `plot3(X,Y,Z)` mit Matrizen `X`, `Y`, `Z`: Wie `plot3(X(:,i), Y(:,i), Z(:,i))` für alle **Spalten** `i` der Matrizen in verschiedenen Farben.

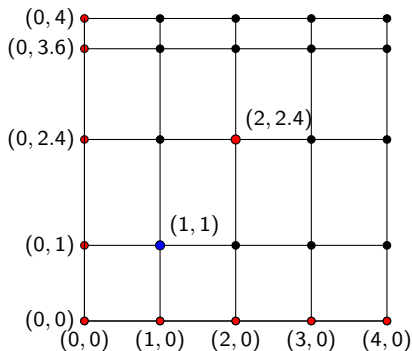
```
>> t = 0:0.01:10*pi;
>> plot3(sin(t),cos(t),t)
```



```
>> t = (0:0.01:10*pi)';
>> plot3([sin(t),t./(10*pi).*sin(t)],...
[cos(t), cos(t)], [t,t])
```



Um Funktionen  $f: \mathbb{R}^2 \rightarrow \mathbb{R}$  zu plotten, müssen wir den zu plottenden Bereich erst diskretisieren.



```
>> x = 0:1:4;
>> y = [4, 3.6, 2.4, 1, 0];
```

```
>> [X, Y] = meshgrid(x,y)
```

X =

0	1	2	3	4
0	1	2	3	4
0	1	2	3	4
0	1	2	3	4
0	1	2	3	4

Y =

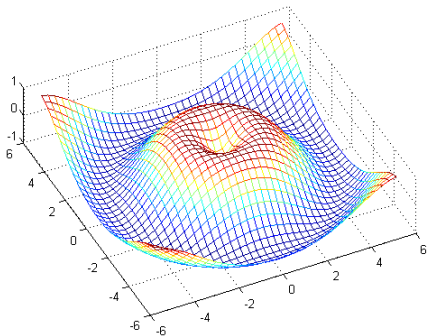
4.0000	4.0000	4.0000	.
3.6000	3.6000	3.6000	.
2.4000	2.4000	2.4000	.
1.0000	1.0000	1.0000	.
0	0	0	.

Haben wir alle Punkte mit `meshgrid` erzeugt, kann die Funktion anschließend  $f: \mathbb{R}^2 \rightarrow \mathbb{R}$  an jedem Gitterpunkt ausgewertet werden.

```
>> x = -5.5:0.3:5.5; y = -5.5:0.3:5.5;  
>> [X, Y] = meshgrid(x,y);  
>> f = @(x,y) sin(sqrt(x.^2+y.^2));  
>> Z = f(X,Y);
```

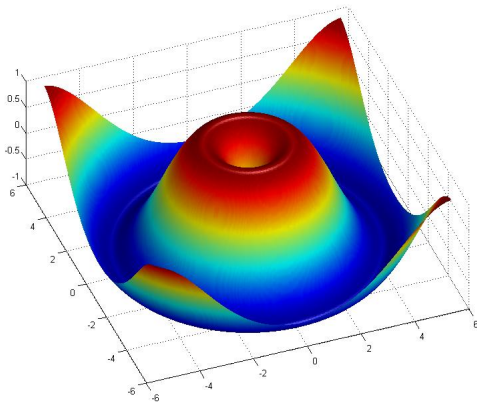
Die Funktion kann dann z. B. mittels des `mesh` Befehls als *Funktionengebirge* geplottet werden:

```
>> mesh(X,Y,Z);
```



Neben `mesh` gibt es noch viele weitere Befehle die ein ähnliches Funktionengebirge liefern, z.B. `surf`, `waterfall`, ... Auch Beleuchtung und Schattenwurf kann eingestellt werden, um die Oberflächen realistischer erscheinen zu lassen:

```
>> x = -5.5:0.05:5.5;  
>> y = -5.5:0.05:5.5;  
>> [X, Y] = meshgrid(x,y);  
>> Z = f(X,Y);  
>> surf1(X,Y,Z, 'light')  
>> shading interp
```



## 1 2D-Plots

- Plotten von Funktionen mit einer Veränderlichen
- Konfigurieren der Plots

## 2 3D-Plots

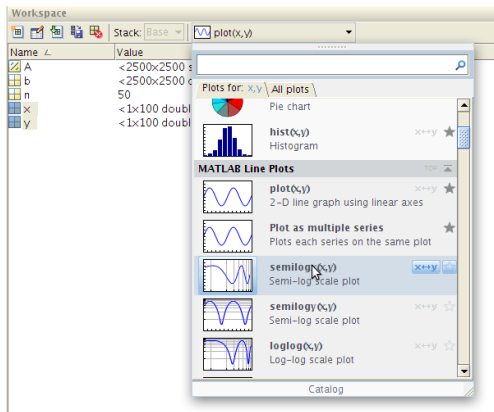
- Kurven in 3D
- Plots bivariater Funktionen in 3D

## 3 Plots erstellen mit Hilfe der GUI

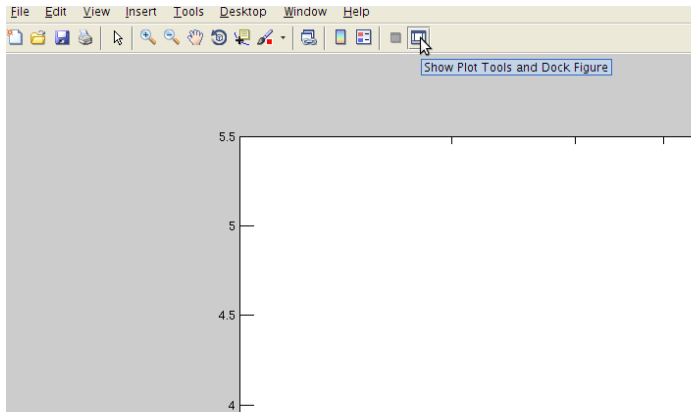
# Plotten mit der GUI - 1

Plots können nicht nur über Befehle, sondern auch über die **Matlab-Oberfläche** verändert werden.

Variablen für den Plot im Workspace markieren und Plot aus Katalog auswählen:



Öffnen des Editors um erzeugten Plot nachzubearbeiten:





## Anpassen der Formattierung und Code-Generierung:

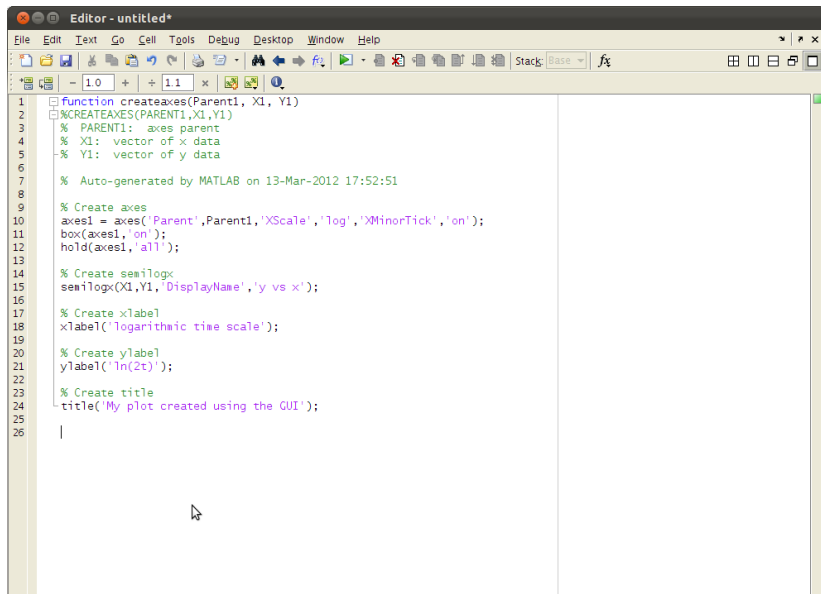
The screenshot shows a software interface for plotting. The main window displays a plot titled "My plot created using the GUI". The plot shows a blue line representing the function  $y = x^2$  on a coordinate system where the x-axis is labeled "logarithmic line scale" and the y-axis is labeled "MyZ". The plot area has a context menu open with the following options: Add Data..., Cut, Copy, Paste, Clear Axes, Delete, Show Legend, Color..., Font..., Grid, Show Property Editor, and Show Logics. The "Show Logics" option is currently selected.

On the left side, there is a "New Subplots" panel with options for 2D Axes and 3D Axes. Below that is a "Variables" list containing: A (250), B (250...), n (1x1), h (1x100), and y (1x100). Underneath is an "Annotations" section with icons for Line, Arrow, Double Arrow, Text Arrow, Text Box, Rectangle, and Ellipse.

On the right side, there is a panel titled "My plot created using the GUI" with a checked checkbox and a legend entry "y vs x".

At the bottom, there are two property editors. The left one is titled "Title: My plot created using the GUI" and has a "Colors" section with a color picker and a "Grid" section with checkboxes for X, Y, Z, and a checked "Box" option. The right one is titled "X Axis | Y Axis | Z Axis | Font |" and has a "Y Label" field containing "MyZ", a "Y Limits" field with values "5" and "10", a "Y Scale" dropdown set to "Linear", and a checked "Reverse" checkbox. There is also a "Ticks" button and a "More Properties..." link.

Verwenden des generierten Codes ohne GUI und Mausclicks:



```
1 function createaxes(Parent1, X1, Y1)
2 %CREATEAXES(PARENT1,X1,Y1)
3 % PARENT1: axes parent
4 % X1: vector of x data
5 % Y1: vector of y data
6
7 % Auto-generated by MATLAB on 13-Mar-2012 17:52:51
8
9 % Create axes
10 axes1 = axes('Parent',Parent1,'XScale','log','XMinorTick','on');
11 box(axes1,'on');
12 hold(axes1,'all');
13
14 % Create semilogx
15 semilogx(X1,Y1,'DisplayName','y vs x');
16
17 % Create xlabel
18 xlabel('logarithmic time scale');
19
20 % Create ylabel
21 ylabel('ln(2t)');
22
23 % Create title
24 title('My plot created using the GUI');
25
26 |
```

Ende Theorie 3.1

**Fragen?**