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

Ablauf

Theorie 1.1+1.2	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
Theorie 2.1+2.2	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
Theorie 3.1+3.2	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
Theorie 4.1+4.2	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
Klausur	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!

Theorie 3.1: Inhalt

Inhalt

Plotten I

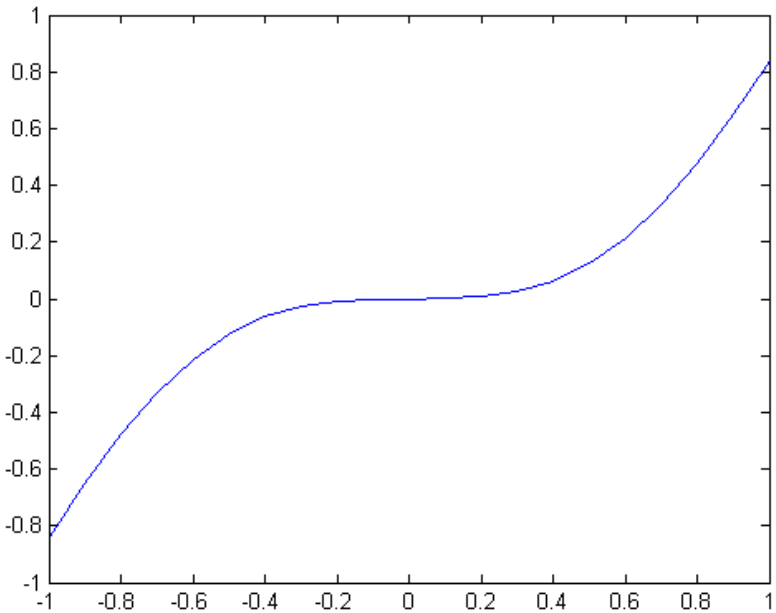
Verwende `plot` zum Plotten von Punkten im $\mathbb{R}^2$ und Kurven $\omega\colon \mathbb{R} \to \mathbb{R}^2$.

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.

Plotten II

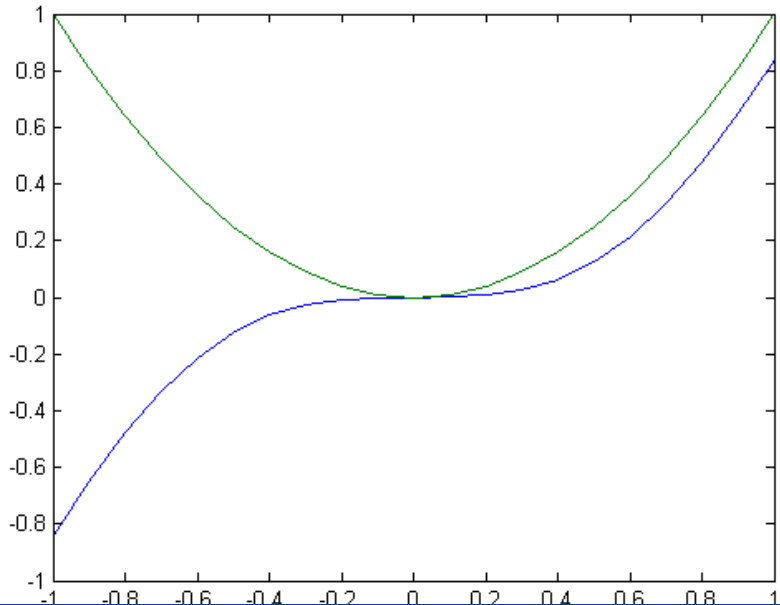
```
>> x = -1:0.1:1;
>> y = x.*sin(x.^2);
>> plot(x,y)
```




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

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

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



Plotten III

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

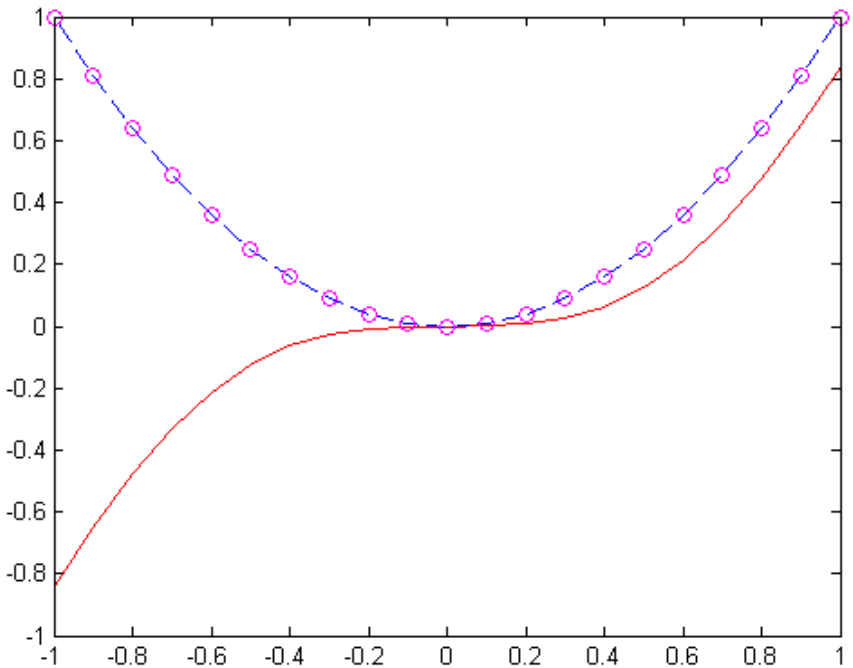
Marker	
o	Kreis
*	Stern
.	Punkt
+	Plus
x	Kreuz
s	Quadrat
d	Diamant
^	Dreieck nach oben
v	Dreieck nach unten
>	Dreieck nach rechts
<	Dreieck nach links

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

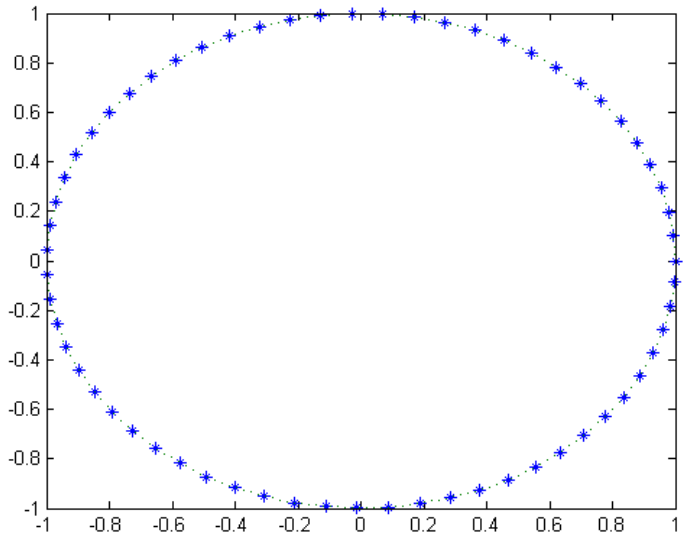
Die Parameter werden als String hinter die zu plottenden Daten geschrieben, z. B. `plot(x,y,'r',x2,y2,'-b')`.
Standard sind die Optionen `plot(...,'-','color',[0 0.45 0.74])`.

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
```



Plotten V

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.

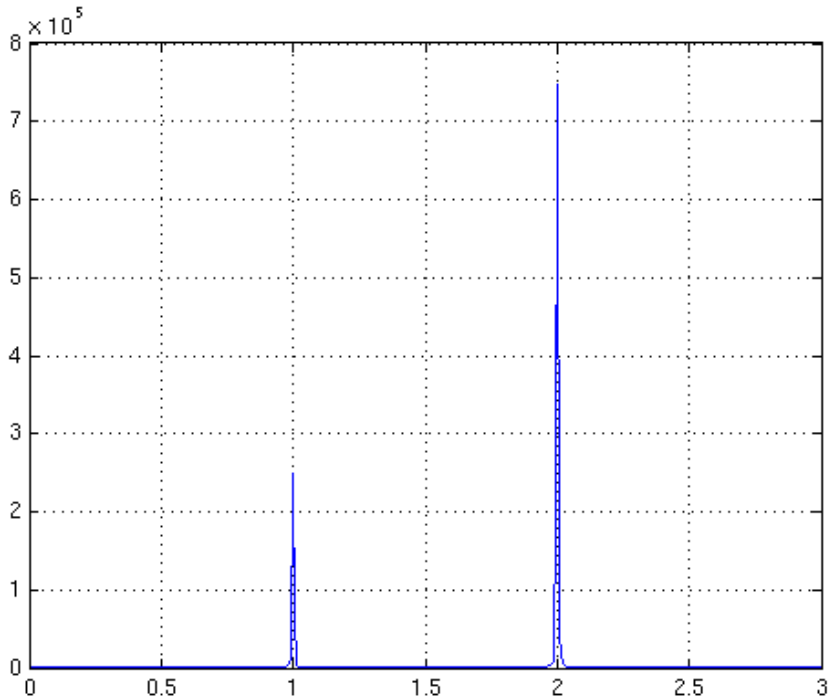
Hinweis: Für 3-D Plots gibt es auch `zlim`.

Plotten VI

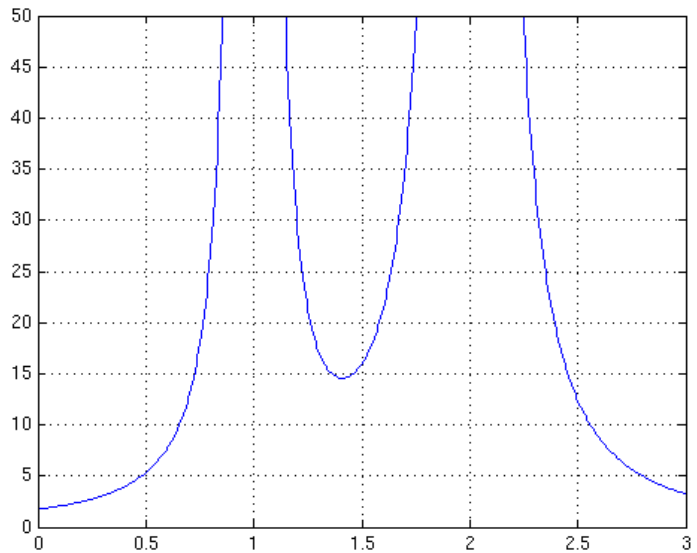
Wir betrachten die Funktion

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

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



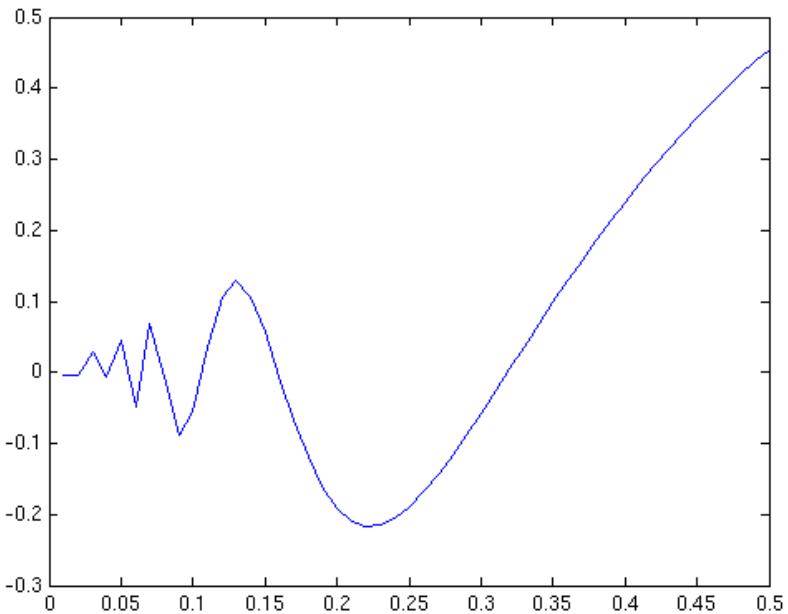

```
>> ylim([0 50])
```



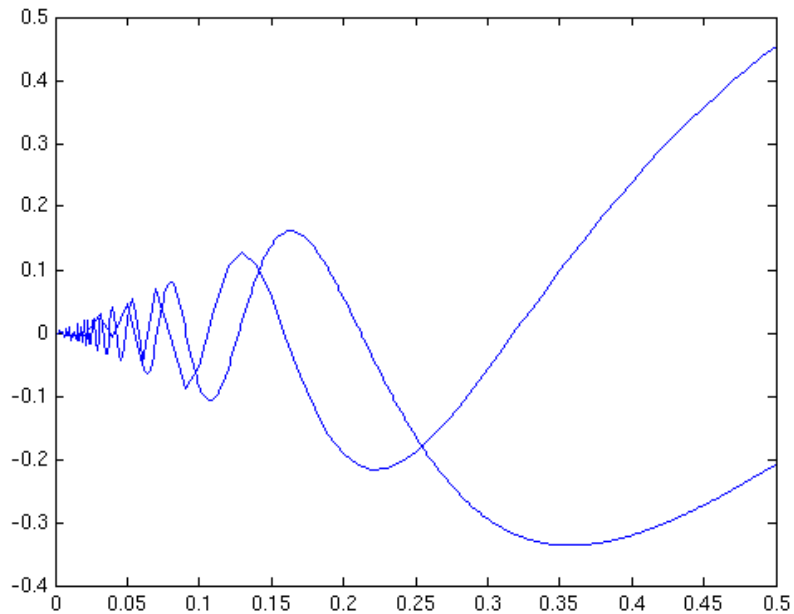
Plotten VII

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))
```

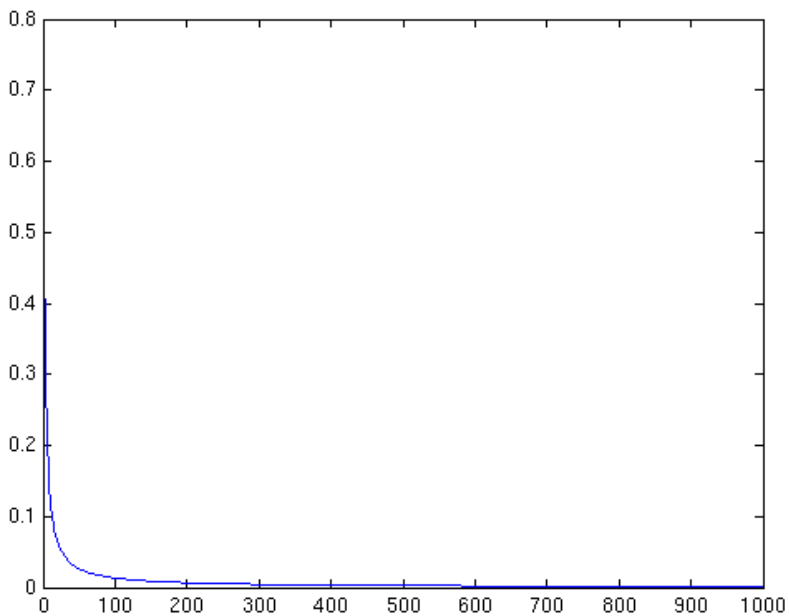


Verwende `figure` um ein neues Plot-Fenster zu öffnen.

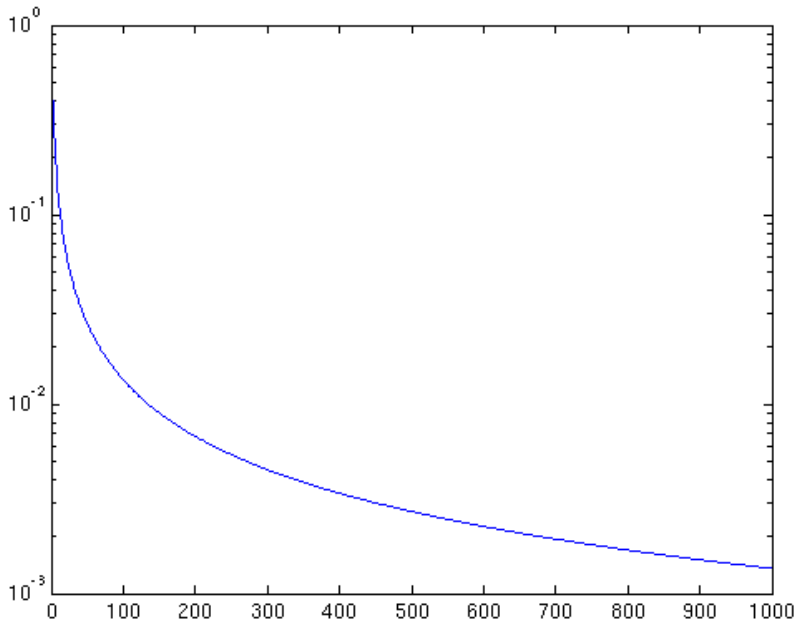
Logarithmische Achsen

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)
```



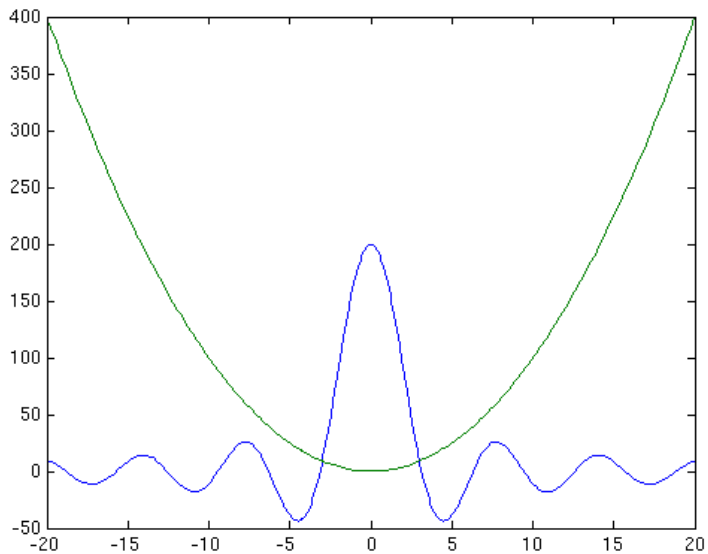
fplot

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

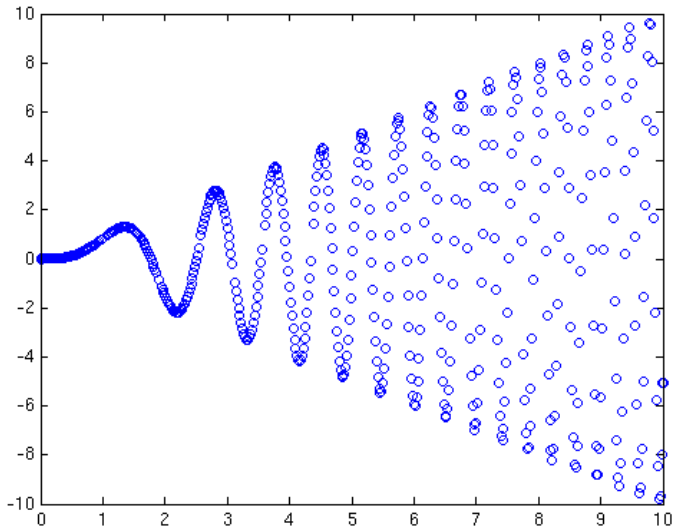
Syntax

`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');
```



subplot I

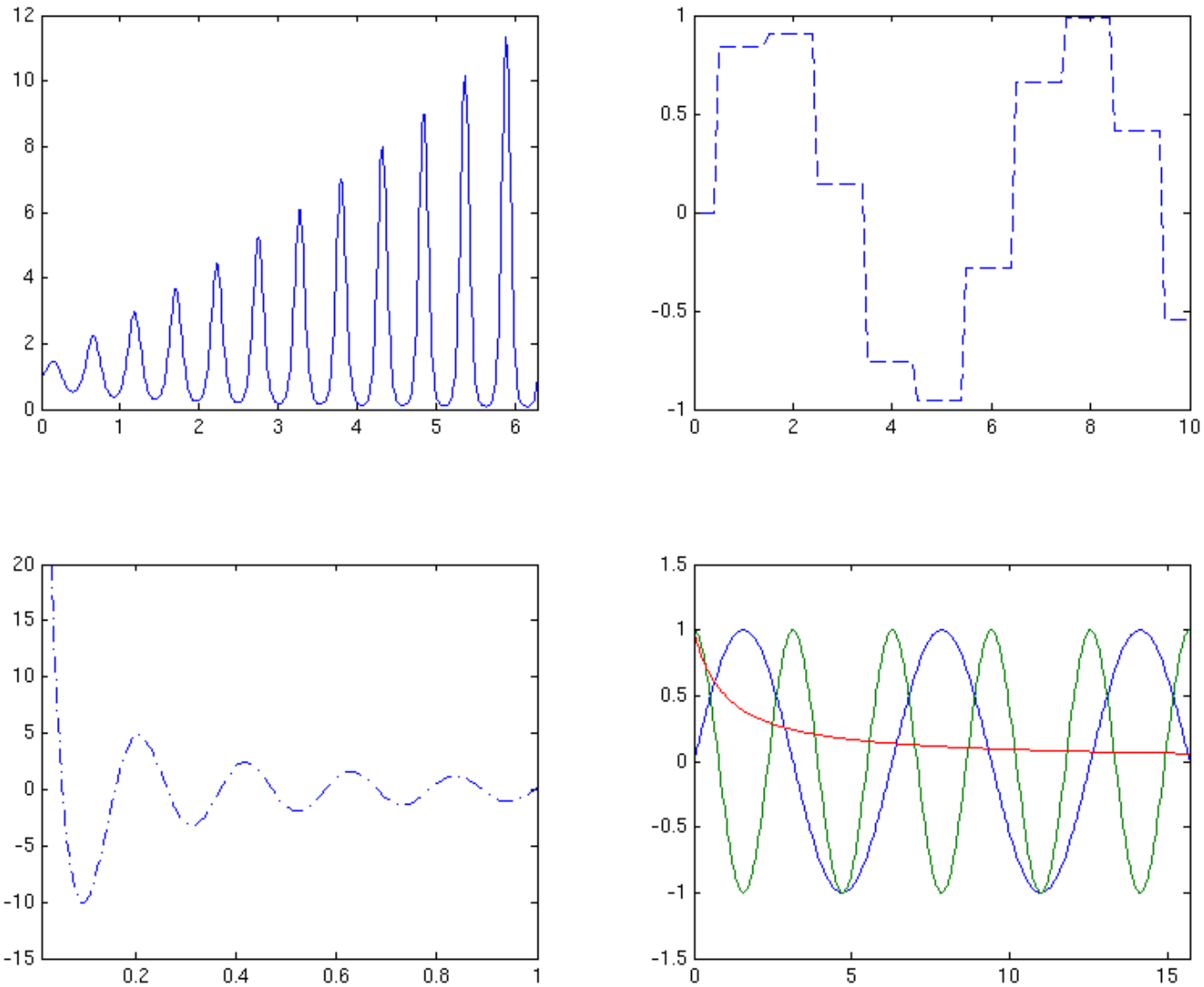
Mit dem Befehl `subplot` können mehrere Plots in ein Fenster in verschiedene Koordinatensysteme gezeichnet werden.

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


12
10
8
6
4
2
0
0 1 2 3 4 5 6
1
0.5
0
-0.5
-1
0 2 4 6 8 10
20
15
10
5
0
-5
-10
-15
0.2 0.4 0.6 0.8 1
1.5
1
0.5
0
-0.5
-1
-1.5
0 5 10 15

Inhalt

plot3 I

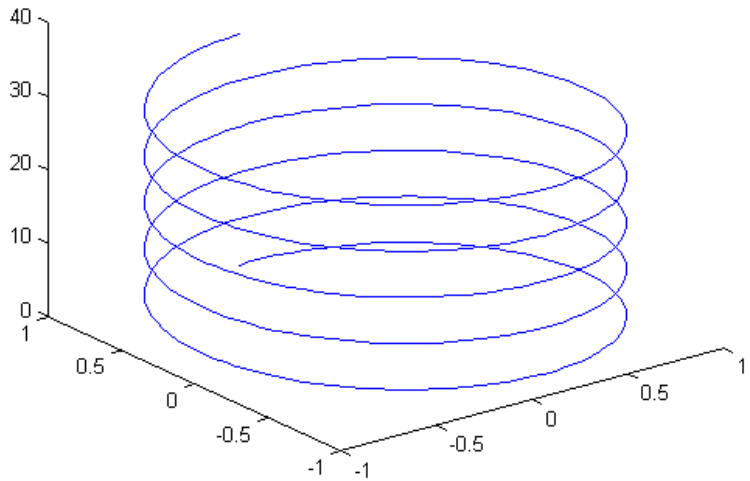
Mit dem Befehl `plot3` können Kurven $\omega\colon \mathbb{R} \to \mathbb{R}^3$ geplottet werden.

Syntax

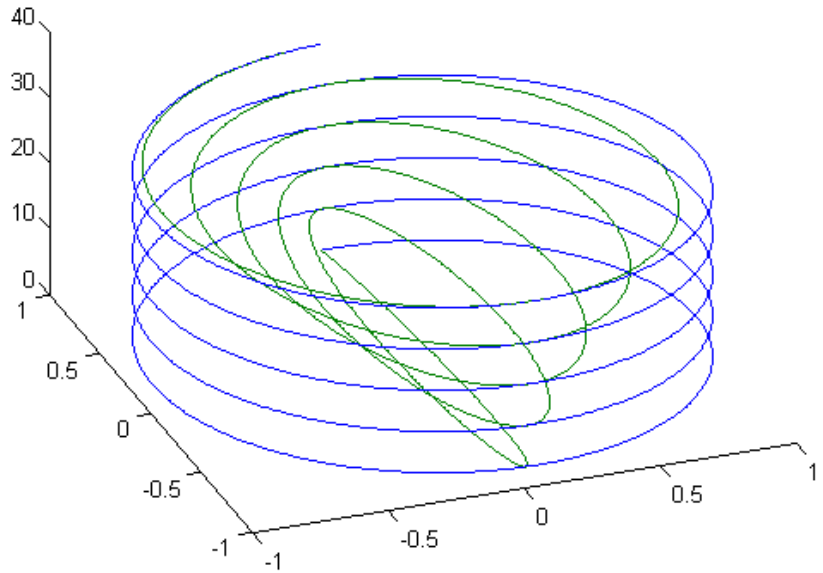
- `plot3(x,y,z)` mit Vektoren `x, y, z`: Plottet die Punkte `(x(i), y(i), z(i))` für alle $i \in 1, \ldots,$ `length(x)` und verbindet sie gegebenenfalls.
  Wichtig: `length(x) == length(y) == length(z)`.
- `plot3(X,Y,Z)` mit Matrizen `X, Y, Z`: Wie `plot3(X(:,i), Y(:,i), Z(:,i))` für alle Spalten `i`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

plot3 II

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

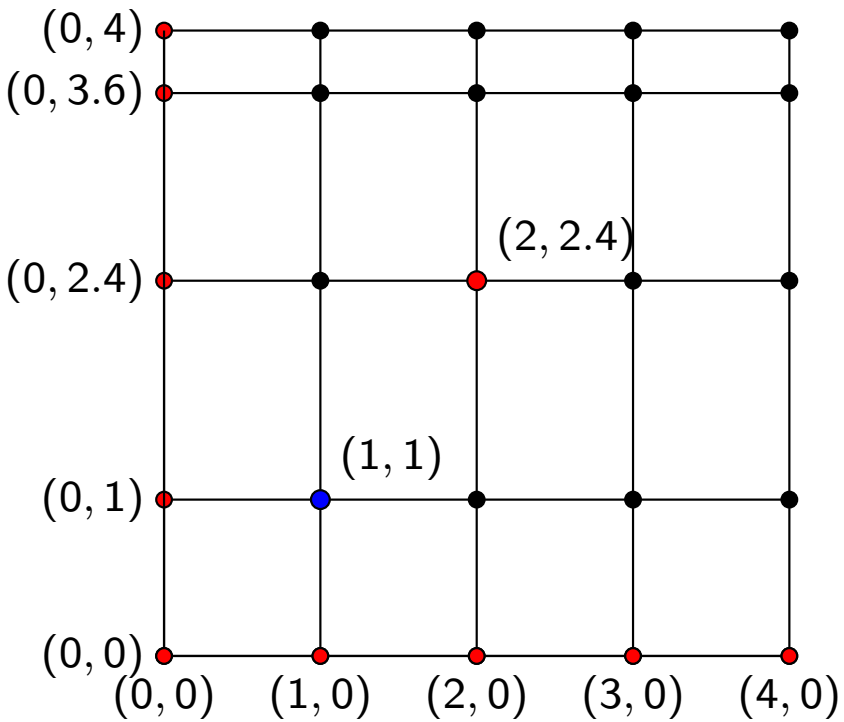



```
>> t = (0:0.01:10*pi)';
>> plot3([sin(t),t./(10*pi).*sin(t)],...
[cos(t), cos(t)], [t,t])
```



meshgrid

Um Funktionen $f : \mathbb{R}^2 \to \mathbb{R}$ 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    .
```

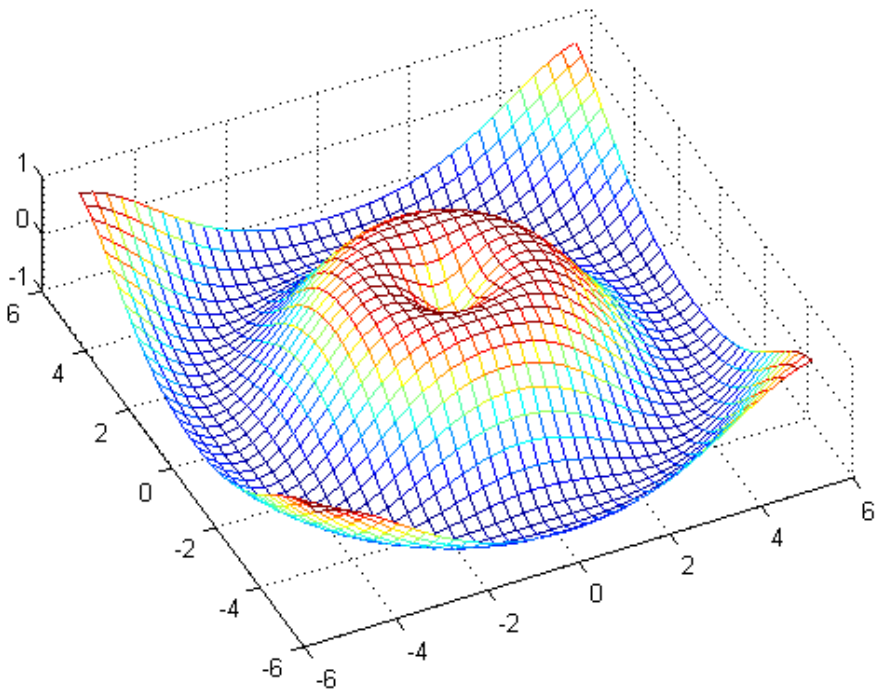
3D - Plots I

Haben wir alle Punkte mit `meshgrid` erzeugt, kann die Funktion anschließend $f : \mathbb{R}^2 \to \mathbb{R}$ 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:

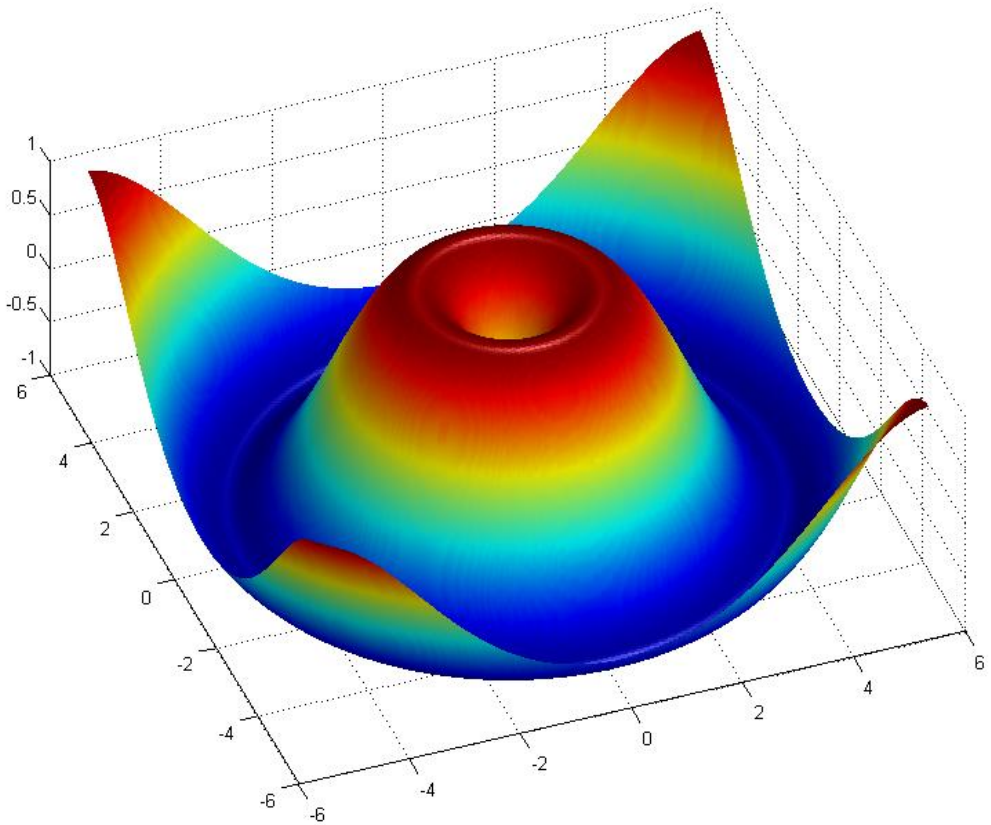
```
>> mesh(X,Y,Z);
```



3D - Plots II

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:

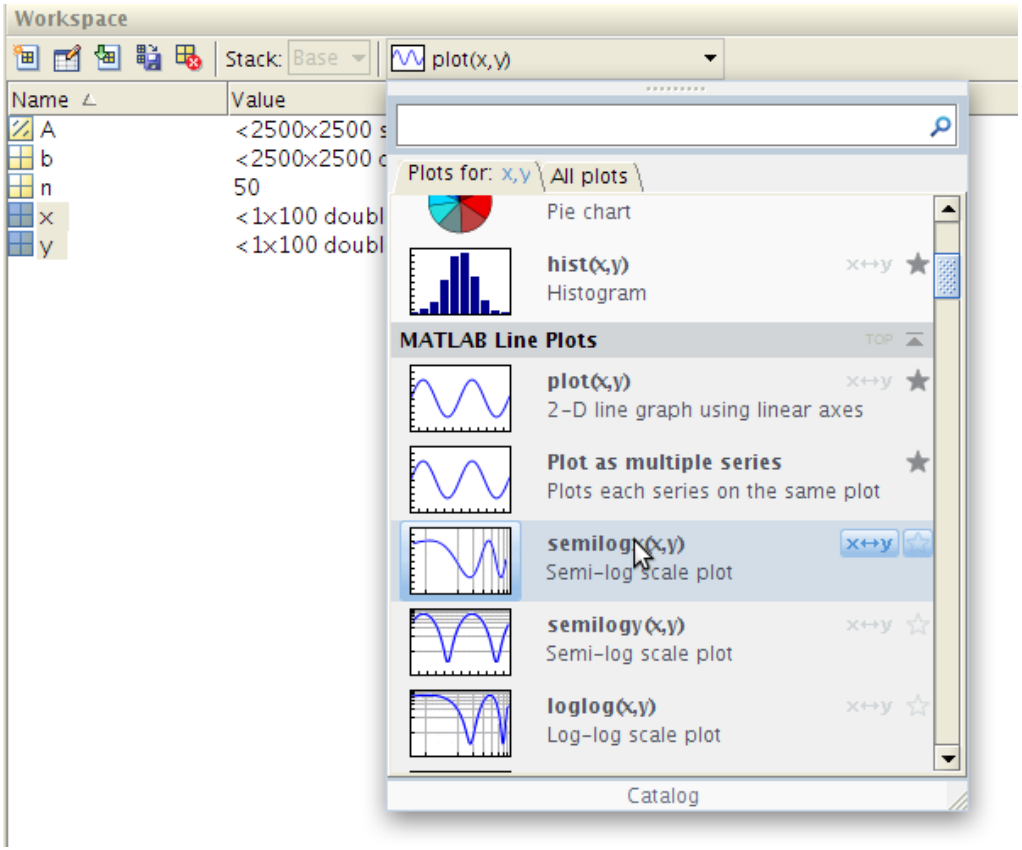
```
>> x = -5.5:0.05:5.5;
>> y = -5.5:0.05:5.5;
>> [X, Y] = meshgrid(x,y);
>> Z = f(X,Y);
>> surfl(X,Y,Z, 'light')
>> shading interp
```



Inhalt

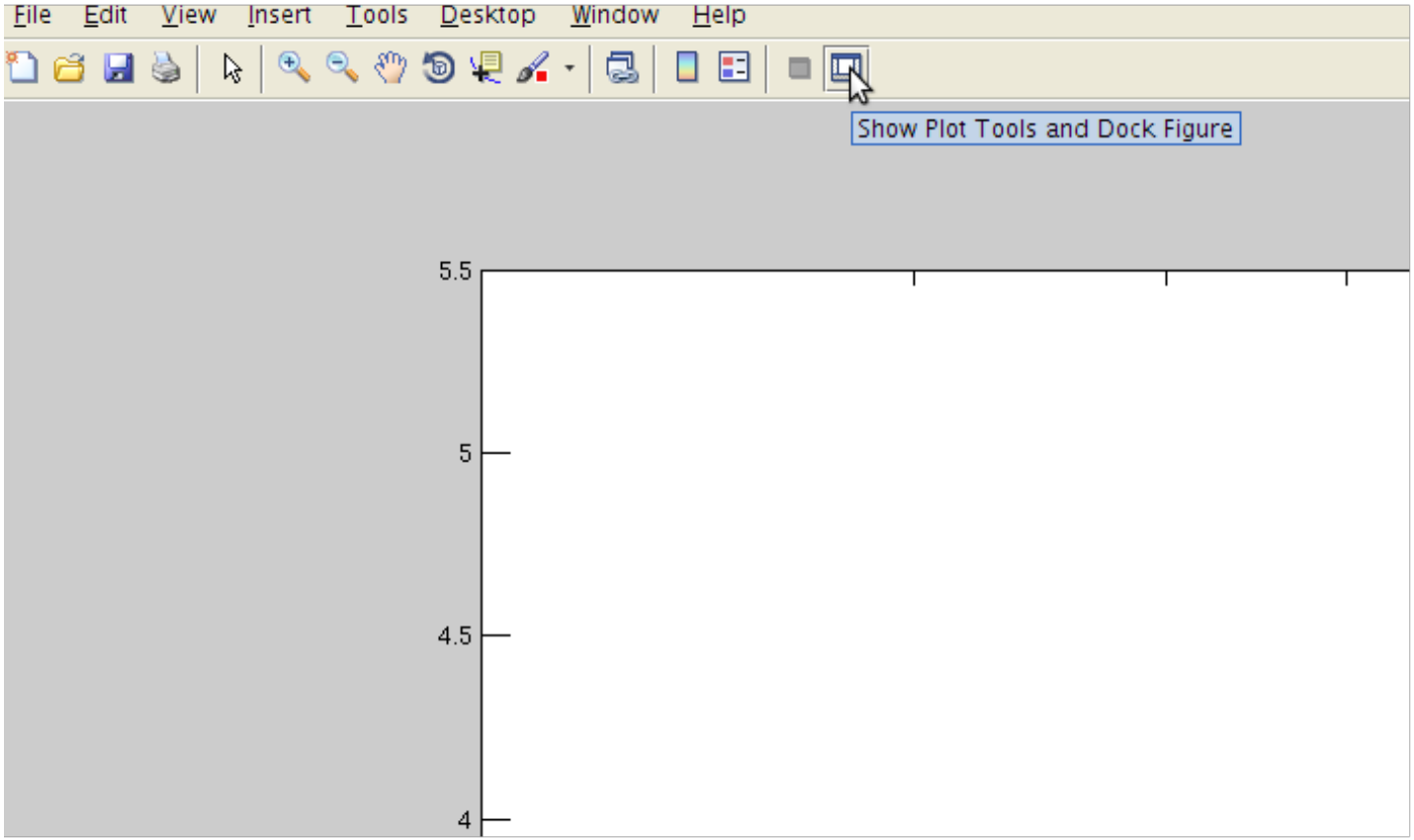
Plotten mit der GUI - 1

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:



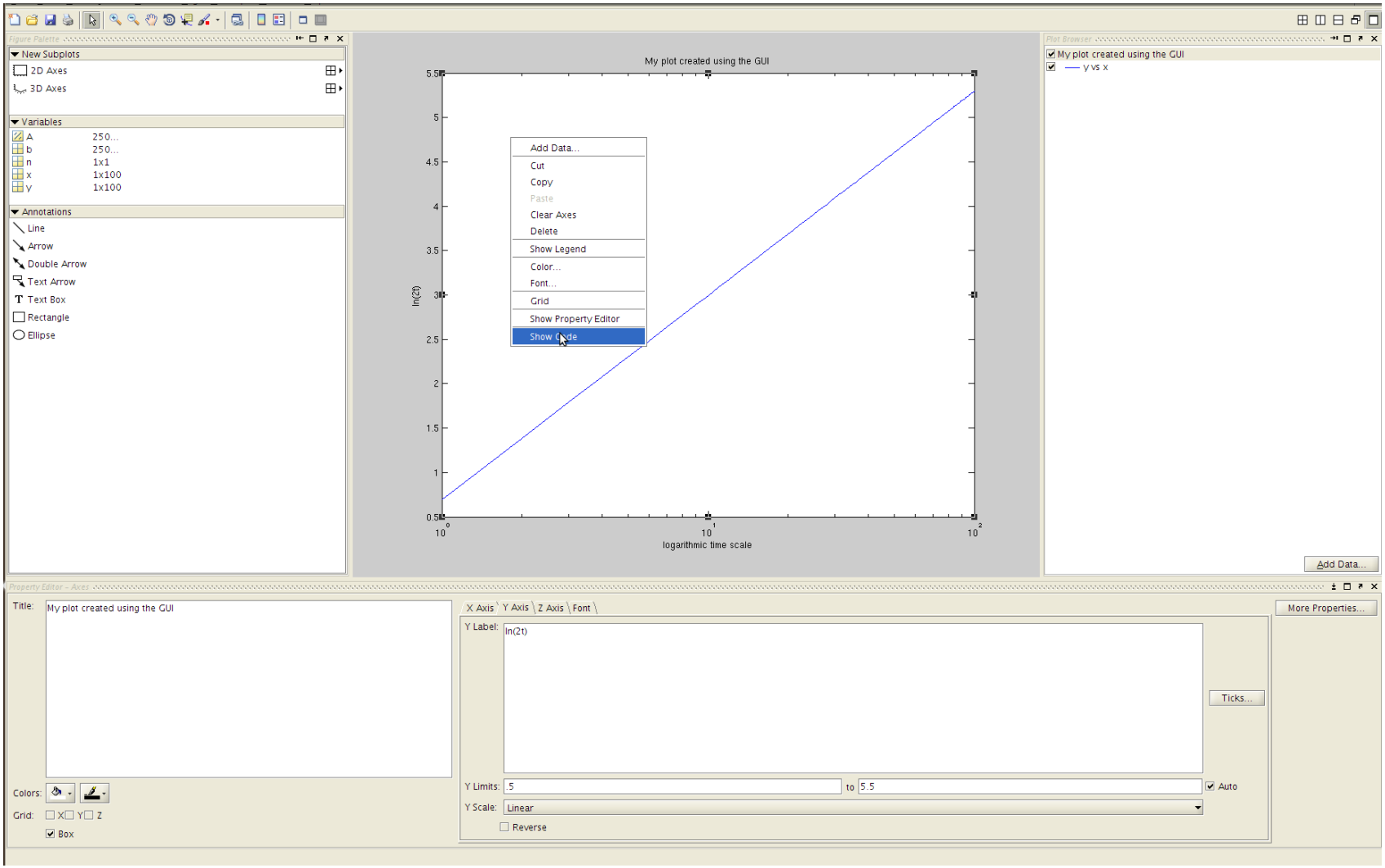
Plotten mit der GUI - 2

Öffnen des Editors um erzeugten Plot nachzubearbeiten:



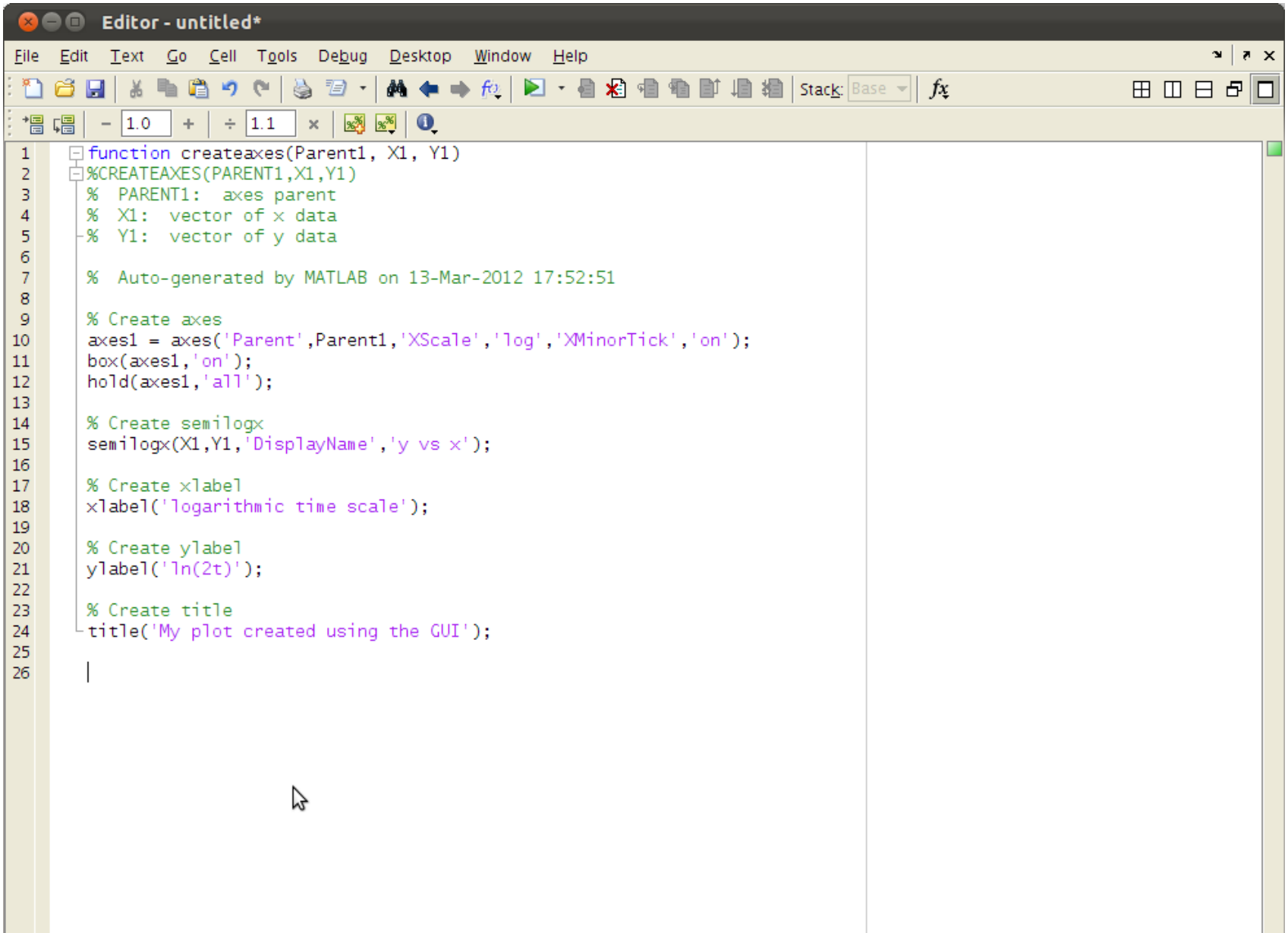
Plotten mit der GUI - 3

Anpassen der Formattierung und Code-Generierung:



Plotten mit der GUI - 4

Verwenden des generierten Codes ohne GUI und Mausclicks:



Ende Theorie 3.1

Fragen?