

Matlab – Einführung

Einführung in die Neuroinformatik SS 12

Miriam Schmidt
Institut für Neuroinformatik
Email: miriam.k.schmidt@uni-ulm.de



ulm university universität
uulm

Was ist Matlab?

Matlab ...

- ist die Abkürzung für Matrix Laboratory.
- ist ein integriertes, interaktives System zur Berechnung, Visualisierung oder der Programmierung mathematischer Ausdrücke.
- ist eine sehr einfach zu erlernende Skriptsprache, die ganz auf die Verarbeitung von Matrizen ausgelegt ist.
- ist eine riesige Sammlung an numerischen und graphischen Funktionen.
- ist eine interpretierte Skriptsprache:
 - Vorteile: Komfort, einfaches Debugging, Absturzsicherheit.
 - Nachteile: langsamer als kompilierter Code, insb. bei Schleifen.

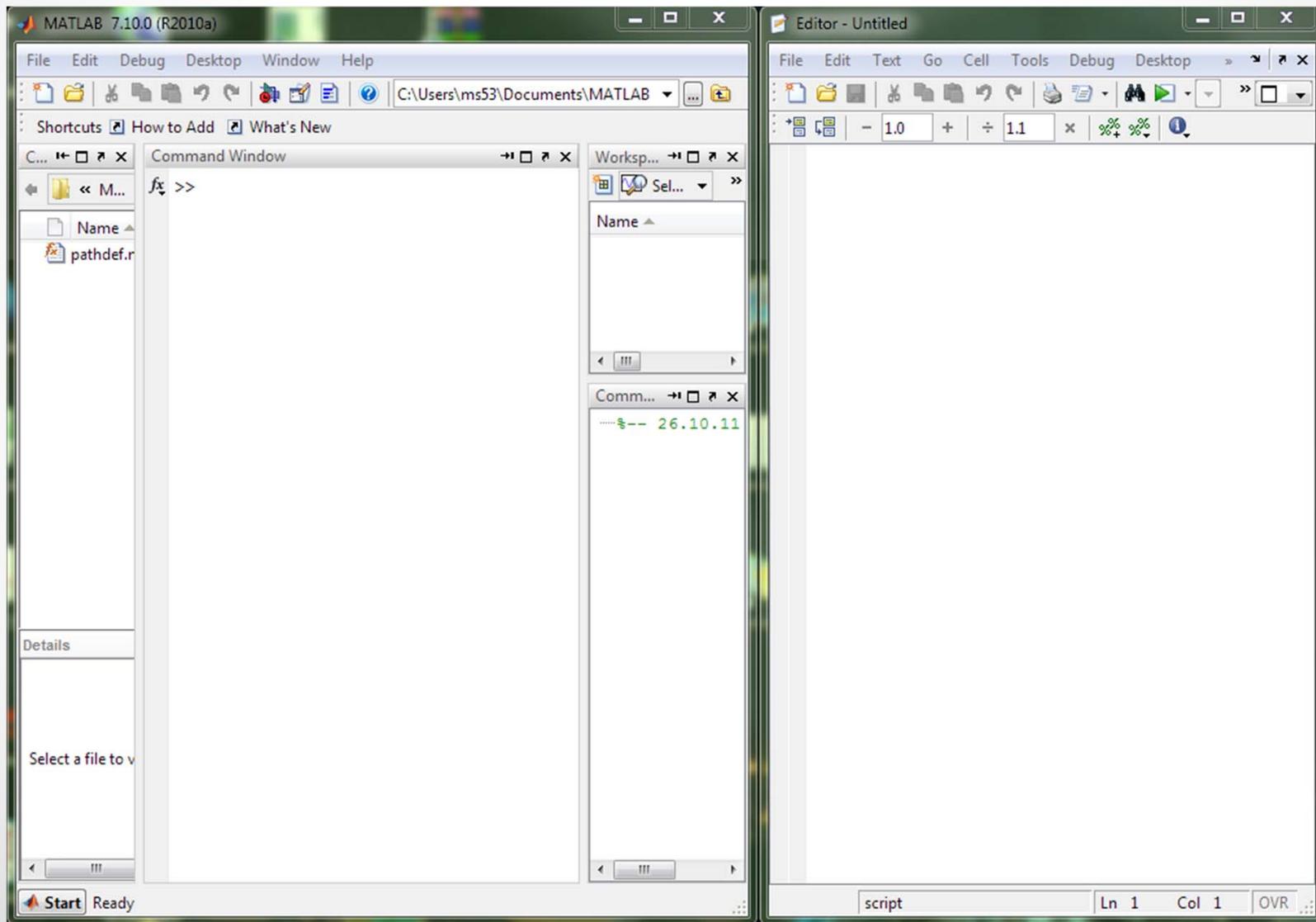
Wo bekomme ich Matlab her?

- Matlab ist in den Pools installiert.
- Es gibt eine Netzwerkversion, die von der KIZ-Seite heruntergeladen werden kann (Achtung: Serverlizenz nur intern im Uni-Netz verfügbar, oder VPN-Client)
- Es gibt eine Studentenversion für 20 €, die über das KIZ (Service Point) erworben werden kann.
- Weitere Informationen:
<http://www.uni-ulm.de/einrichtungen/kiz/it/software-betriebssysteme/software-liste/mathematische-anwendungssoftware/matlab-simulink.html>

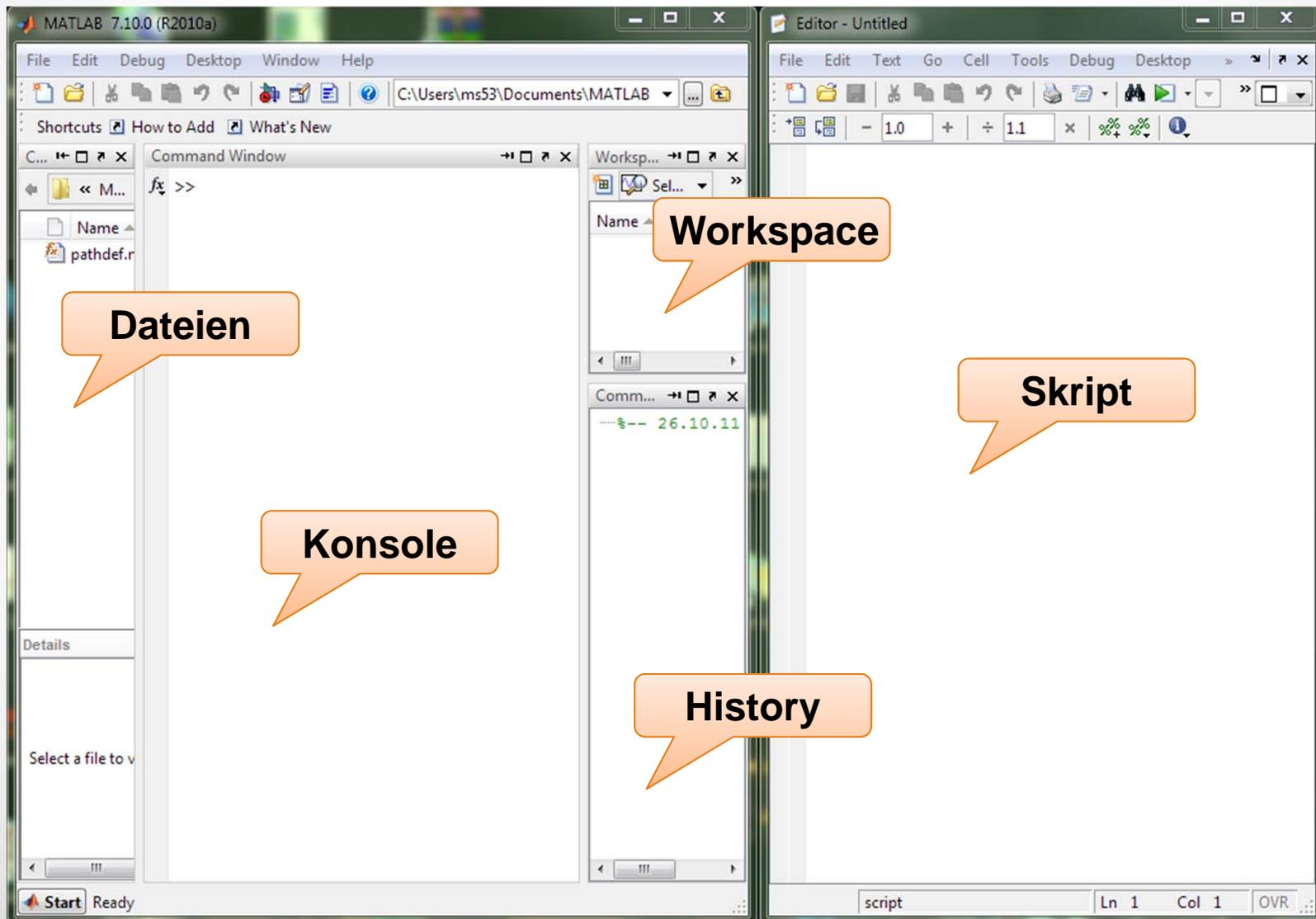
Inhalt der Einführung

- Graphical User Interface
- Matlab Konsole
- Los geht's – wie funktioniert Matlab?
- Kommandos
- Datentypen
- Indexierung
- Ablaufsteuerung
- Programmierung: Skripte und Funktionen
- Grafik
- Laden und Speichern
- Weitere nützliche Kommentare
- Debuggen

Graphical User Interface



Graphical User Interface (2)



Matlab Konsole

```
tschechne@adrenalin:~/workspace/matlab$ ./localmatlab -nojvm -nodisplay
```

```
      < M A T L A B (R) >  
      Copyright 1984-2010 The MathWorks, Inc.  
      Version 7.10.0.499 (R2010a) 32-bit (glnx86)  
      February 5, 2010
```

```
To get started, type one of these: helpwin, helpdesk, or demo.  
For product information, visit www.mathworks.com.
```

```
>> █
```

```
>> 1+1
```

```
ans =
```

```
2
```

```
>> 2^8
```

```
ans =
```

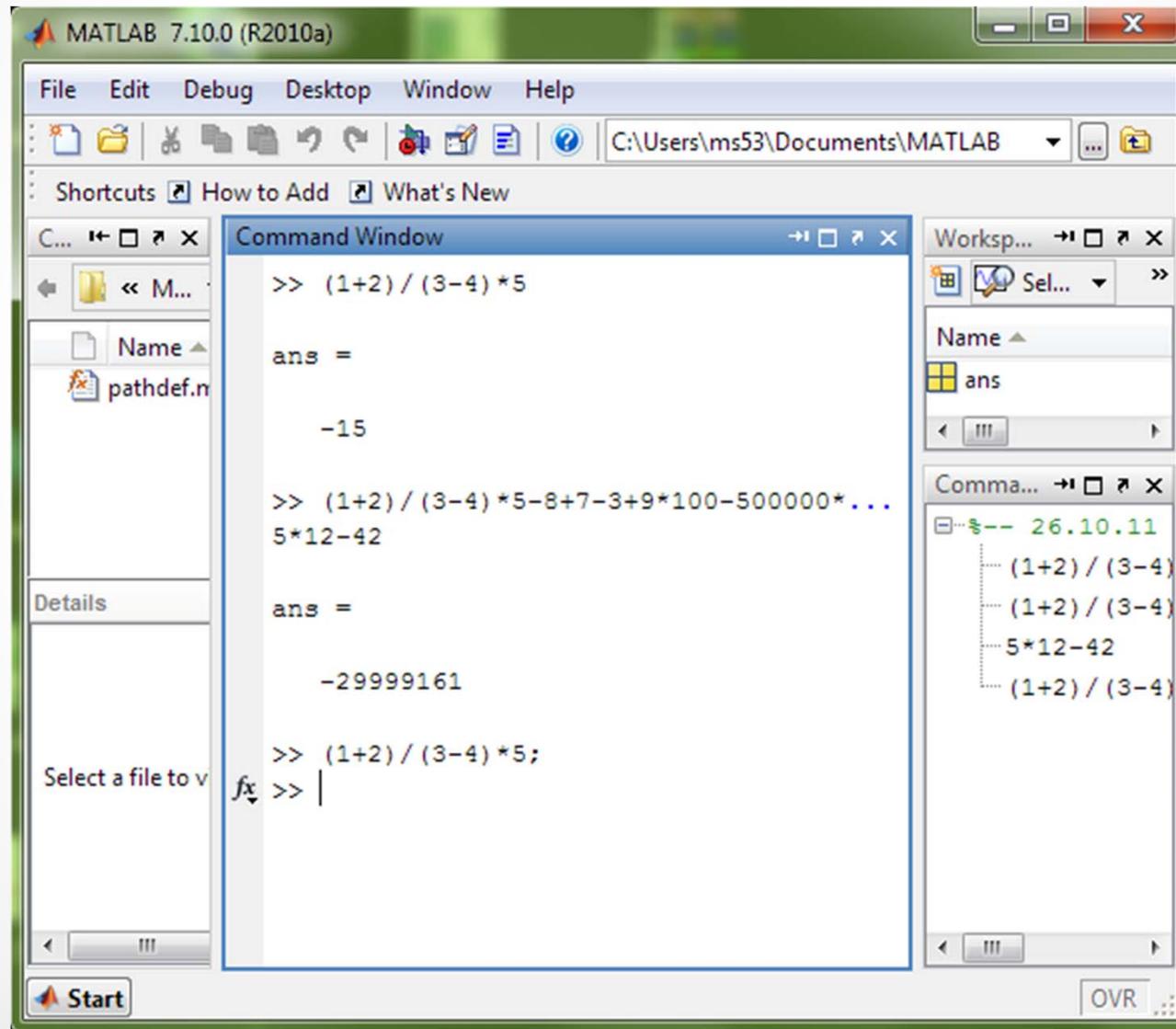
```
256
```

```
>> █
```

Los geht's – wie funktioniert Matlab?

- Im Code-Fenster können Matlab-Kommandos eingegeben werden.
- Mit „Enter“ werden diese ausgeführt.
- Reicht die Zeilenlänge nicht aus, können drei Punkte „...“ als Fortsetzungszeichen eingegeben werden.
- Auf vergangene Eingaben kann man mit den Pfeiltasten zurückgreifen.
- Nicht spezifizierte Ausgaben werden der Variable `ans` zugeordnet.
- Beendet man eine Eingabe mit einem Semikolon „;“ wird die Ausgabe unterdrückt.

Los geht's – wie funktioniert Matlab?

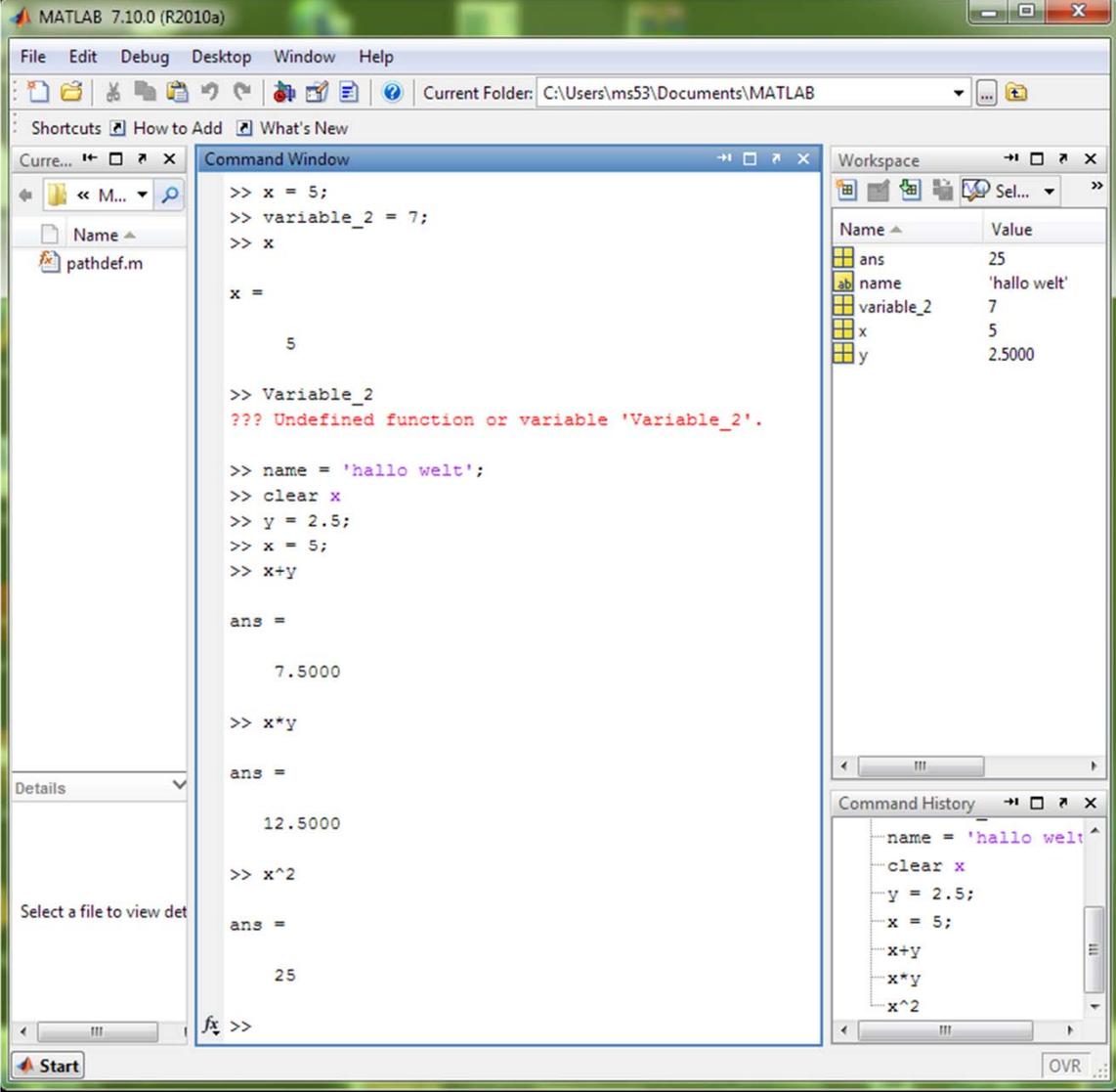


Kommandos

Kommandos: Ausdrücke, die sich aus folgenden Elementen zusammensetzen:

- Variable:
 - Zeichenkette (maximal 31 Zeichen, beginnt mit Buchstabe)
 - Zuordnung mit Gleichheitszeichen
 - Um den Wert anzuzeigen, einfach Name der Variable eingeben ... Enter
 - Groß- und Kleinschreibung beachten
 - Typ muss nicht spezifiziert werden ... Matlab erkennt den „richtigen“ Typ
 - Anzeige der Variablen im Workspace
 - Löschen/Freigeben mit: `clear variable`
- Zahl
 - Dezimalpunkt
- Operatoren
 - + Addition - Subtraktion * Multiplikation / Division ^ Exponent
 - Elementweise Operatoren durch Punktnotation (später)

Kommandos



The screenshot shows the MATLAB 7.10.0 (R2010a) Command Window with the following commands and outputs:

```
>> x = 5;
>> variable_2 = 7;
>> x

x =

    5

>> Variable_2
??? Undefined function or variable 'Variable_2'.

>> name = 'hallo welt';
>> clear x
>> y = 2.5;
>> x = 5;
>> x+y

ans =

    7.5000

>> x*y

ans =

   12.5000

>> x^2

ans =

    25
```

The Workspace window shows the following variables:

Name	Value
ans	25
name	'hallo welt'
variable_2	7
x	5
y	2.5000

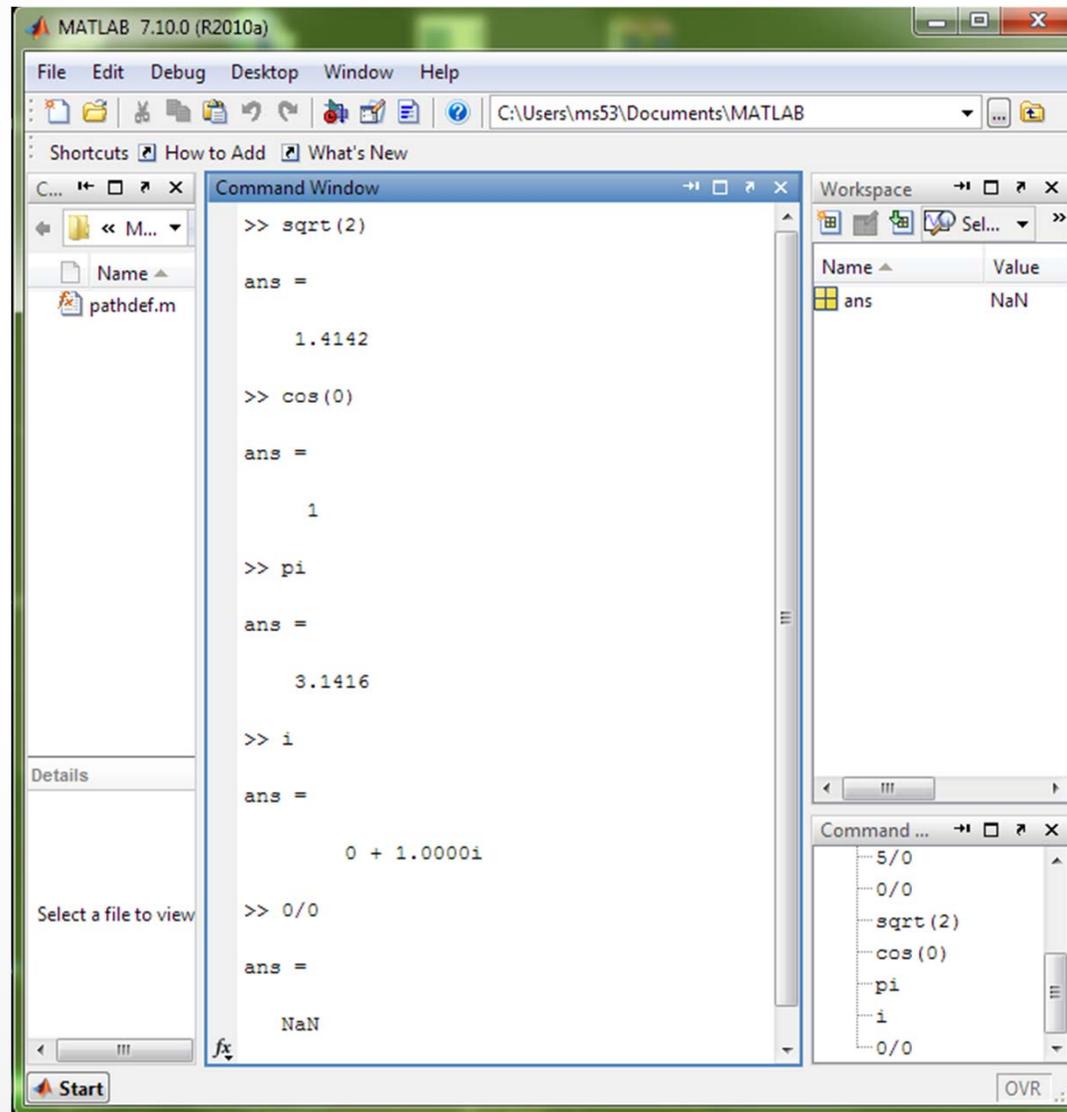
The Command History window shows the following commands:

```
name = 'hallo welt'
clear x
y = 2.5;
x = 5;
x+y
x*y
x^2
```

Kommandos (2)

- Funktionen:
 - Matlab kennt viele elementare Funktionen, z.B. `sqrt ()`, `cos`, `rand ...`
 - Eine Liste der elementaren Funktionen erhält man mit: `help elfun`
 - Eine Liste spezieller Funktionen mit: `help specfun`
 - Eine Liste besonderer Funktionen für Matrizen mit: `help elmat`
- Konstanten
 - Häufig benutzte Konstanten sind vordefiniert:
 - `pi` = 3.1415...
 - `i` = imaginärer Teil bei komplexen Zahlen
 - `inf` = unendlich
 - `nan` = mathematisch nicht definierte Zahl, z.B. `0/0 = nan`

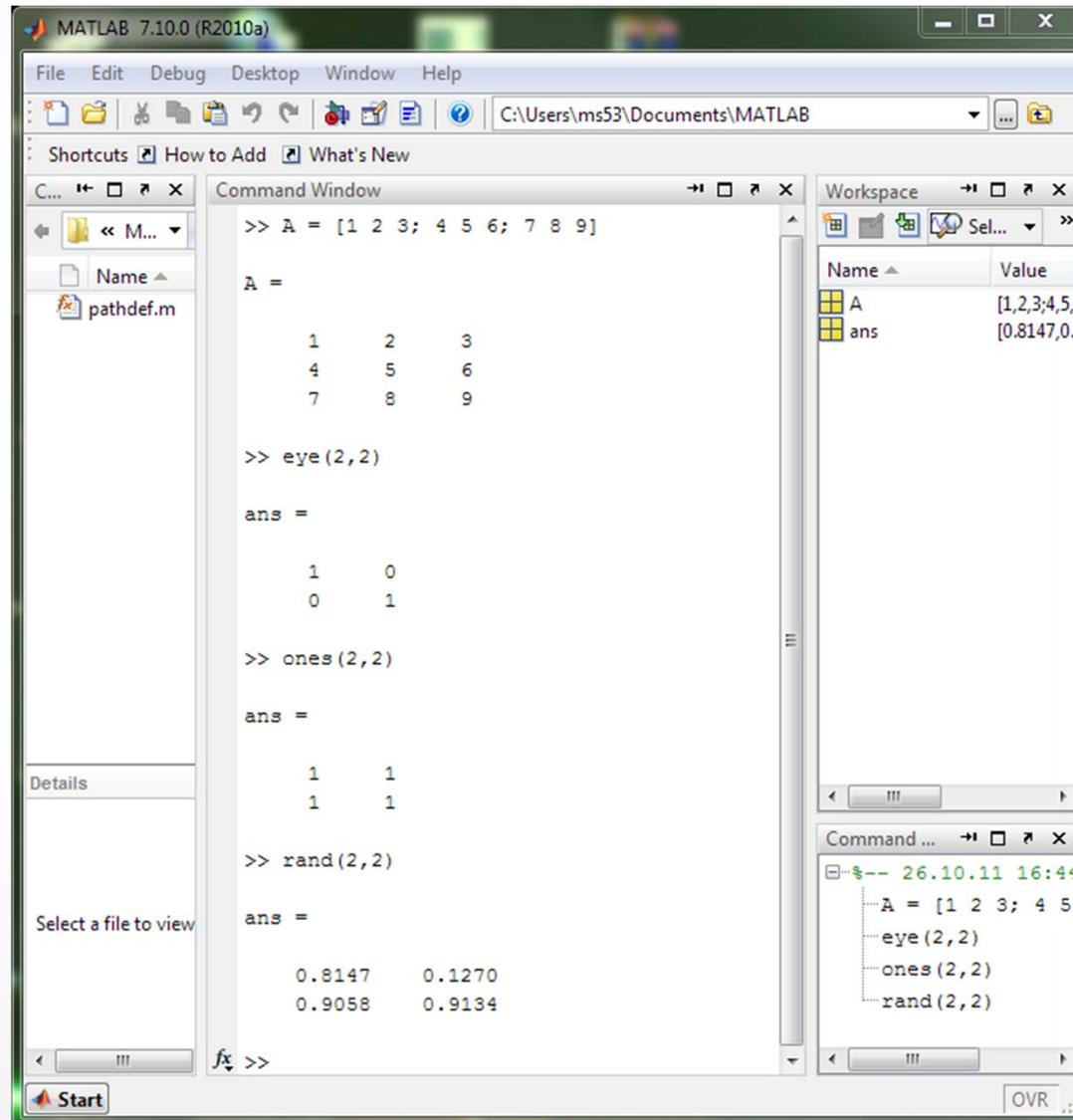
Kommandos (2)



Datentypen

- Matlab kennt sechs verschiedene Felder als Datentypen (Felder innerhalb von Matrizen):
 - numeric: double, sparse, uint8
 - char
 - cell
 - struct
- Matrizen
 - $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$
 - Zahlenwerte in eckigen Klammern
 - Elemente durch Leerzeichen oder Komma getrennt
 - Zeilen durch Semikolon „;“ getrennt
 - Spezielle Matrizen durch Funktion: `eye`, `ones`, `zeros`, `rand`, ...
 - Es gibt auch mehrdimensionale Arrays

Datentypen



The image shows the MATLAB 7.10.0 (R2010a) interface. The Command Window displays the following code and output:

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> eye(2,2)

ans =

     1     0
     0     1

>> ones(2,2)

ans =

     1     1
     1     1

>> rand(2,2)

ans =

    0.8147    0.1270
    0.9058    0.9134
```

The Workspace window shows the following variables:

Name	Value
A	[1,2,3;4,5,6;7,8,9]
ans	[0.8147,0.1270;0.9058,0.9134]

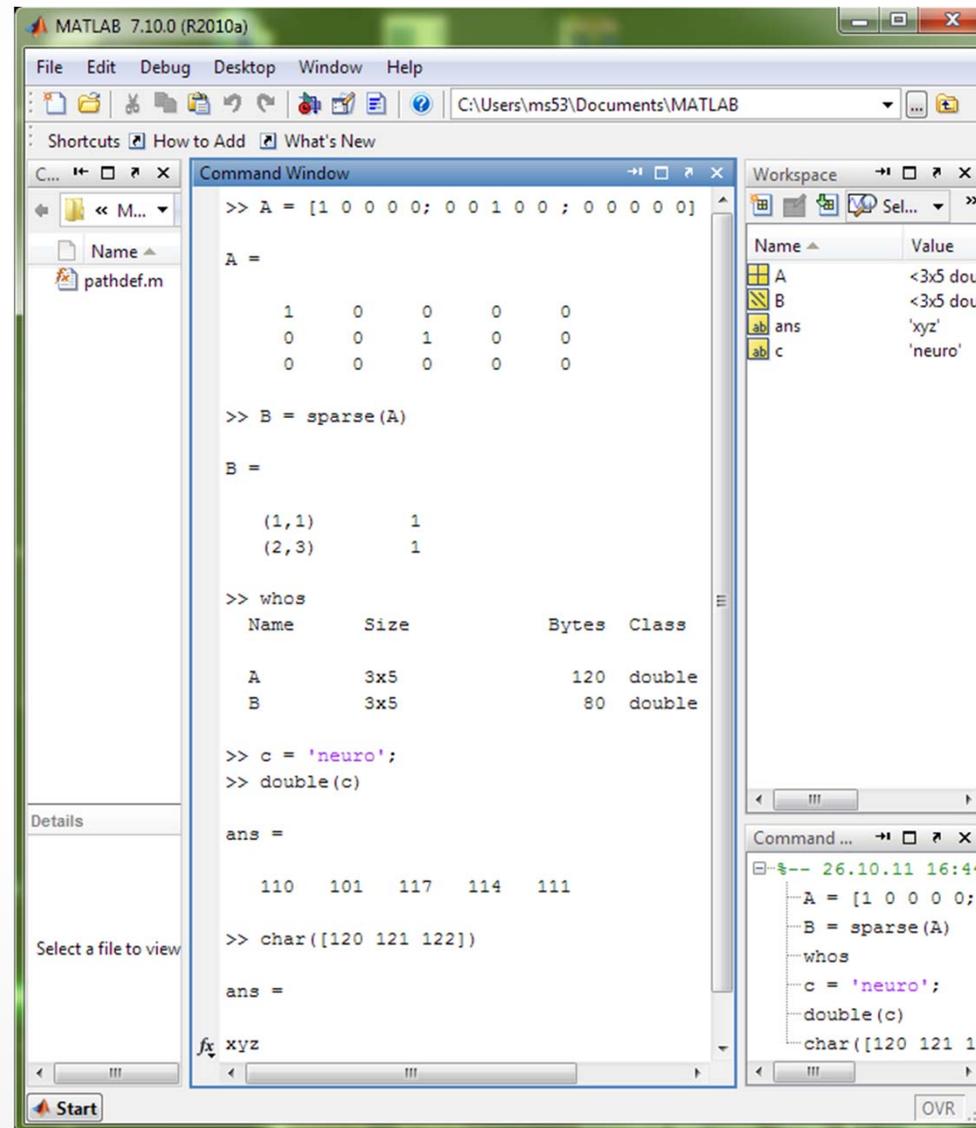
The Command Window also shows a history of commands:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
>> eye(2,2)
>> ones(2,2)
>> rand(2,2)
```

Datentypen (2)

- **Dünn besetzte Matrizen: sparse**
 - Matrizen mit sehr vielen Nullen können effizienter als Datentyp „sparse“ gespeichert werden.
 - Es werden nur die Indizes und Werte der von Null verschiedenen Einträge gespeichert.
 - Mit dem Befehl `whos` lässt sich der Speicherverbrauch anzeigen
- **Bilddaten: uint8**
 - Große 8-Bit Matrizen (Zahlenwerte von 0-255)
 - Verbraucht weniger Speicherplatz als normale Matrix
- **Zeichenketten: Strings**
 - Darstellung durch Apostrophs `'name'`
 - Einzeilige Matrizen mit internem Wert „char“
 - Numerische Werte durch Funktion „double“
 - Buchstaben durch Funktion „char“

Datentypen (2)



The image shows the MATLAB 7.10.0 (R2010a) interface. The Command Window displays the following code and output:

```
>> A = [1 0 0 0 0; 0 0 1 0 0; 0 0 0 0 0]
A =
    1     0     0     0     0
    0     0     1     0     0
    0     0     0     0     0

>> B = sparse(A)
B =
    (1,1)    1
    (2,3)    1

>> whos
Name      Size      Bytes  Class
A         3x5       120    double
B         3x5       80     double

>> c = 'neuro';
>> double(c)
ans =
    110    101    117    114    111

>> char([120 121 122])
ans =
xyz
```

The Workspace window shows the following variables:

Name	Value
A	<3x5 dou
B	<3x5 dou
ans	'xyz'
c	'neuro'

The Command Window also shows a history of the executed commands:

```
26.10.11 16:49
A = [1 0 0 0 0;
B = sparse(A)
whos
c = 'neuro';
double(c)
char([120 121 122])
```

Datentypen (3)

- **Cell:** Kombiniert verschiedene Datentypen
 - Ähnliche Definition wie bei Matrizen nur mit geschweiften Klammern „{ }“
- **Structs:**
 - Tabellen mit Feldern und enthaltenen Werten
 - `Variablenname = struct('Feld1',Wert1,'Feld2',Wert2,...)`
 - Weitere Elemente (Spalten) können mit `structname.neuesElement` hinzugefügt werden
 - Bestehende Felder können so auch abgefragt werden

Datentypen (3)

The image shows the MATLAB 7.10.0 (R2010a) interface. The Command Window displays the following code and its output:

```
>> A = [1 2 3; 4 5 6];  
>> B = 'Hello';  
>> C = {A B}  
  
C =  
  
    [2x3 double]    'Hello'  
  
>> Adresse = struct('Name','Otto','Ort','Ulm','Alter',42);  
  
Adresse =  
  
    Name: 'Otto'  
    Ort: 'Ulm'  
    Alter: 42  
  
>> Adresse.Telefon = 777777  
  
Adresse =  
  
    Name: 'Otto'  
    Ort: 'Ulm'  
    Alter: 42  
    Telefon: 777777  
  
>> Adresse.Alter  
  
ans =  
  
    42  
  
fx >>
```

The Workspace window shows the following variables and their values:

Name	Value
A	[1,2,3;4,5,6]
Adresse	<1x1 struct array>
B	'Hello'
C	<1x2 cell array>
ans	42

The Command Window also shows a history of the executed commands:

```
>> A = [1 2 3; 4 5 6];  
>> B = 'Hello';  
>> C = {A B}  
Adresse = struct('Name','Otto','Ort','Ulm','Alter',42);  
Adresse.Telefon = 777777  
Adresse.Alter  
ans = 42
```

Indexierung

- Zugriff auf Matrix-Elemente

- Zugriff durch runde Klammern
- Erster Index = Zeile, zweiter Index = Spalte $A(2, 3)$
- Mit Doppelpunkt kann auf einen ganzen Bereich zugegriffen werden
 $A(1, 1:2)$ oder auch auf ganze Spalte/Zeile $A(:, 1)$

- Zusammenfügen (concatenate)

- Angabe durch $[]$, Komma „ $,$ “ horizontal, Semikolon „ $;$ “ vertikal

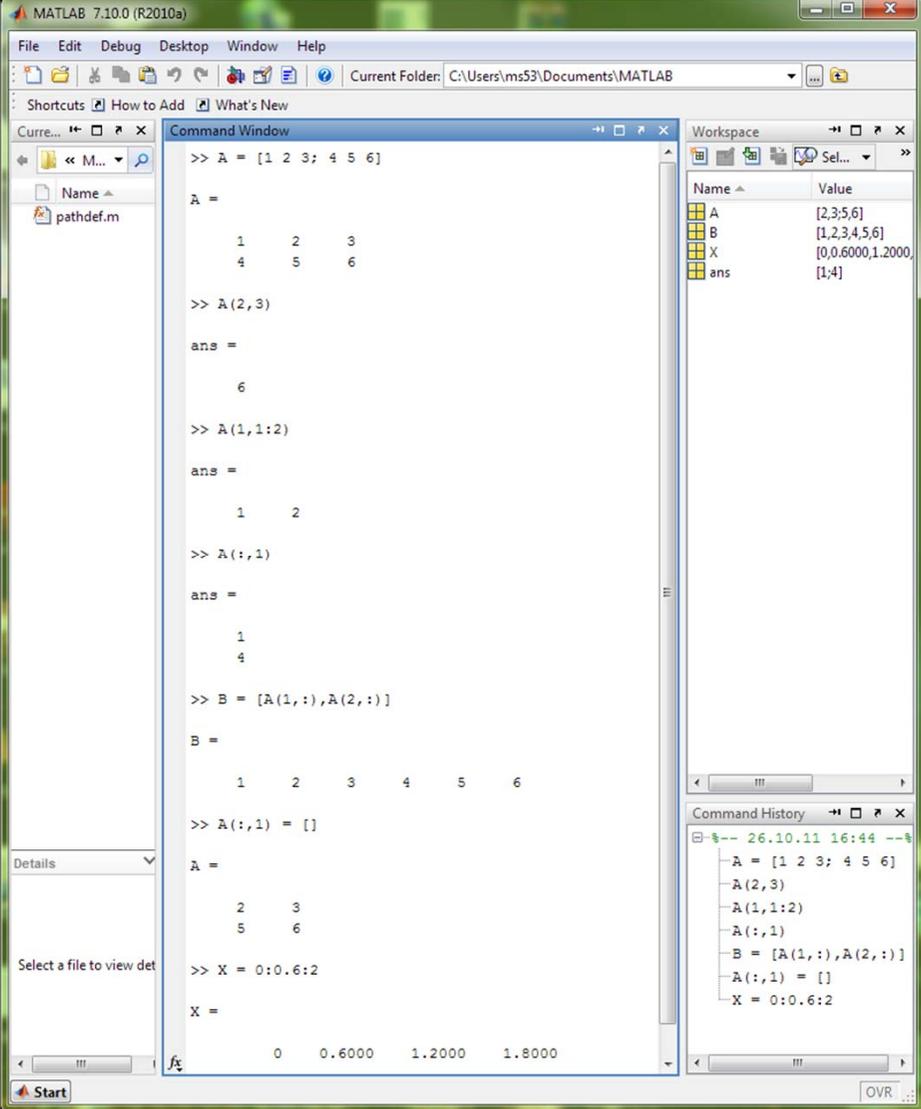
- Löschen

- Ganze Zeilen/Spalten können durch Zuweisung einer leeren Matrix gelöscht werden $A(:, 1) = []$

- Reihen erzeugen

- Mit $[Anfang:Schrittweite:Ende]$ können Reihen erstellt werden.

Indexierung



The image shows a screenshot of the MATLAB 7.10.0 (R2010a) Command Window. The window displays a series of commands and their outputs, demonstrating various indexing techniques on a 2x6 matrix A. The Workspace window on the right shows the current state of variables: A, B, X, and ans. The Command History window at the bottom shows the sequence of commands entered.

```
>> A = [1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6

>> A(2,3)
ans =
     6

>> A(1,1:2)
ans =
     1     2

>> A(:,1)
ans =
     1
     4

>> B = [A(1,:), A(2,:)]
B =
     1     2     3     4     5     6

>> A(:,1) = []
A =
     2     3
     5     6

>> X = 0:0.6:2
X =
     0    0.6000    1.2000    1.8000
```

Name	Value
A	[2,3;5,6]
B	[1,2,3,4,5,6]
X	[0,0.6000,1.2000,1.8000]
ans	[1,4]

Command History:

```
-- 26.10.11 16:44 --%
-- A = [1 2 3; 4 5 6]
-- A(2,3)
-- A(1,1:2)
-- A(:,1)
-- B = [A(1,:), A(2,:)]
-- A(:,1) = []
-- X = 0:0.6:2
```

Indexierung (2)

- Vektorisierung und Schleifen

- Um einen Vektor zu besetzen, können alle Elemente der Reihe nach gefüllt werden

- Vektor s mit Sinus-Werten von 0 bis 2π mit Schrittweite 0.6

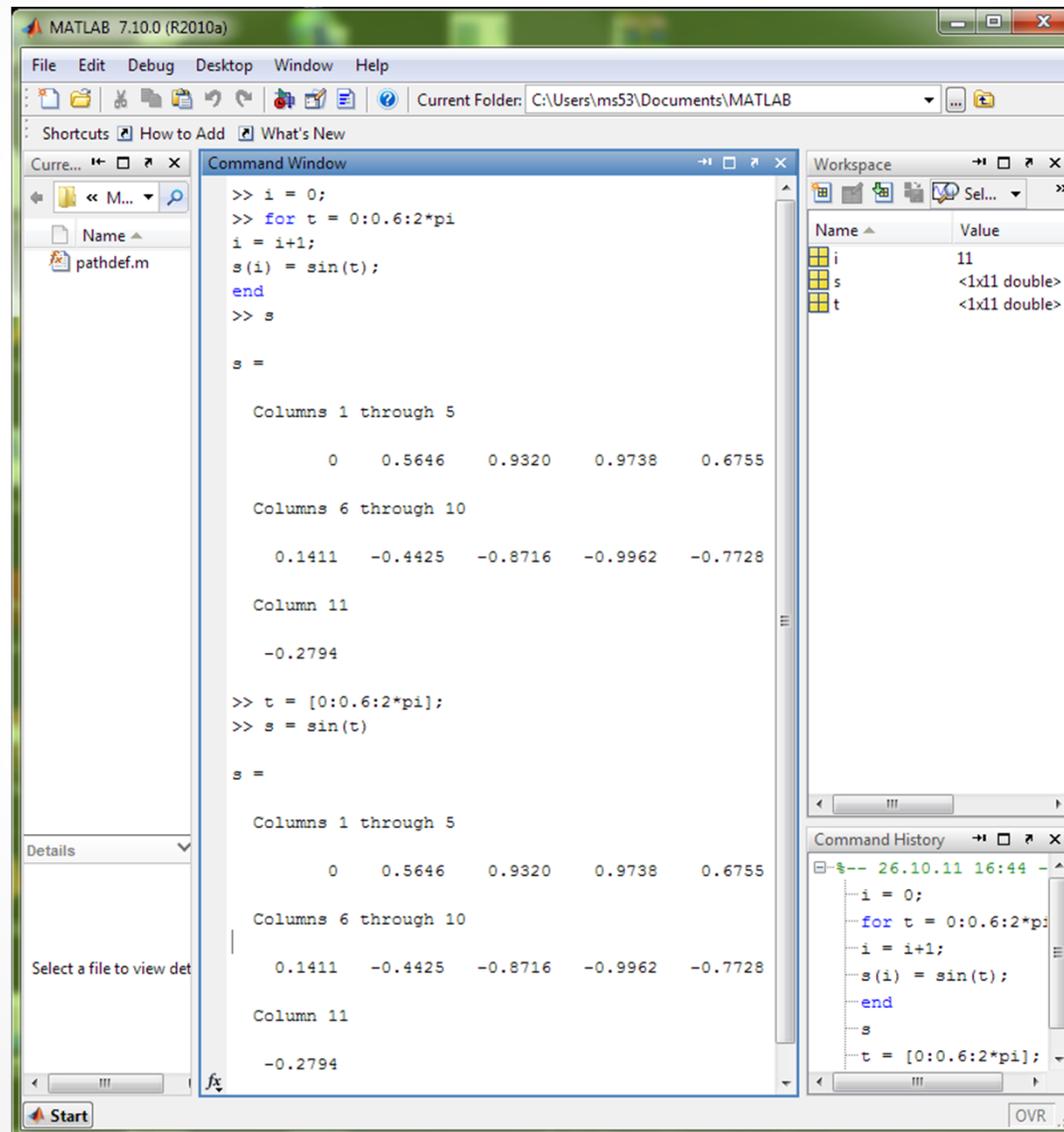
```
i = 0;  
for t = 0:0.6:2*pi  
    i = i + 1;  
    s(i) = sin(t);  
end
```

- So nicht gut! Länge des Vektors ist unbestimmt und während dem Schleifendurchlauf muss immer wieder neuer Speicherplatz geschaffen werden.

- Viele Funktionen arbeiten auf ganzen Vektoren, daher besser:

```
t = [0:0.6:2*pi];  
s = sin(t);
```

Indexierung (2)



The screenshot shows the MATLAB 7.10.0 (R2010a) interface. The Command Window displays the following code and output:

```
>> i = 0;
>> for t = 0:0.6:2*pi
    i = i+1;
    s(i) = sin(t);
end
>> s

s =

Columns 1 through 5
    0    0.5646    0.9320    0.9738    0.6755

Columns 6 through 10
    0.1411   -0.4425   -0.8716   -0.9962   -0.7728

Column 11
   -0.2794

>> t = [0:0.6:2*pi];
>> s = sin(t)

s =

Columns 1 through 5
    0    0.5646    0.9320    0.9738    0.6755

Columns 6 through 10
    0.1411   -0.4425   -0.8716   -0.9962   -0.7728

Column 11
   -0.2794
```

The Workspace window shows the following variables:

Name	Value
i	11
s	<1x11 double>
t	<1x11 double>

The Command History window shows the following commands:

```
-- 26.10.11 16:44 --
>> i = 0;
>> for t = 0:0.6:2*pi;
>>     i = i+1;
>>     s(i) = sin(t);
>> end
>> s
>> t = [0:0.6:2*pi];
```

Indexierung (3)

- Logische Indexierung

- Es gibt viele Funktionen, die auf Eigenschaften der Einträge eingehen, z.B. `isfinite` (entfernt NaN aus Matrix):

```
A = [-3:3];
```

```
B = A./A (Elementweiser Operator)
```

```
B(isfinite(B))
```

- Aus einer zufälligen normalverteilten 5x5 Matrix alle Elemente auf Null setzen, die größer als 0 sind:

```
A = randn(5,5)
```

```
A(A>0)=1
```

- Operator auf Elemente in Matrizen

- `*` und `./` elementweise Multiplikationen und Division
- `.^` elementweise Exponentiation

Indexierung (3)

The screenshot shows the MATLAB 7.10.0 (R2010a) interface. The Command Window contains the following code and output:

```
>> A = [-3:3];
>> B = A./A

B =

     1     1     1   NaN     1     1     1

>> B(isfinite(B))

ans =

     1     1     1     1     1     1

>> A = randn(5,5)

A =

 -0.0068    1.1174    1.5442    2.3505   -0.7648
  1.5326   -1.0891    0.0859   -0.6156   -1.4023
 -0.7697    0.0326   -1.4916    0.7481   -1.4224
  0.3714    0.5525   -0.7423   -0.1924    0.4882
 -0.2256    1.1006   -1.0616    0.8886   -0.1774

>> A(A>0) = 0

A =

 -0.0068         0         0         0   -0.7648
         0   -1.0891         0   -0.6156   -1.4023
 -0.7697         0   -1.4916         0   -1.4224
         0         0   -0.7423   -0.1924         0
 -0.2256         0   -1.0616         0   -0.1774
```

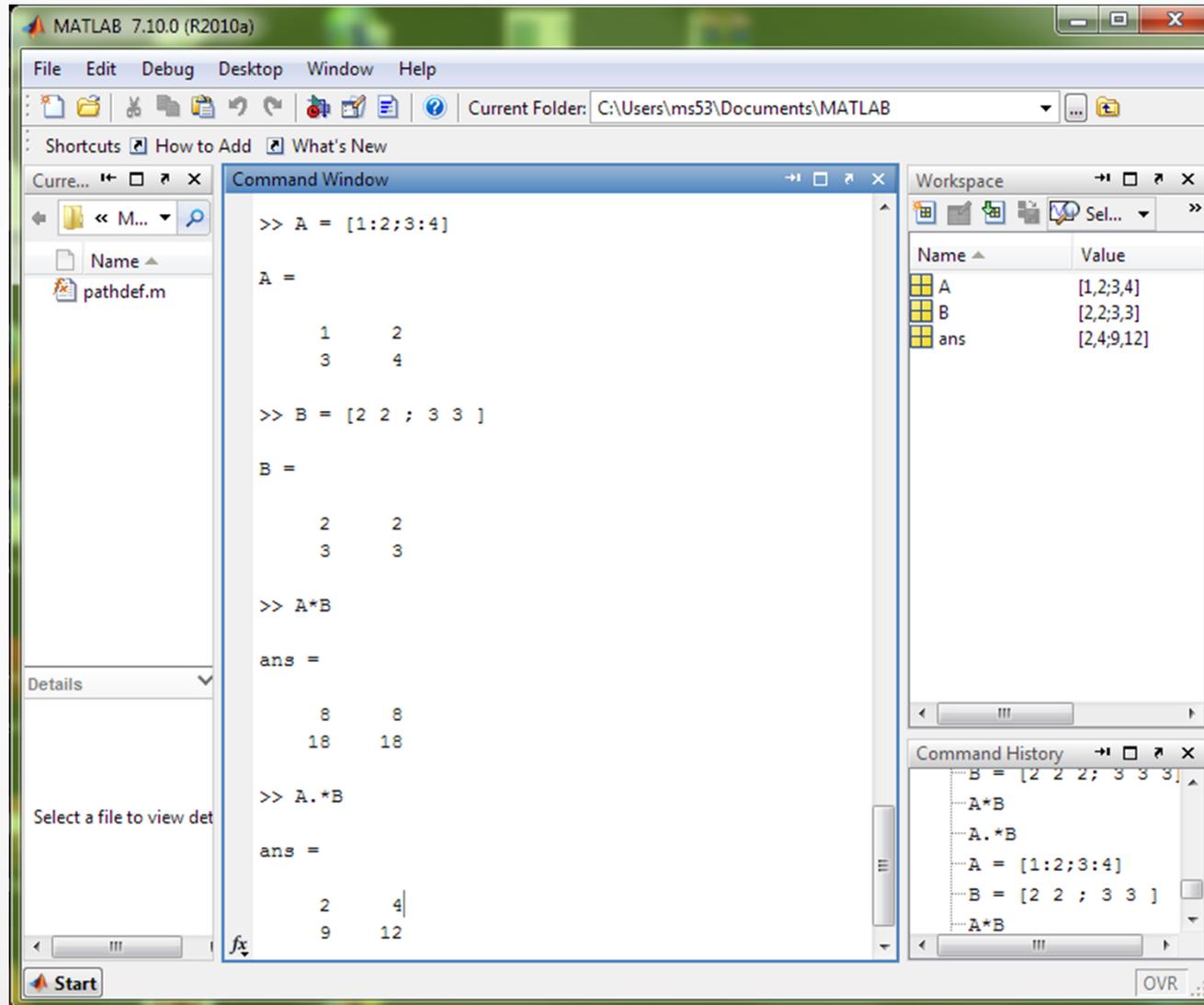
The Workspace window shows the following variables:

Name	Value
A	<5x5 double>
B	[1,1,NaN,1,1,1]
ans	[1,1,1,1,1]

The Command History window shows the following commands:

```
A = randn(5,5)
A(A>0) = 1
A(A>0) = 0
A = [-3:3];
B = A./A
B(isfinite(B))
A = randn(5,5)
A(A>0) = 0
```

Indexierung (3)



The screenshot shows the MATLAB 7.10.0 (R2010a) interface. The Command Window displays the following code and output:

```
>> A = [1:2;3:4]

A =

     1     2
     3     4

>> B = [2 2 ; 3 3 ]

B =

     2     2
     3     3

>> A*B

ans =

     8     8
    18    18

>> A.*B

ans =

     2     4
     9    12
```

The Workspace window shows the following variables and their values:

Name	Value
A	[1,2;3,4]
B	[2,2;3,3]
ans	[2,4;9,12]

The Command History window shows the following commands and their outputs:

```
B = [2 2 ; 3 3 ]
A*B
A.*B
A = [1:2;3:4]
B = [2 2 ; 3 3 ]
A*B
```

Programmierung

- Es gibt zwei Arten von Matlab-Programmen
 - Skripte
 - Funktionen
- Skripte
 - sind Textdateien, die einfach zeilenweise eine Folge von Matlab-Befehlen enthalten.
 - mit irgendeinem Editor erstellbar, oder in Matlab integrierten Editor verwenden.
 - werden als .m-Dateien abgespeichert.
 - kann man ausführen, indem man in der Konsole einfach den Dateinamen (ohne .m) eingibt.
 - müssen im Suchpfad enthalten sein.
 - Man kann keine Parameter übergeben.
 - Variablen werden im Workspace abgelegt.

Programmierung

The screenshot displays the MATLAB 7.10.0 (R2010a) environment. The main window is split into two panes. The left pane contains the Command Window and the Workspace. The Command Window shows the execution of the script 'BeispielSkript', with the prompt '>>' and a small 'fx' icon. The Workspace shows three variables: 'a' with value 5, 'b' with value 7, and 'c' with value 12. The right pane is the Editor, showing the script 'BeispielSkript.m' with the following code:

```
1 - a = 5;  
2 - b = 7;  
3  
4 - c = a+b;
```

The status bar at the bottom of the Editor shows 'Ln 4 Col 9'. The taskbar at the bottom of the screen shows the Start button and the 'OVR' (Overwrite) indicator.

Programmierung (2)

- Funktionen
 - werden wie Skripte erzeugt, gespeichert und aufgerufen.
 - können Parameter übergeben werden.
 - können Parameter zurückgeben.
 - erzeugen lokale Variablen, die nicht im Workspace abgelegt werden.
- Hilfezeile: Die ersten Zeilen der Funktion sollten aus einem Kommentar bestehen, der den Namen der Funktion enthält und eine kurze Beschreibung der Funktionsweise.
- Kommentare werden mit vorangestelltem „%“ erzeugt.
- Unterfunktionen können angefügt werden, die aber nur innerhalb der eigentlichen Funktion benutzbar sind.

Programmierung (2)

The screenshot displays the MATLAB 7.10.0 (R2010a) environment. The left window shows the Command Window with the following execution steps:

```
>> BeispielSkript
>> c = BeispielFunktion(a,b)

c =

    12

>> [c,i] = BeispielFunktion(a,b)

c =

    12

i =

     1

>> help BeispielFunktion
BeispielFunktion(a,b)
berechnet die Summe der Zahlen a u
```

The Workspace window shows the following variables:

Name	Value
a	5
b	7
c	12
i	1

The right window shows the Editor with the following code for `BeispielFunktion.m`:

```
1 % BeispielFunktion(a,b)
2 % berechnet die Summe der Zahlen a und b
3 function [c] = BeispielFunktion(a,b)
4     c = a+b;
5     i = 1; %i wird nicht zurückgeben
6 end

9 % BeispielFunktion(a,b)
10 % berechnet die Summe der Zahlen a und b
11 %function [c,i] = BeispielFunktion(a,b)
12 %c = a+b;
13 %i = 1; %i wird zurückgeben
14 %end
15
```

Programmierung (3)

- Flusskontrolle durch:

- o `if i == 1 ... end; (elseif, else)`

- o `switch i ...`

- case 1 ...

- case 2 ...

- otherwise ...

- end;

- o `for i=1:10 ... [break] end;`

- o `while i > 0 ... [break] end;`

Programmierung (3)

The image displays the MATLAB 7.10.0 (R2010a) environment. The left window shows the Command Window with the following execution steps:

```
>> x = 5;  
>> o = sigpi(x)  
  
o =  
  
    1  
  
>> x = pi;  
>> o = sigpi(x)  
  
o =  
  
    0  
  
>> x = 1;  
>> o = sigpi(x)  
  
o =  
  
   -1
```

The right window shows the Editor with the code for the `sigpi.m` function:

```
1 %sigpi(x)  
2 %sigsum funktion: gibt 1 aus wenn x > pi  
3 %0 wenn x = pi und -1 wenn x < pi  
4  
5 function output = sigpi(x)  
6     if x > pi  
7         output = 1;  
8     elseif x == pi  
9         output = 0;  
10    else  
11        output = -1;  
12    end  
13 end  
14
```

The Command Window also shows a partial view of the function call history:

```
x = pi;  
o = sigpi(:  
x = 1;  
o = sigpi(:
```

Programmierung (3)

The image shows the MATLAB 7.10.0 (R2010a) environment. The left window displays the Command Window with the following code and output:

```
>> x = 1;
>> for i = 1:5
    x = x+i;
end
>> x
x =
    16
>> n = 5;
>> while n > 0
    disp(n)
    n = n-1;
end
5
4
3
2
1
```

The right window shows the Editor with the following code:

```
113 %% Folie: Programmierung (3)
114 % addiert nacheinander die Zahlen 1-5
115 x = 1;
116 for i = 1:5
117     x = x+i;
118 end
119
120 n = 5;
121 while n > 0
122     disp(n)
123     n = n-1;
124 end
125
126
127
128
129
130
131
132
133
134
135
```

The Command Window also shows a preview of the while loop code:

```
while n > 0
disp(n)
n = n-1;
end
```

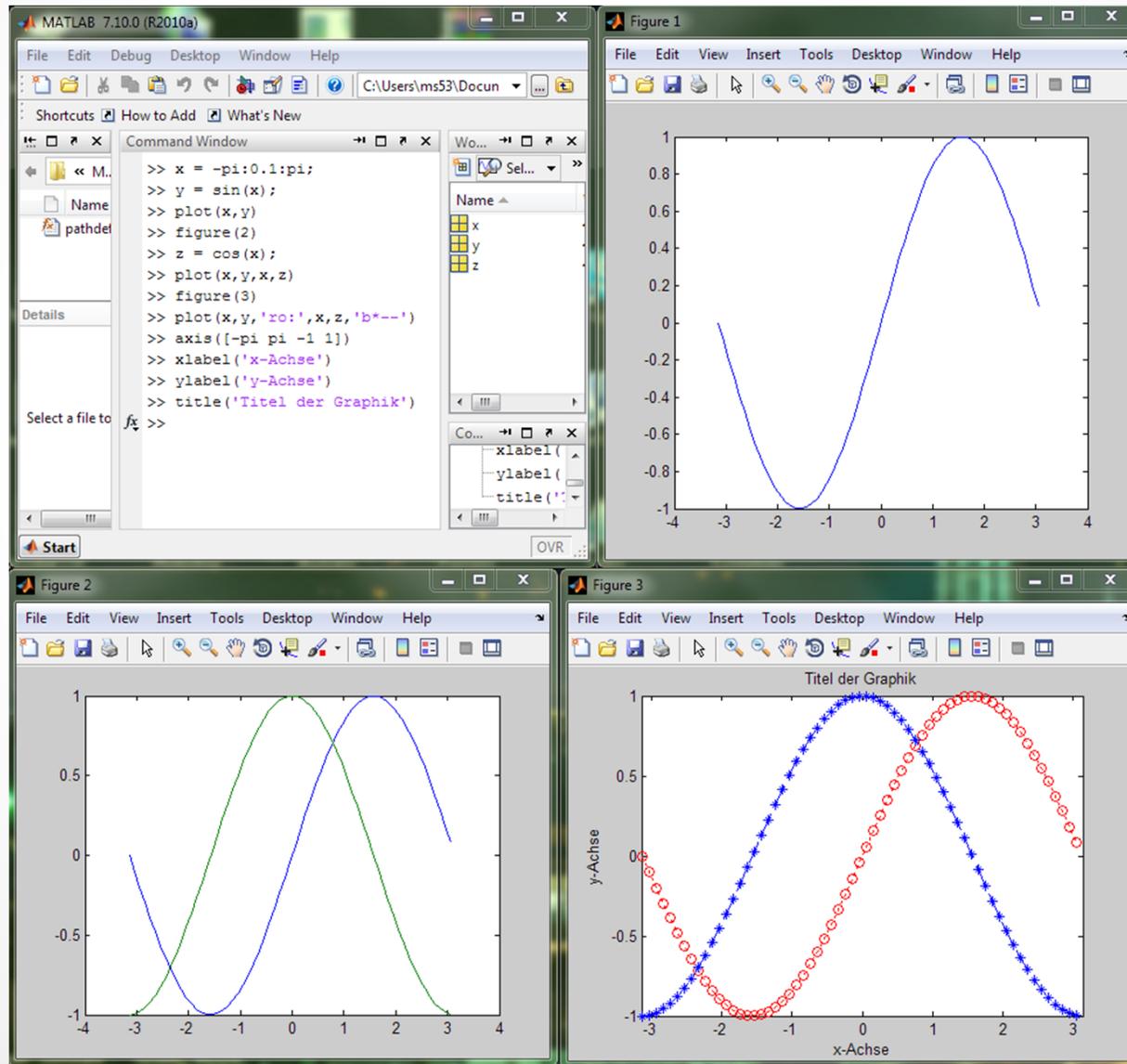
The workspace window shows the following variables:

Name	Value
i	5
n	0
x	16

Grafik

- Grafiken werden in Matlab in einem extra Fenster angezeigt.
- Dieses Fenster kann mit `figure(nummer)` angewählt werden.
- Einfacher 2D-Plot mit `plot`
 - Linienarten, z.B.: „-“ oder „-.“
 - Farben, z.B.: „c“ für cyan oder „r“ für rot
 - Symbole für Punkte, z.B.: „+“ oder „o“
 - Achsenbereiche mit `axis([minx maxx miny maxy])`
 - Achsenbeschriftungen mit `xlabel('name')` und `ylabel('name')`
 - Titel mit `title('name')`

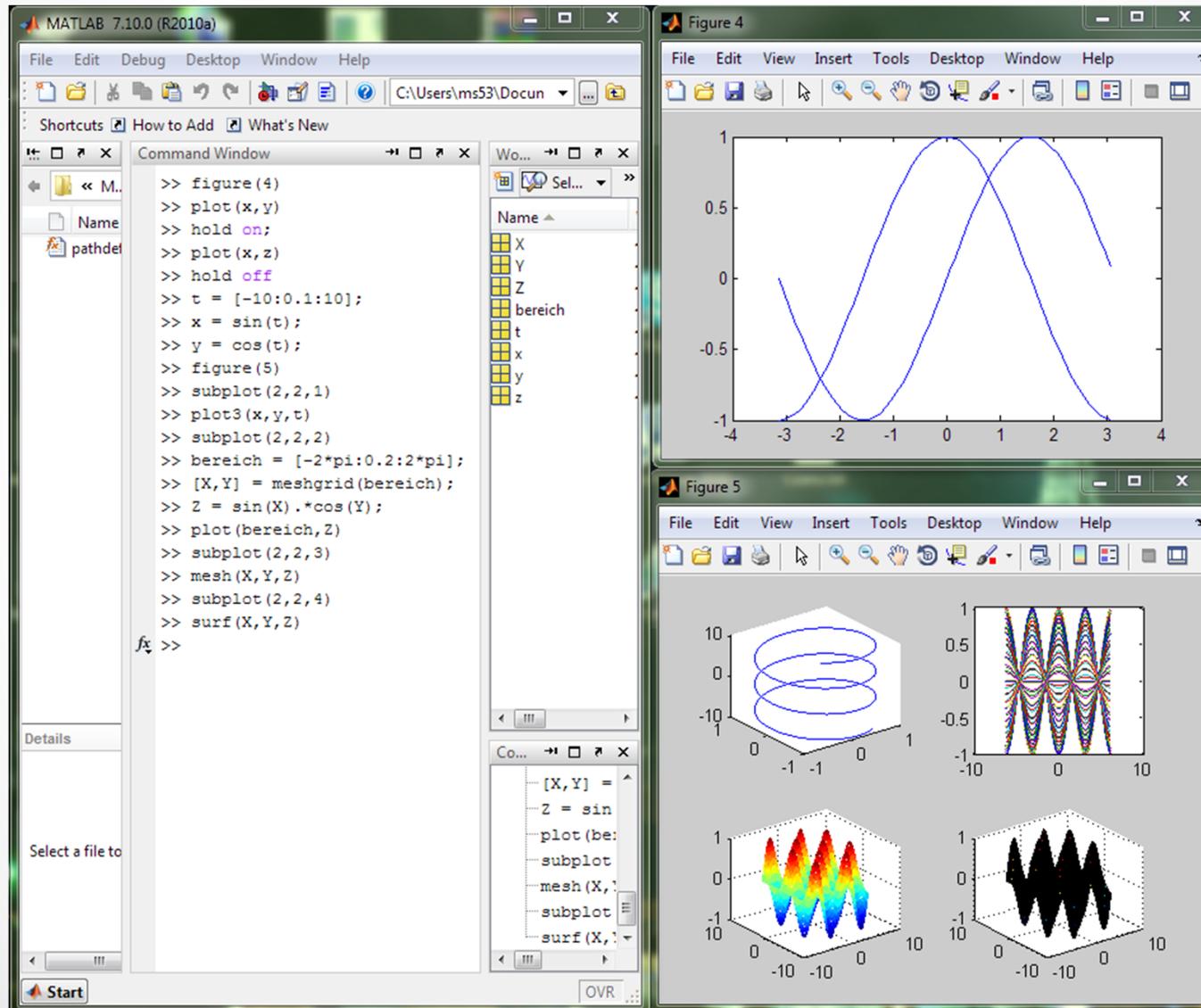
Grafik



Grafik (2)

- Fenster wird bei neuem Plot überschrieben. Um in ein Fenster mehrere Funktionen zu plotten, kann das Fenster mit `hold on;` festgehalten werden. `hold off` lässt es wieder los.
- Plots können auch in ein Fenster als Subplot geplottet werden:
`subplot(#rows,#cols,no.);`
- Einfache 3D-Plot mit `plot3`
- Gitterlinien-Plot mit `mesh`
- (Ober-)flächen-Plot mit `surf`

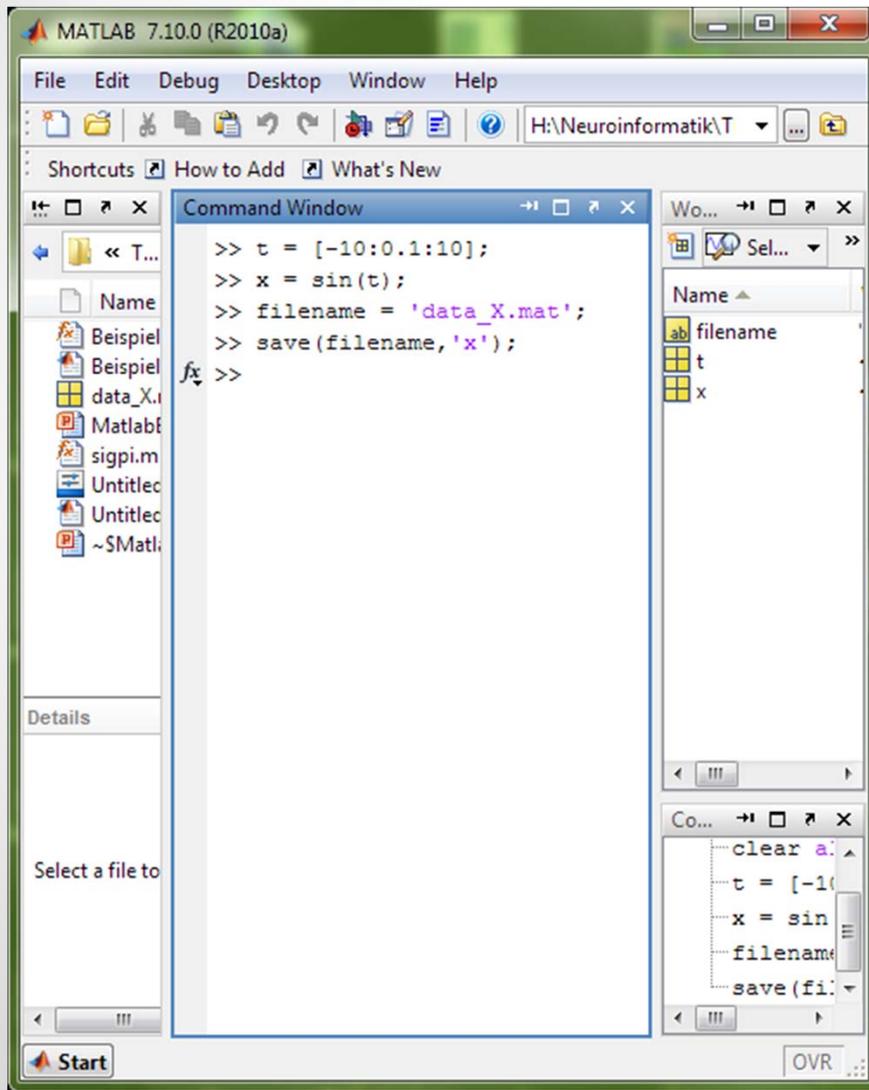
Grafik (2)



Laden und Speichern

- Daten des Workspaces können mit `save(filename, Variable)` gespeichert werden.
- `filename` kann aus einem Pfad, dem Namen und der `.mat` Endung zusammengesetzt werden.
- Geladen werden die Daten mit `load(filename)`
- Weitere wichtige Funktionen:
 - `xlsread` und `xlswrite`
 - `csvread` und `csvwrite`
 - `textread`

Laden und Speichern



The screenshot shows the MATLAB 7.10.0 (R2010a) Command Window. The Command Window contains the following code:

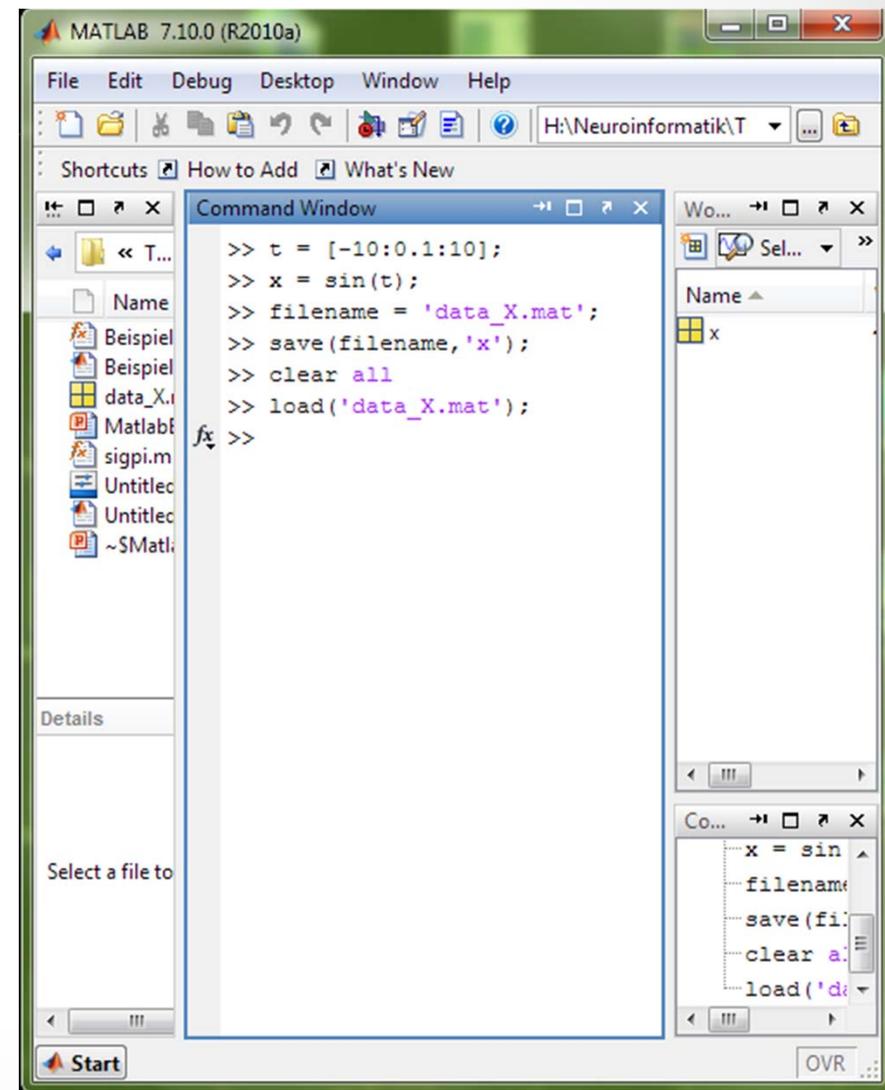
```
>> t = [-10:0.1:10];  
>> x = sin(t);  
>> filename = 'data_X.mat';  
>> save(filename, 'x');
```

The Workspace window on the right shows the following variables:

Name
filename
t
x

The Command Window also shows a scrollable list of commands, including:

```
clear a...  
t = [-1...  
x = sin...  
filename...  
save(fi...
```



The screenshot shows the MATLAB 7.10.0 (R2010a) Command Window. The Command Window contains the following code:

```
>> t = [-10:0.1:10];  
>> x = sin(t);  
>> filename = 'data_X.mat';  
>> save(filename, 'x');  
>> clear all  
>> load('data_X.mat');
```

The Workspace window on the right shows the following variables:

Name
x

The Command Window also shows a scrollable list of commands, including:

```
x = sin...  
filename...  
save(fi...  
clear a...  
load('d...
```

Weitere nützliche Kommentare

- `clc` – Konsole aufräumen
- `close all` – alle Fenster schließen
- `clear variable` – Variable aus Workspace löschen
- `clear all` – alle Variablen aus Workspace löschen

Debuggen

- Man kann an jeder Stelle in einem Skript oder einer Funktion einen Breakpoint setzen, indem man auf den Rand des Editors klickt.
- Beim Ausführen des Skripts oder der Funktion stoppt Matlab an dieser Stelle (vor dem Ausführen).
- Mit `Step` (Windows F10) kann man schrittweise weitergehen.
- Mit `Step in` (Windows F11) kann man in eine Funktion „absteigen“.
- Mit `Run` (Windows F5) lässt man das Programm weiterlaufen.

Debuggen

The image shows a MATLAB 7.10.0 (R2010a) interface with two windows open: the Command Window and the Editor.

Command Window:

```
>> %fehler im programm
>> o = sigpi(pi)
??? Undefined function or
variable 'y'.

Error in ==> sigpi at 9
        output = y;

>> o = sigpi(pi)
6         if x > pi
fx K>>
```

Editor: H:\Neuroinformatik\Theorie neuronaler Netz...

```
1 %sigpi(x)
2 %sigmum funktion: gibt 1 aus wenn x > pi
3 %0 wenn x = pi und -1 wenn x < pi
4
5 function output = sigpi(x)
6     if x > pi
7         output = 1;
8     elseif x == pi
9         output = y;
10    else
11        output = -1;
12    end
13 end
14
```

Status Bar: Start Stopped in debugger