Technische Mechanik mit Computern

Labor-Einführung

Prof. Dr. Jürgen Dankert
Prof. Dr. Thomas Frischgesell
Prof. Dr. Michael Plenge
Prof. Dr. Stefan Reh

Übersicht

1. **Allgemeines zu TMC und MATLAB**
2. **MATLAB als Entwicklungs-Plattform**
3. **MATLAB als Programmiersprache**
4. **MATLAB als Graphik-Programm**

Übersicht

1. **Allgemeines zu TMC und MATLAB**
2. **MATLAB als Entwicklungs-Plattform**
3. **MATLAB als Programmiersprache**
4. **MATLAB als Graphik-Programm**

Allgemeine Informationen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

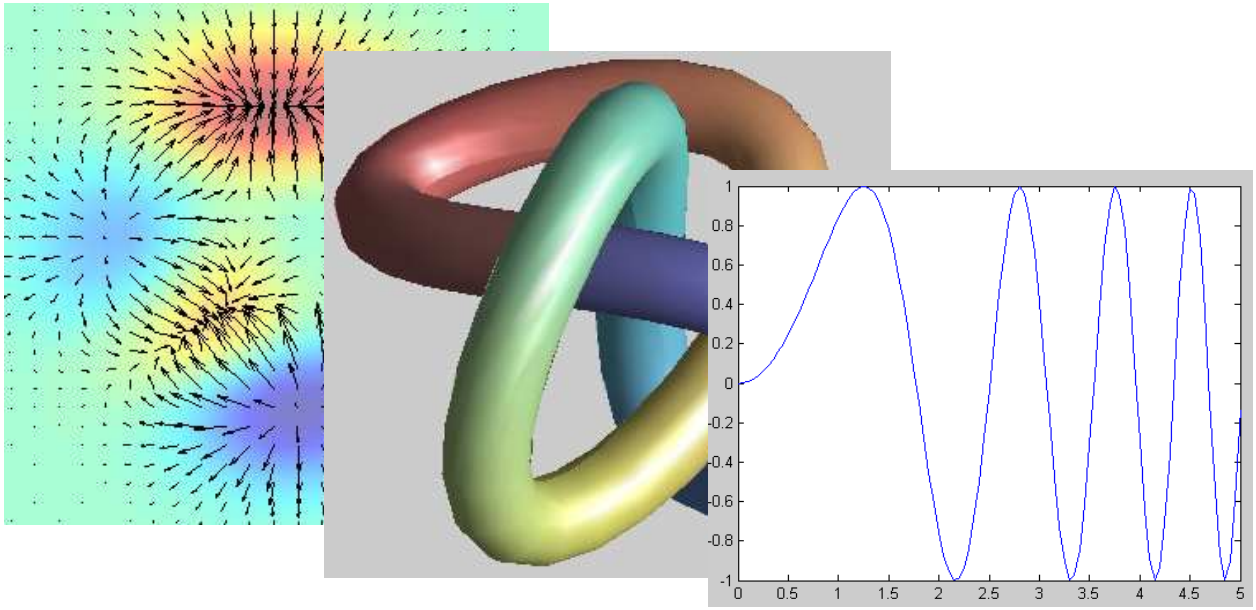
Homepage: http://www.dankertdankert.de

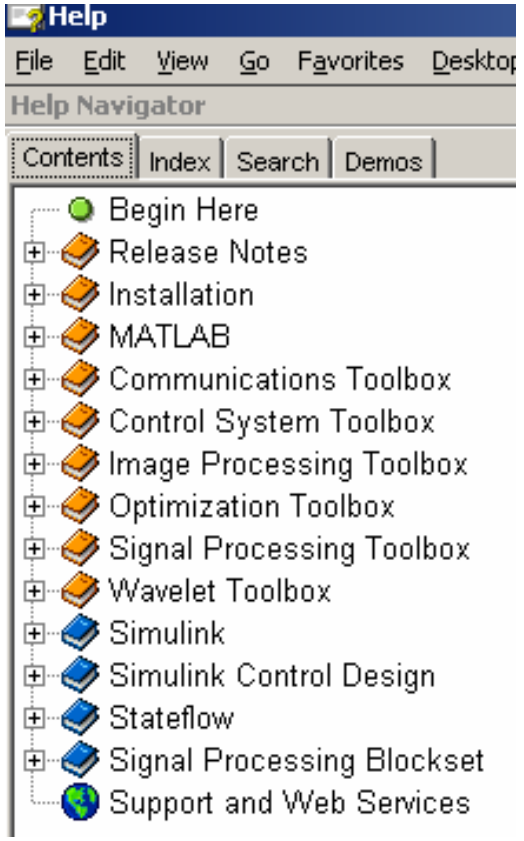
Link zur Startseite

Buch: Jürgen Dankert, Helga Dankert
„Technische Mechanik“
Teubner Verlag, 2004 (auch in Bibliothek)

MATLAB ist…

- **Integrierte Entwicklungsplatform (GUI, Befehle, Graphik)**
- **Programmiersprache**
- **Programmbibiliothek für Lösung aller Arten mathematischer Probleme (Standard & extra "Toolboxen")**
- **Graphik-Platform**





The MathWorks Inc.

The MathWorks Inc.
Natick, MA, USA

- **gegründet 1984 von Cleve Moler (CTO) und Jack Little (CEO)**
- **Hauptprodukte: MATLAB, Simulink**
- **1’200 Angestellte**
- **1’000’000 Kunden in mehr als 100 Ländern**
- **3’500 Hochschulen verwenden/lehren MATLAB**

MATLAB Literatur

MATLAB - Zahlreiche Literatur zu MATLAB:

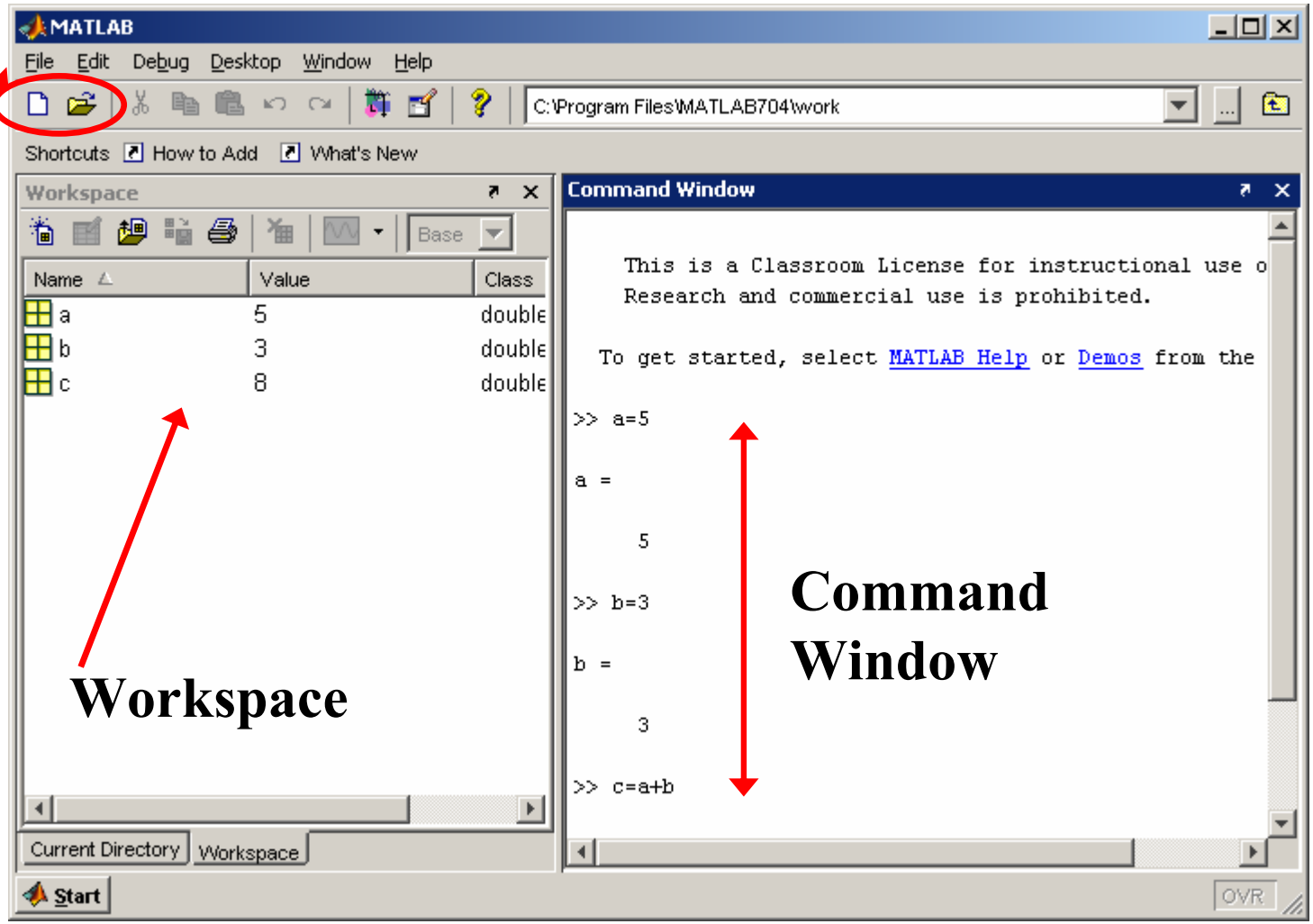
- Cleve Moler „Numerical Computing with MATLAB“, verfügbar kapitelweise unter http://www.mathworks.com/moler/
- Steven Chapra „Applied Numerical Methods with MATLAB for Engineers and Scientists“, McGrawHill, 2005
- Adrian Biran, Moshe Breiner „MATLAB 5 für Ingenieure”, Addison-Wesley (auch in Bibliothek)

Übersicht

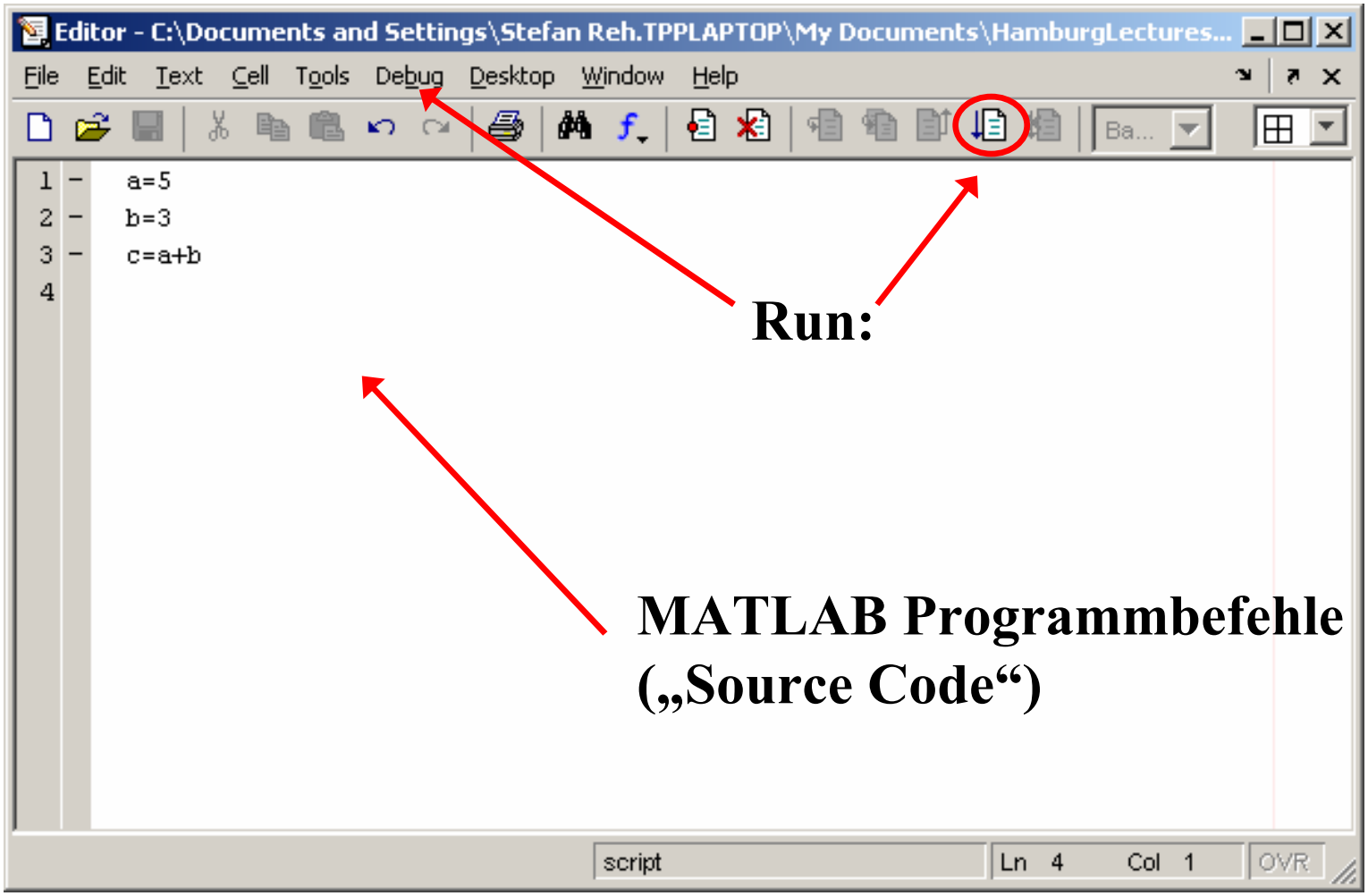
1. Allgemeines zu TMC und MATLAB
2. **MATLAB als Entwicklungs-Plattform**
3. MATLAB als Programmiersprache
4. MATLAB als Graphik-Programm

Main-Window

File Editor: Neue Datei oder existierende Datei



File Editor


Editor - C:\Documents and Settings\Stefan Reh.TPPLAPTOP\My Documents\HamburgLectures...
File Edit Text Cell Tools Debug Desktop Window Help
1 a=5
2 b=3
3 c=a+b
4
Run:
MATLAB Programmbefehle
(„Source Code“)
script
Ln 4 Col 1
OVR

File Editor

Achtung:

Name von „Source Code“-Dateien (m-Files) darf nur rein Alpha-numerisch sein!! D.h.:

- **keine Sonderzeichen &%$@#...**
- **keine Leerzeichen**
- **keine Umlaute öüäß ....**

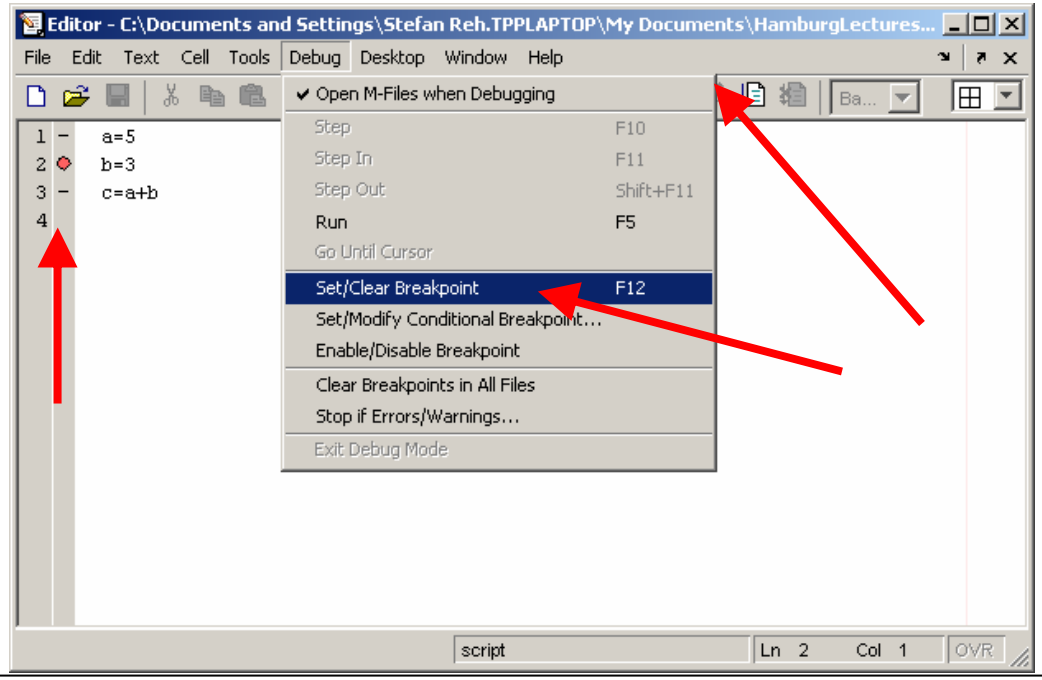
Nur „_“ ist erlaubt (besser DOS kompatibel)

Starten von MATLAB nicht durch Doppelklick des m-Files im File Explorer. Dies startet File Editor im reinen Editiermodus, d.h. ohne Programmausführung (kein Run).

Debugger

1. Breakpoint setzen
   - Curser in Zeile + Drücke Breakpoint Icon 
   - Curser in Zeile + Debug>Set/Clear Breakpoint
   - Click vor Zeile nach Zeilennummernspalte

   Ggf. Modifiziere Bedingung für Breakpoint (Rechte Mouse-click)
2. Run
3. Step (F10) / Step In (F11) / Step Out (F12) … wie Visual Studio

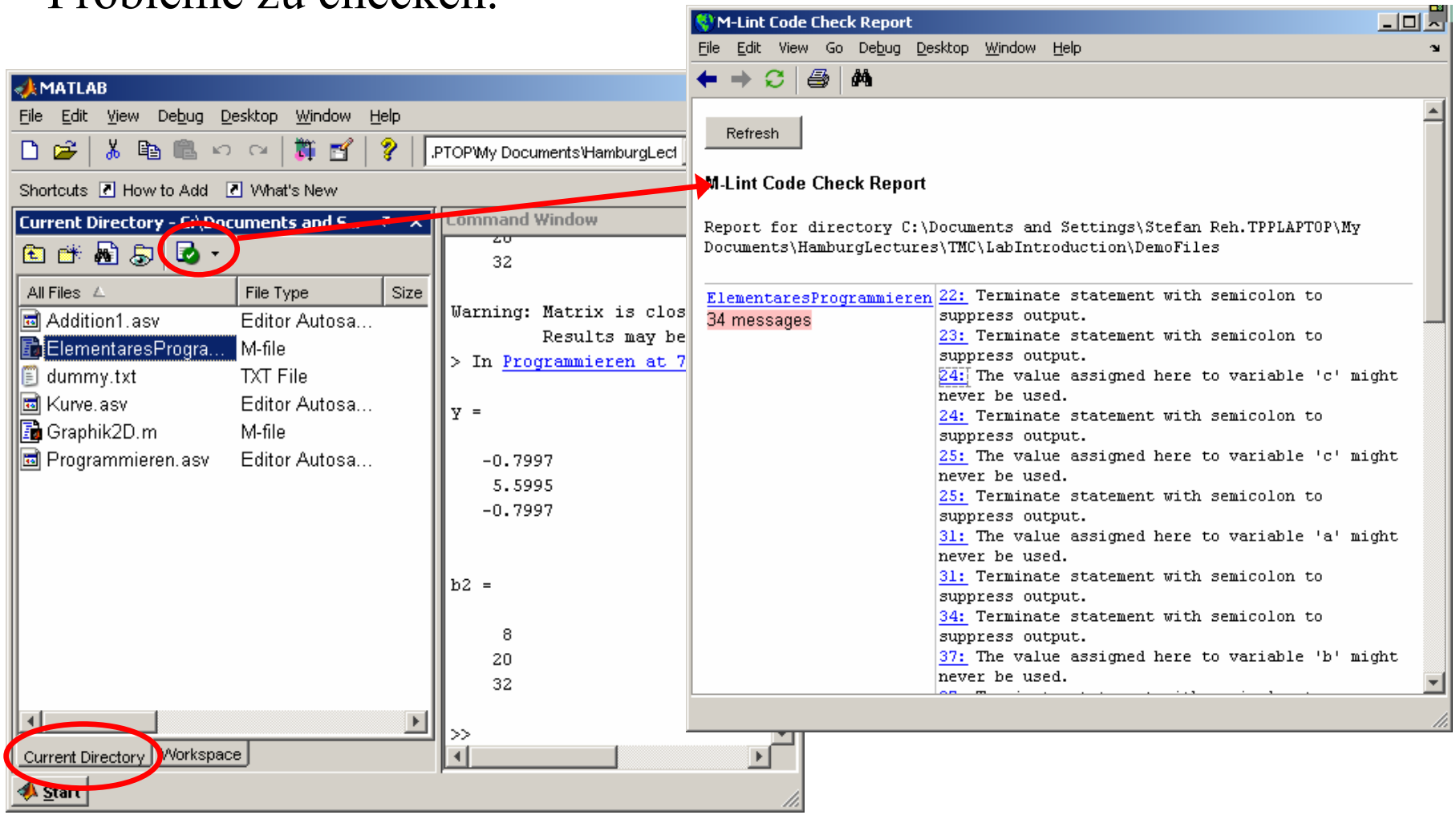


Debugger

Bei Auftreten eines Fehlers, stellt MATLAB im Command-Window einen Link zu der Stelle zur Verfügung, an der der Fehler auftritt.
ACHTUNG: Bitte auch Fehlertext sorgfältig lesen!!

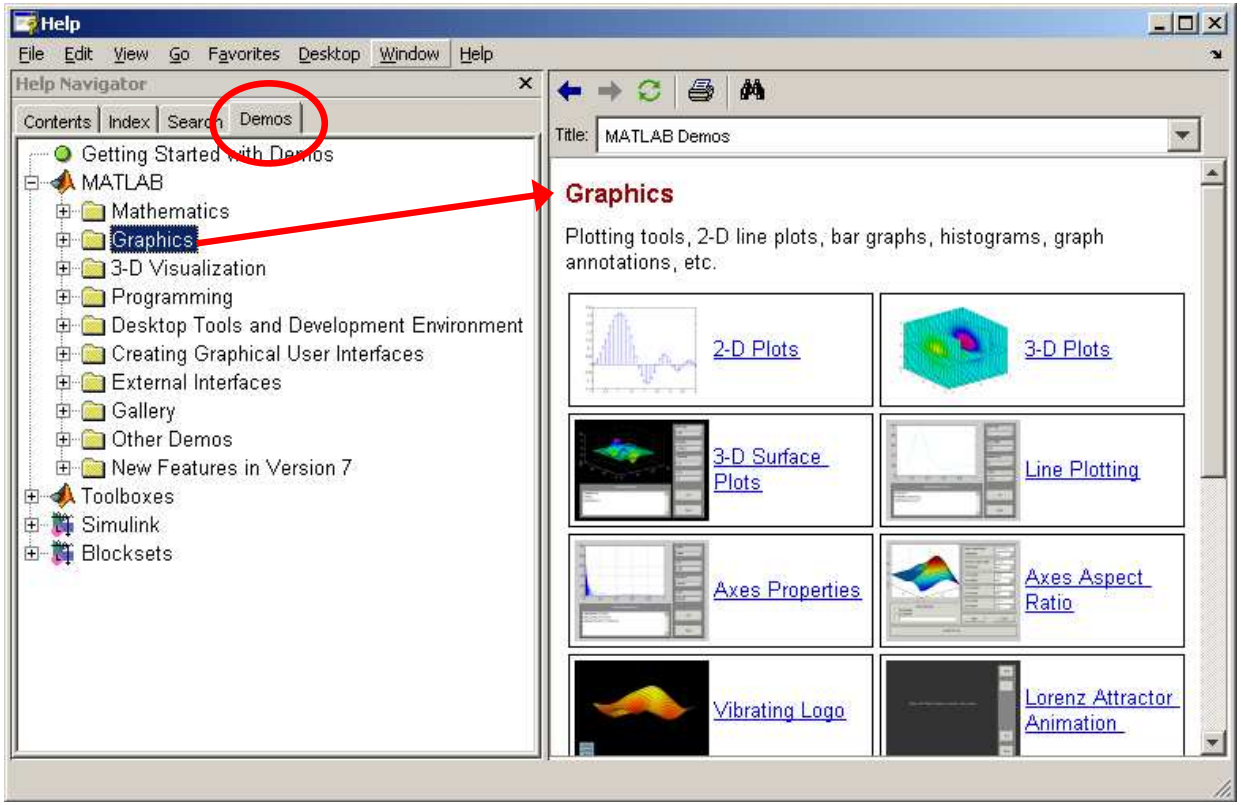
M-Lint Code Checker

M-Lint ist ein komfortables Tool um Code automatisch auf Probleme zu checken.



MATLAB Help

MATLAB help bietet auch Demos, die zu allen wichtigen Themen Videos und Presentationsfolien.



Siehe auch:

http://www.mathworks.com/products/matlab/demos.jsp

Ansehen: Mathematics -> Basic Matrix Operations

Übersicht

1. Allgemeines zu TMC und MATLAB
2. MATLAB als Entwicklungs-Plattform
3. **MATLAB als Programmiersprache**
4. MATLAB als Graphik-Programm

Die wichtigsten Regeln

- **MATLAB ist ein Interpreter (Scripting language)**
- **Es ist in vielen Bereichen der Sprache C sehr ähnlich**
- **Es ist keine Vereinbarung von Variablen erforderlich**
- **Initialisierung ist aber ein MUSS (wie in C auch)!!**
- **Bei Variablennamen ist Groß- und Kleinschreibung signifikant**
- **Alles was nach einem "%" steht ist Kommentar**
- **Semikolon heißt nach Befehlen: "Kein Prompt"**
- **Semikolon heißt in einer Matrix: "Nächte Zeile"**
- **Bekannte Konstanten: "pi", "i"**
- **Abbruch eines zu lange laufenden Programms (z.B. Endlos-Schleife) mit "Ctrl + C" ("Strg + C") im Main-Window**

Programmieren: Eingabe/Ausgabe

Eingabe:

```
Variable = input('PromptString')
```

z.B.:

```
Masse = input('Gib die Masse (kg) ein:')
```

Einfache Ausgabe:

```
disp('Text')
disp(Variable)
```

Formatierte Ausgabe:

```
fprintf('FormatAngaben', Variable, Variable, …)
```

Dabei gelten die gleichen Formatspezifizierungen wie bei "C" (%d, %e, %f,…).

Programmieren: Matrizen-Definition

Zeilen-Vektor

```
>> d=[1 2 3 4 5 7]
d =
     1    2    3    4    5    7
```

Matrix

```
>> f=[1 2 3 4;
      5 6 7 8]
f =
     1     2     3     4
     5     6     7     8
```

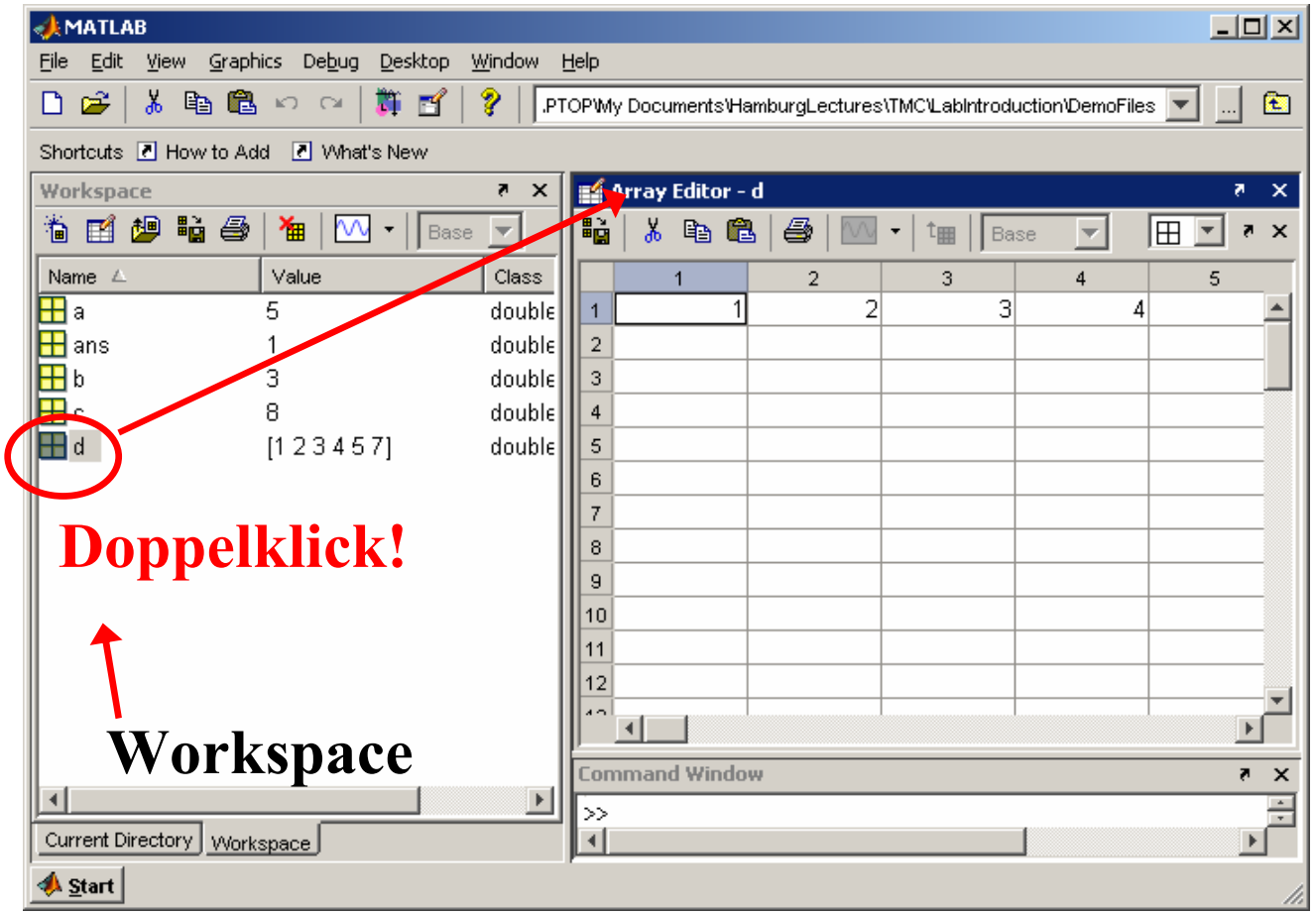
Spalten-Vektor

```
>> e=[3;5;8;12;354]
e =
     3
     5
     8
    12
   354
```

Semikolon heißt hier:
Neue Zeile!

Array (Variablen) - Editor


```
>> d=[1  2  3  4  5  7]
d =
     1     2     3     4     5     7
```



Programmieren: Rechen-Operationen

- **Alle Grundrechenoperationen (+, -, *, /) und auch potenzieren (^) können angewendet werden auf**
  - **Skalare Größen**
  - **Skalare Größen in Verbindung mit Matrizen**
  - **Matrizen untereinander**
- **Werden Rechenoperationen auf Matrizen untereinander angewendet, dann muss die Dimension stimmen**

Matrix-Operationen

```
>> d1=[1 2 3 4 5 7];
>> d2 = 2*d1;
d2 =
     2    4    6    8    10    14
>> d3=d1+d2;
d3 =
     3    6    9    12    15    21
```

Programmieren: Matrix-Operationen

- **Linksseitige Division zum Lösen von Gleichungssystemen**

  $A*x = b$ | $A^{-1}*(\ldots)$ links mit Invertierten von A multiplizieren

  $A^{-1}*A*x = A^{-1}*b$

  $x = A^{-1}*b$

  In MATLAB: `x = A\b`

- **Transponieren** `B = A'`

- **Elementweise Operationen:**
  - **Potenzieren:** `B = A.^2` **ist nicht gleich** `B = A^2`

Programmieren: Funktionsbibliothek

MATLAB hat eine riesige Bibliothek vordefinierter Funktionen

- **Mathematische Funktionen**
  sin(x), exp(x), cos(x)

- **Lösungsroutinen**
  - **Nullstellensuche:**

```
Nstelle = fzero(@FunHdl, AnfWert, options)
Nstelle = fzero(@FunHdl, Ival, options)
```

  - **Minimalwertsuche:**

```
Minstelle = fminsearch(@FunHdl, AnfWert,
                                options)
```

Bedingungen:

```
if LogischeBedingung
        …
elseif LogischeBedingung        ← optional
        …
elseif LogischeBedingung        ← optional
        …
else                            ← optional
        …
end
```

besser: Bedingung klammern!! `if (LogischeBedingung)`

LogischeBedingung **sind (gleich wie bei "C") z.B.:**

A<0
b>1 && c=5 (und-Verknüpfung)
b>1 || c=5 (oder-Verknüpfung)

while-Schleifen:

```
while (LogischeBedingung)
        …
end
```

for-Schleifen:

```
for index = StartWert:EndWert
        …
end
```

SchrittWeite automatisch 1

Oder

```
for index = StartWert:SchrittWeite:EndWert
        …
end
```

SchrittWeite explizit gegeben

Implizite for-Schleifen: **SchrittWeite automatisch 1**

```
t = StartWert:EndWert
y(StartWert:EndWert) = … StartWert:EndWert …
```

Oder **SchrittWeite explizit gegeben**

```
t = StartWert:SchrittWeite:EndWert
y(StartWert:SchrittWeite:EndWert) =
                … StartWert:SchrittWeite:EndWert …
```

Oder

```
fprintf(' y=%5.2f',y);
```

Schleife, da y Vektor

- **Mögliches Ergebnis ist dann ein Vektor.**
- **Automatische Re-dimensionierung von Ergebnisvektoren**

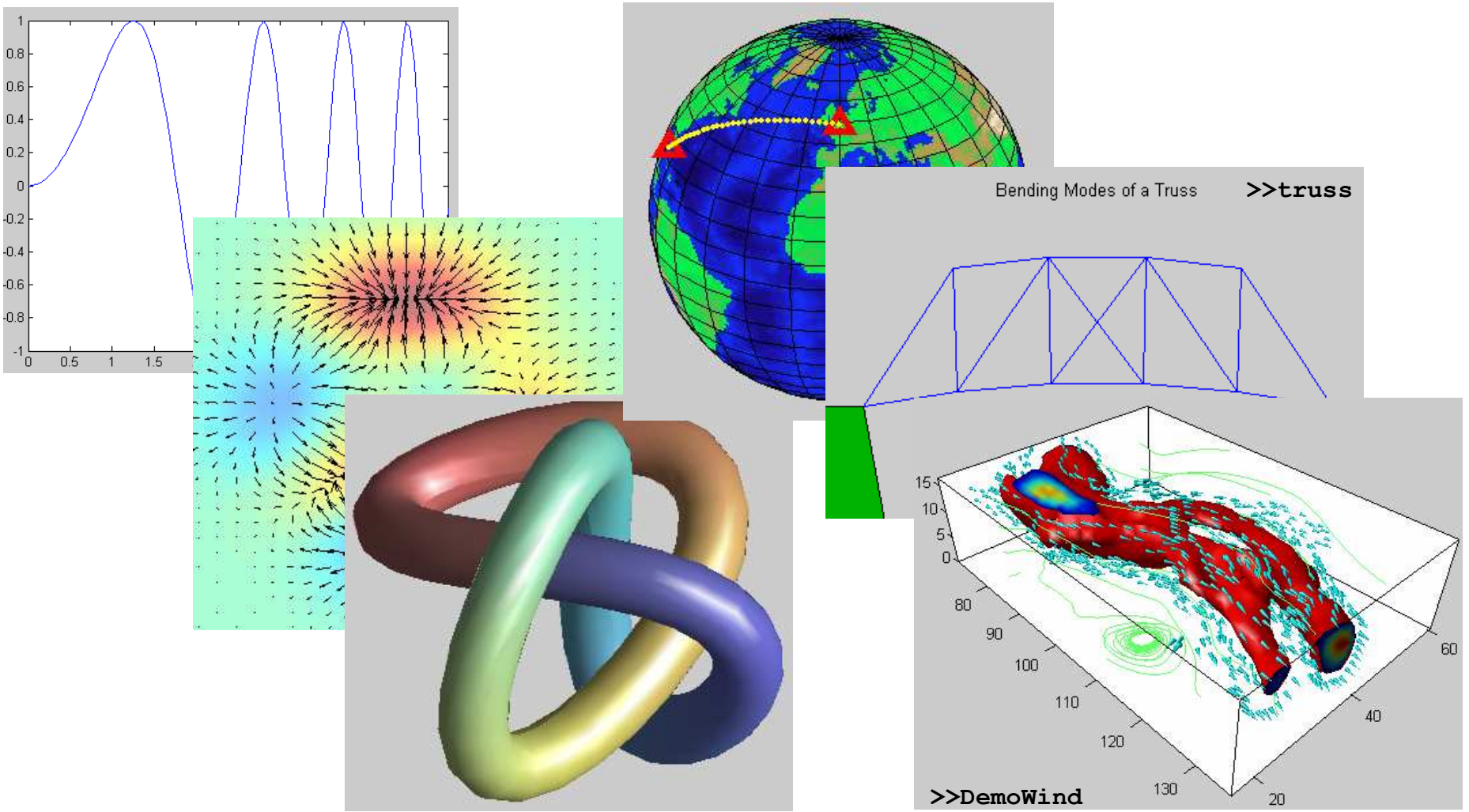
Übersicht

1. Allgemeines zu TMC und MATLAB
2. MATLAB als Entwicklungs-Plattform
3. MATLAB als Programmiersprache
4. **MATLAB als Graphik-Programm**

MATLAB Graphik

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

MATLAB hat zahlreiche Graphikmöglichkeiten



Einfache XY-Plots

Syntax:

`x, y` sind Vektoren

`plot(y)`

plotten von y mit Index auf x-Achse

`plot(x, y)`

plotten y =f(x)

`plot(x, y, LineSpec)`

plotten y =f(x) mit Spezifizierung von Farbe, Marker und Linenart

`plot(…, 'PropertyName', PropertyValue,… )`

plotten y =f(x) mit detailierter Spezifizierung aller Eigenschaften für Linie, Marker, Farbe

Mehrfache XY-Plots in einem Diagramm

Mehrfache XY-Plots gleichen Typs

Syntax: **x_1, y_1, x_2, y_2 sind Vektoren**

`plot(`y_1`, `y_2`)`

plotten von y_1 und y_2 mit Index auf x-Achse

`plot(`x_1`, `y_1`, `x_2`, `y_2`)`

plotten $y_1 = f(x_1)$ und $y_2 = f(x_2)$

`plot(`x_1`, `y_1`, `x_2`, `y_2`, LineSpec)`

plotten $y_1 = f(x_1)$ und $y_2 = f(x_2)$ mit Spezifizierung von Farbe, Marker und Linenart

`plot(…,'PropertyName', PropertyValue,… )`

plotten $y_1 = f(x_1)$ und $y_2 = f(x_2)$ mit detailierter Spezifizierung aller Eigenschaften für Linie, Marker, Farbe

Hinweis:
Dies geht auch mit mehr als 2 XY-Paaren

Mehrfache XY-Plots in einem Diagramm

Mehrfache XY-Plots gleichen oder unterschiedlichen Typs

Syntax:

```
plot(...)     <= erster Plot
hold on;      <= ... es kommen noch weitere Plots
plot(...)     <= zweiter Plot
...
hold off;     <= ... jetzt sind wir fertig
```

ODER

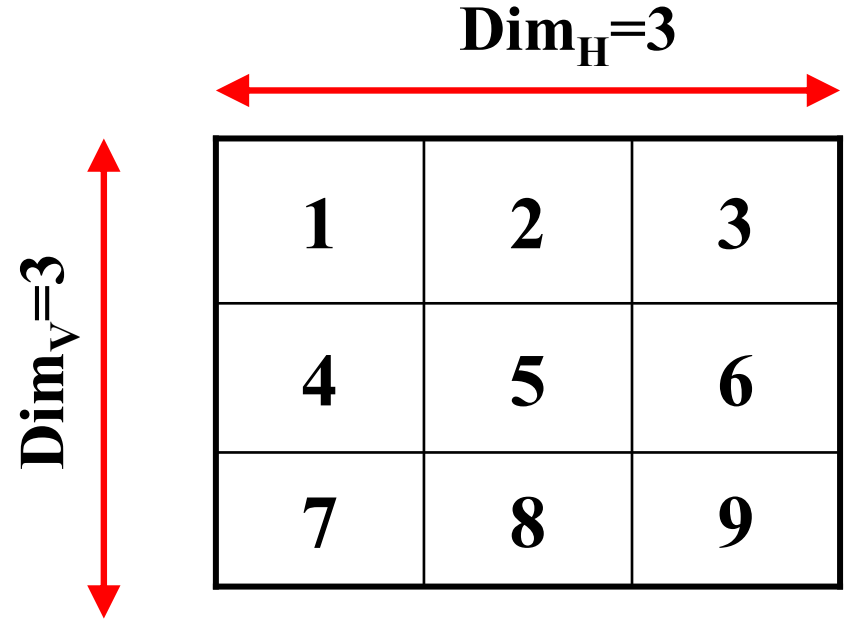
```
plot(...)     <= erster Plot - Linenplot
hold on;      <= ... es kommen noch weitere Plots
bar(...)      <= zweiter Plot - Balkendiagramm
...
hold off;     <= ... jetzt sind wir fertig
```

Mehrfache XY-Plots in Unter-Diagramm

Syntax:

```
subplot(Dim_H, Dim_V, Index1);
plot(…);
subplot(Dim_H, Dim_V, Index2);
plot(…);
…
```

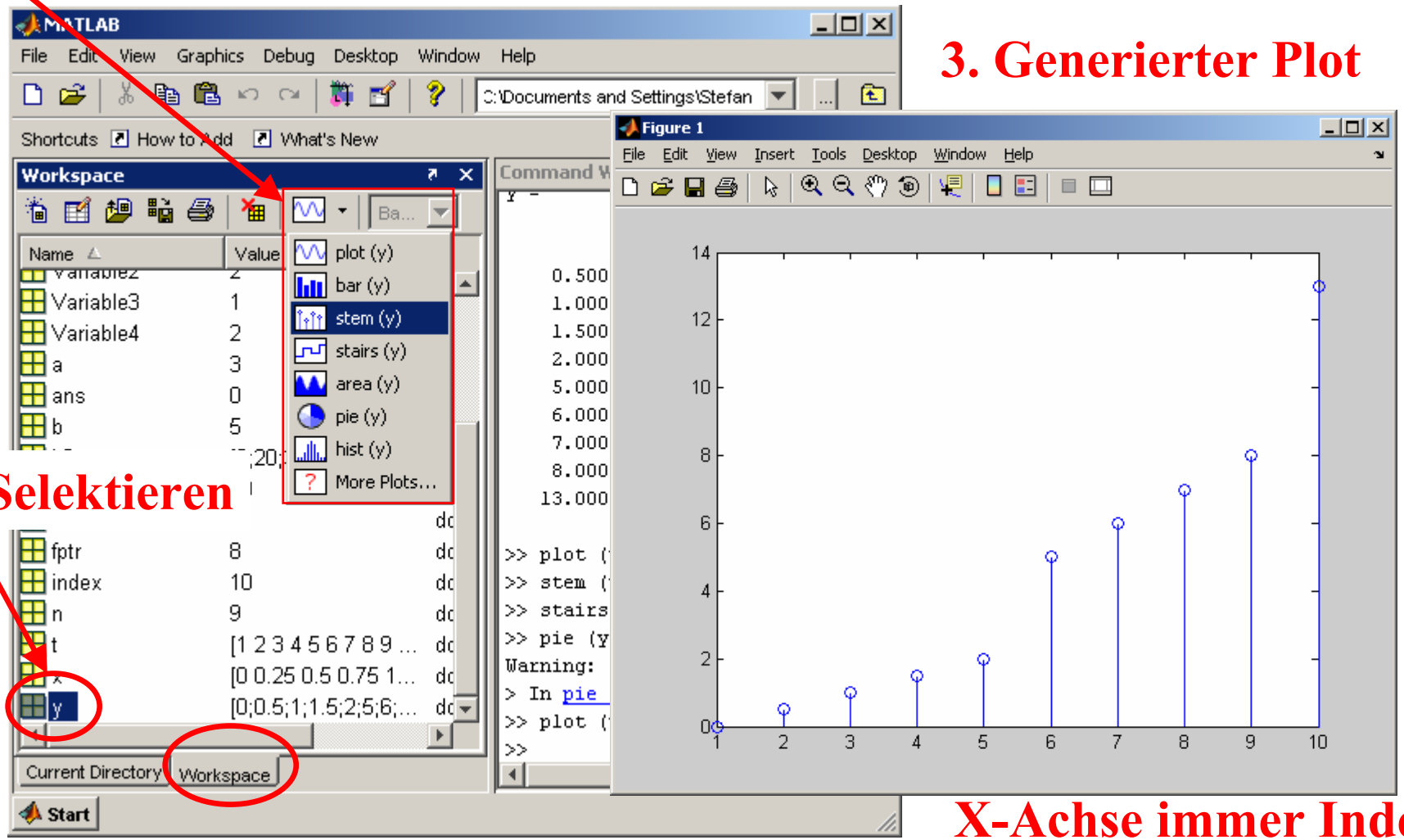
Indizierung Beispiel 3x3 plots



Indizierung erfolgt zeilenweise!

Quick-Plots von Vektoren/Matrizen

2. Drop-Down: Typ wählen (Unterschiede Vektor 2D/Matrix 3D)



3. Generierter Plot

1. Selektieren

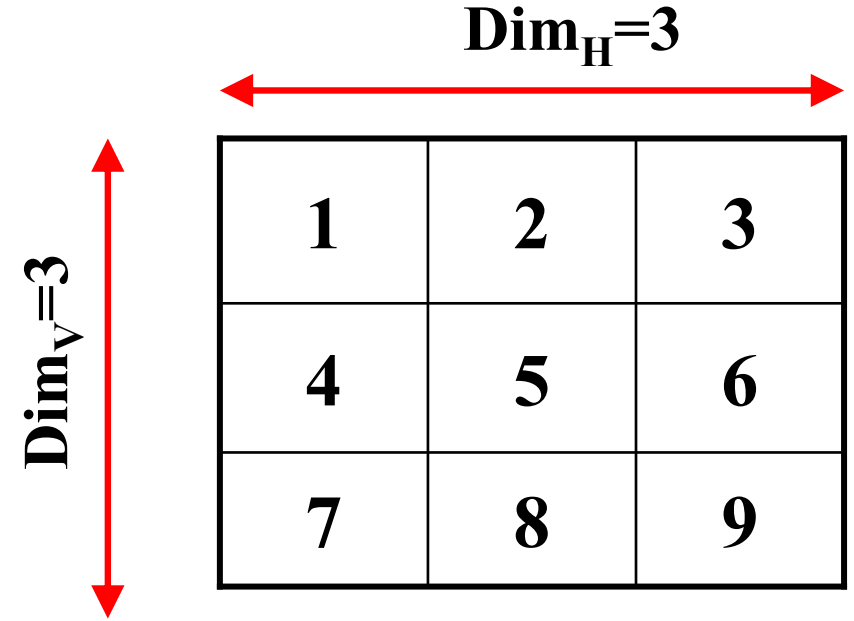
X-Achse immer Index

Mehrfache 3D-Plots in Unter-Diagramm

Syntax:

```
subplot(DimH, DimV, Index1);
surf(…);
subplot(DimH, DimV, Index2);
surf(…);
…
```

Indizierung Beispiel 3x3 plots



Indizierung erfolgt zeilenweise!