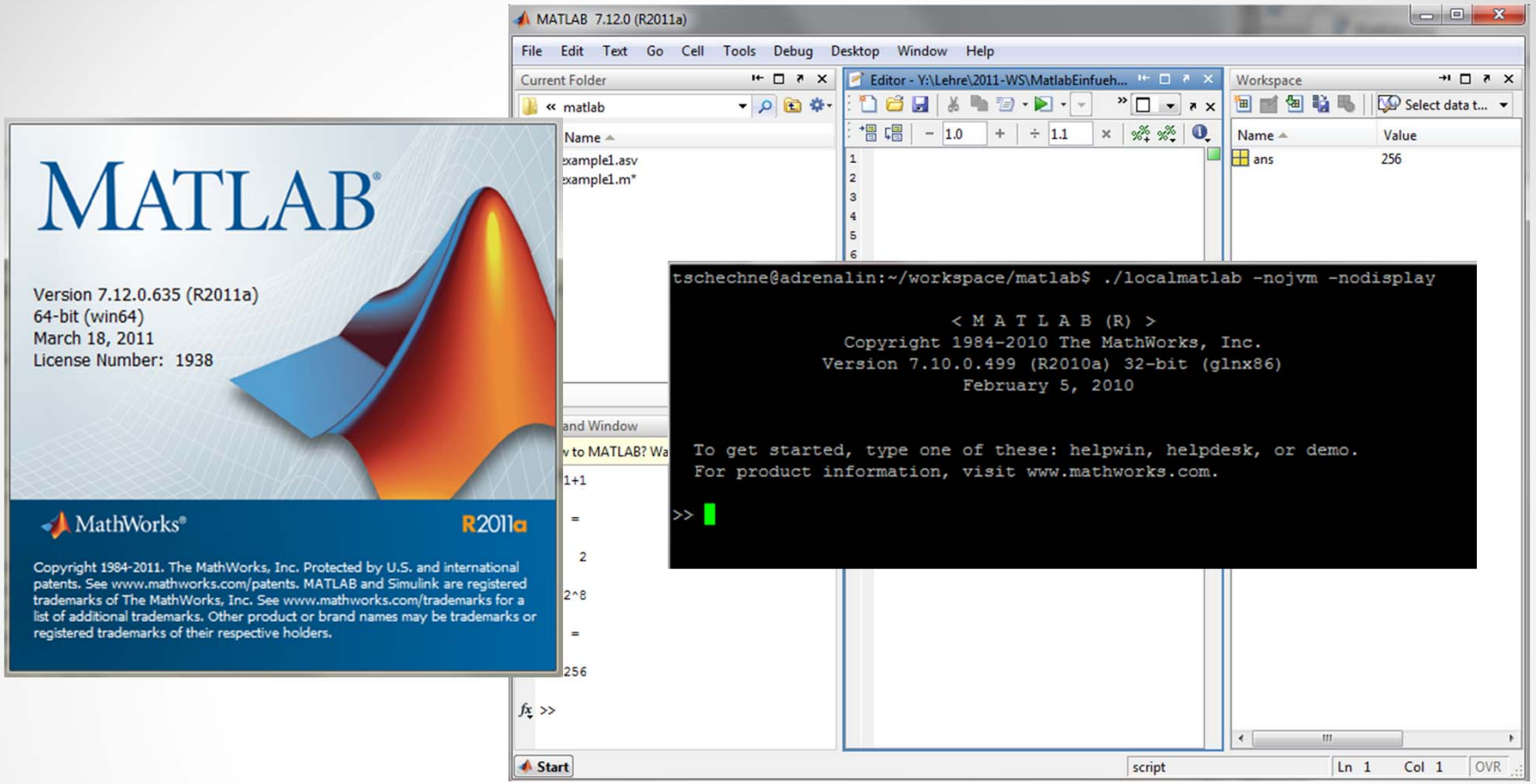


Matlab – Einführung

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?

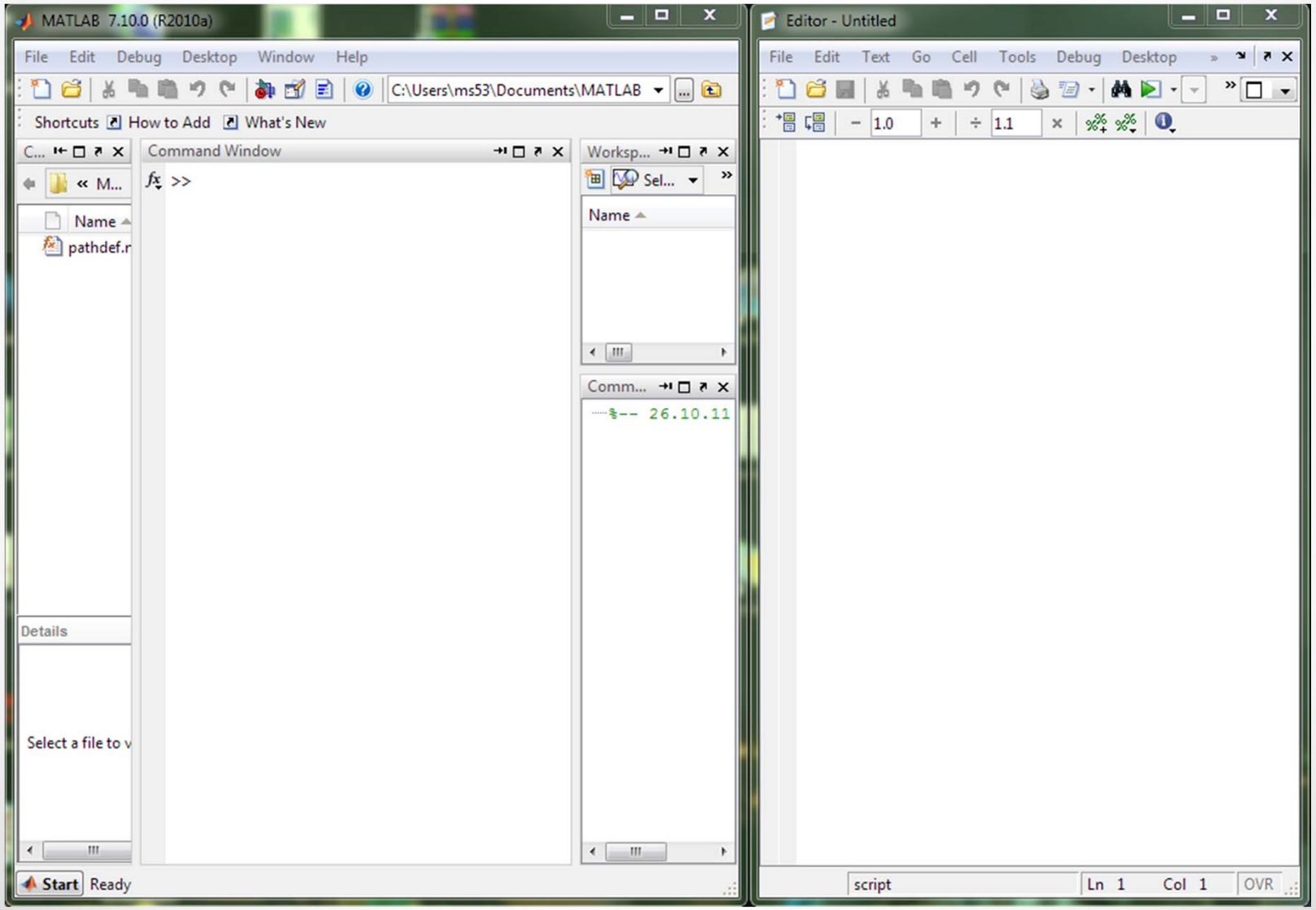
- Matlab ist in den Pools installiert.
- Es gibt eine Netzwerkversion, die von der KIZ-Seite runtergeladen werden kann (Achtung: Serverlizenz nur intern im Uni-Netz verfügbar, oder VPN-Client)
- Es gibt eine Studentenversion für 20 €, die über das KIZ (Service Point) erworben werden kann.
- Weitere Informationen:

  http://www.uni-ulm.de/einrichtungen/kiz/it/software-betriebssysteme/software-liste/mathematische-anwendungssoftware/matlabsimulink.html

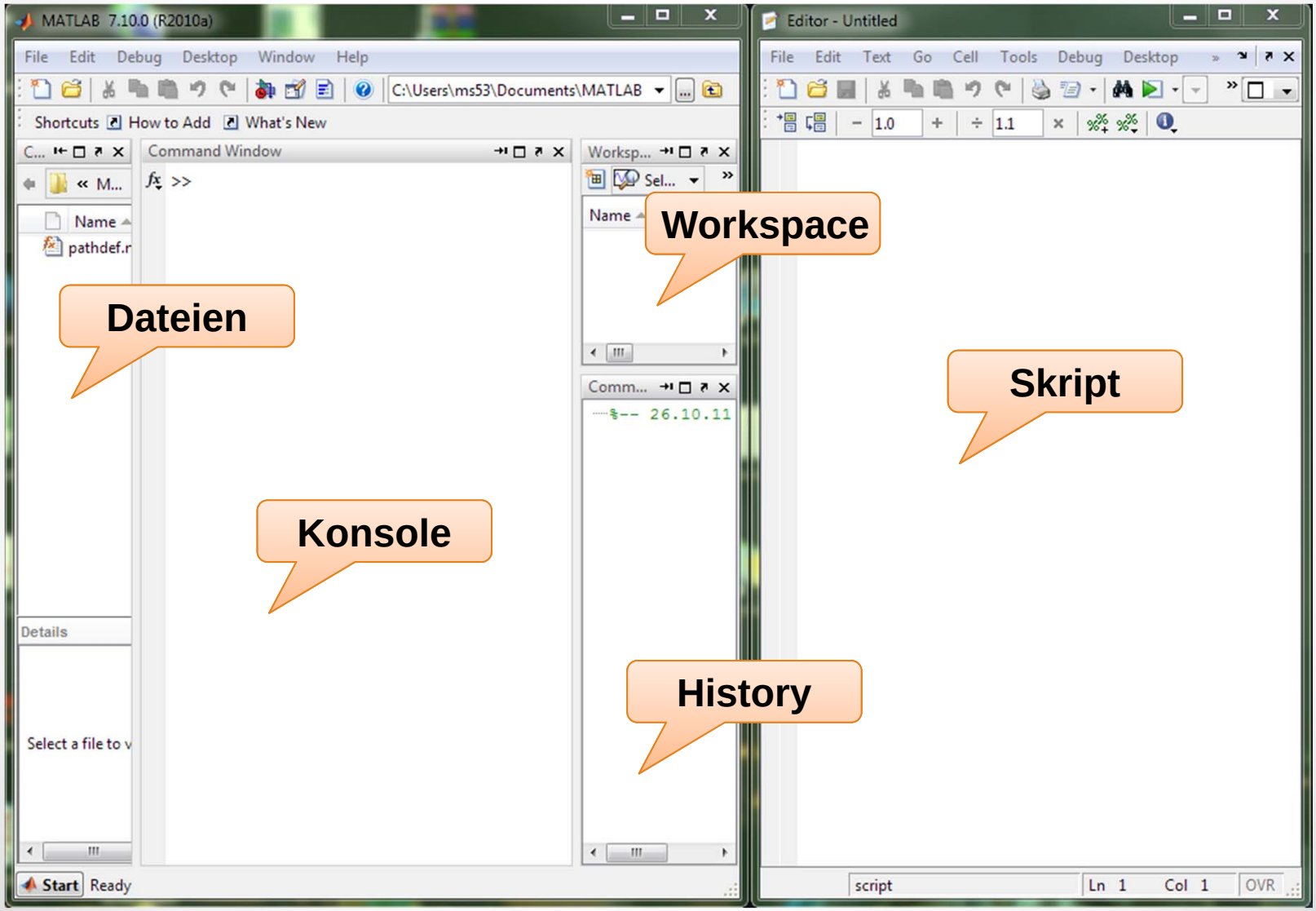
Inhalt der Einführung

- 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

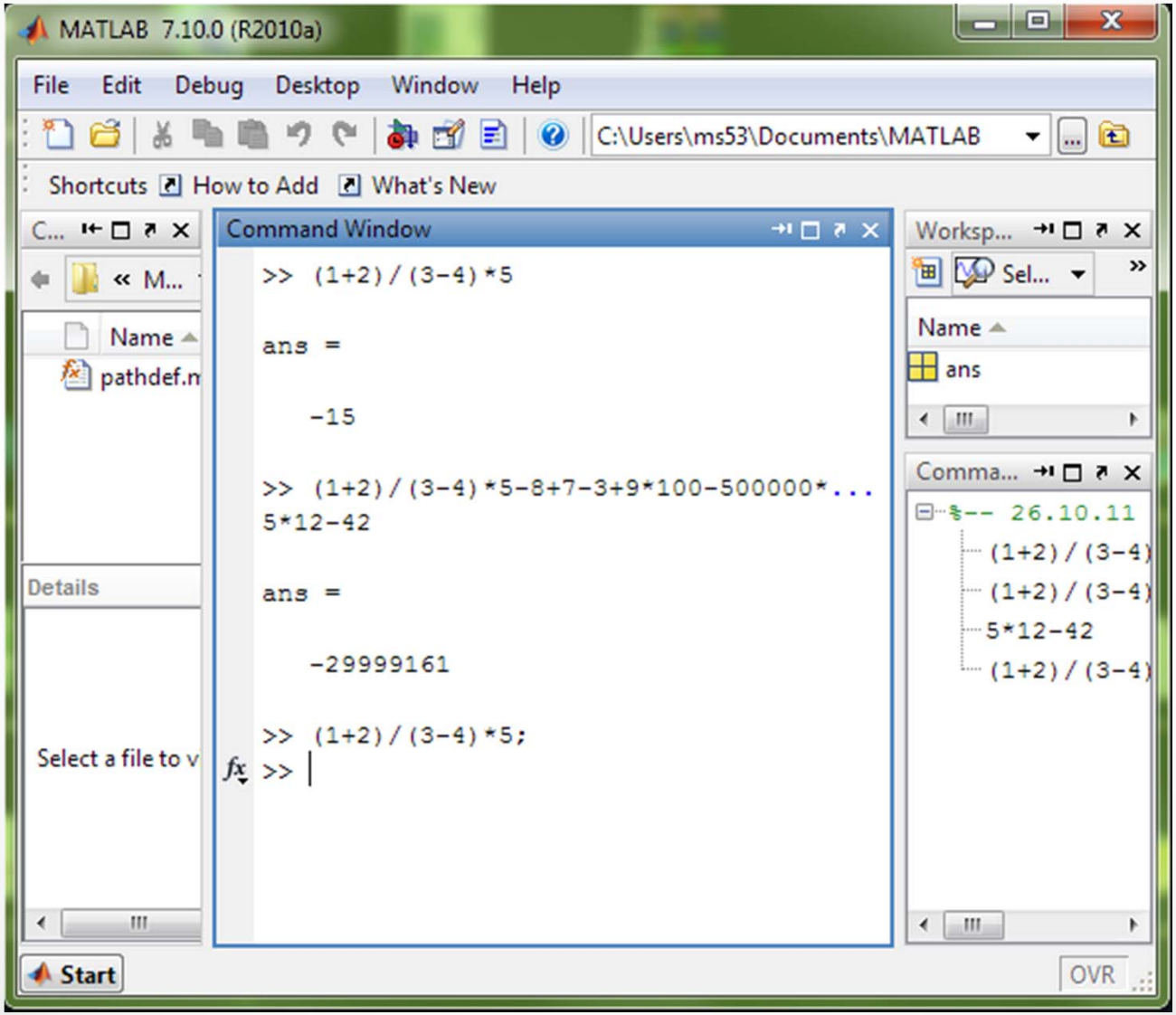
```
tschechne@adrenalin:~/workspace/matlab$ ./localmatlab -nojvm -nodisplay

                            < 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?

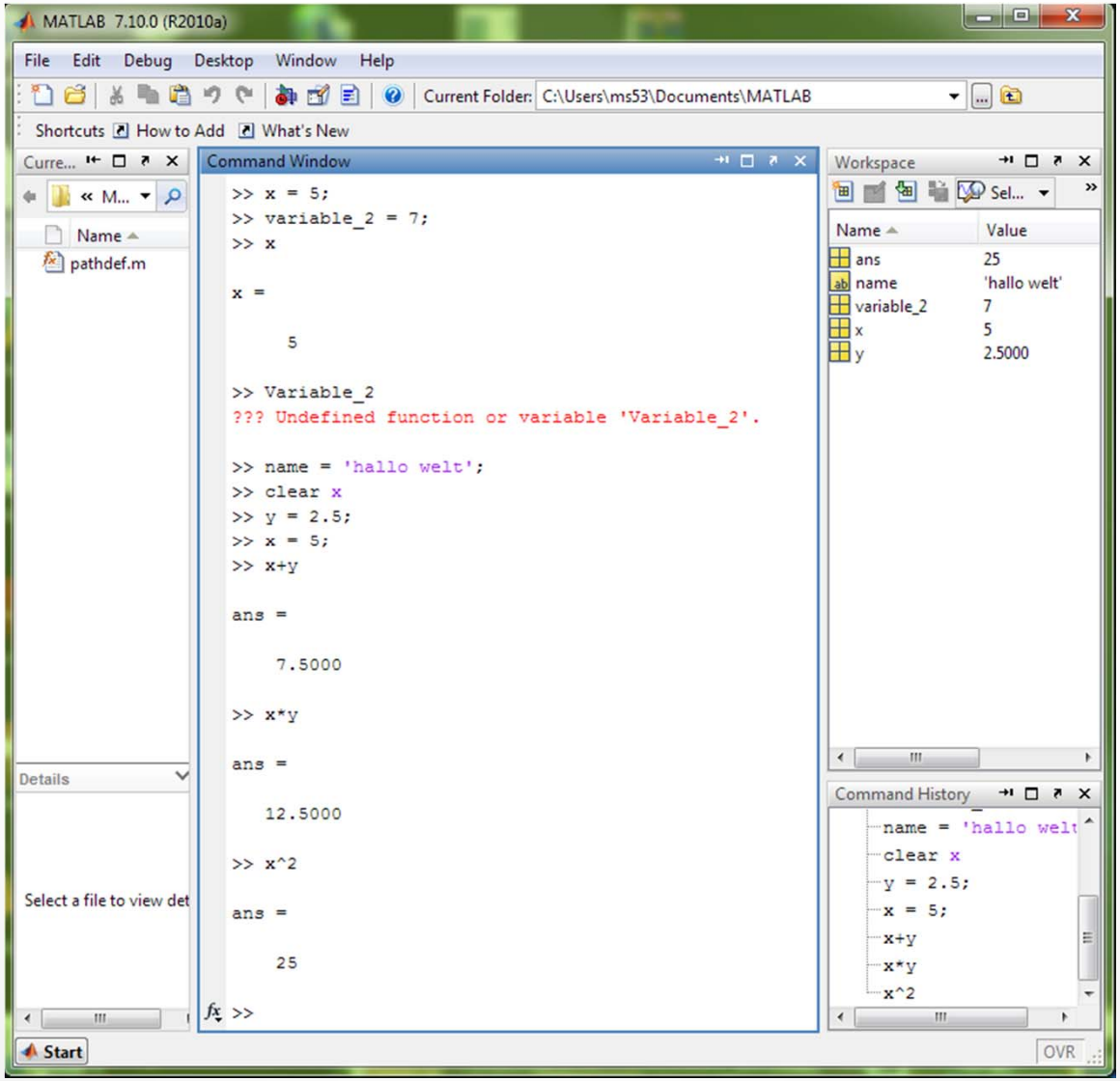


Kommandos

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)

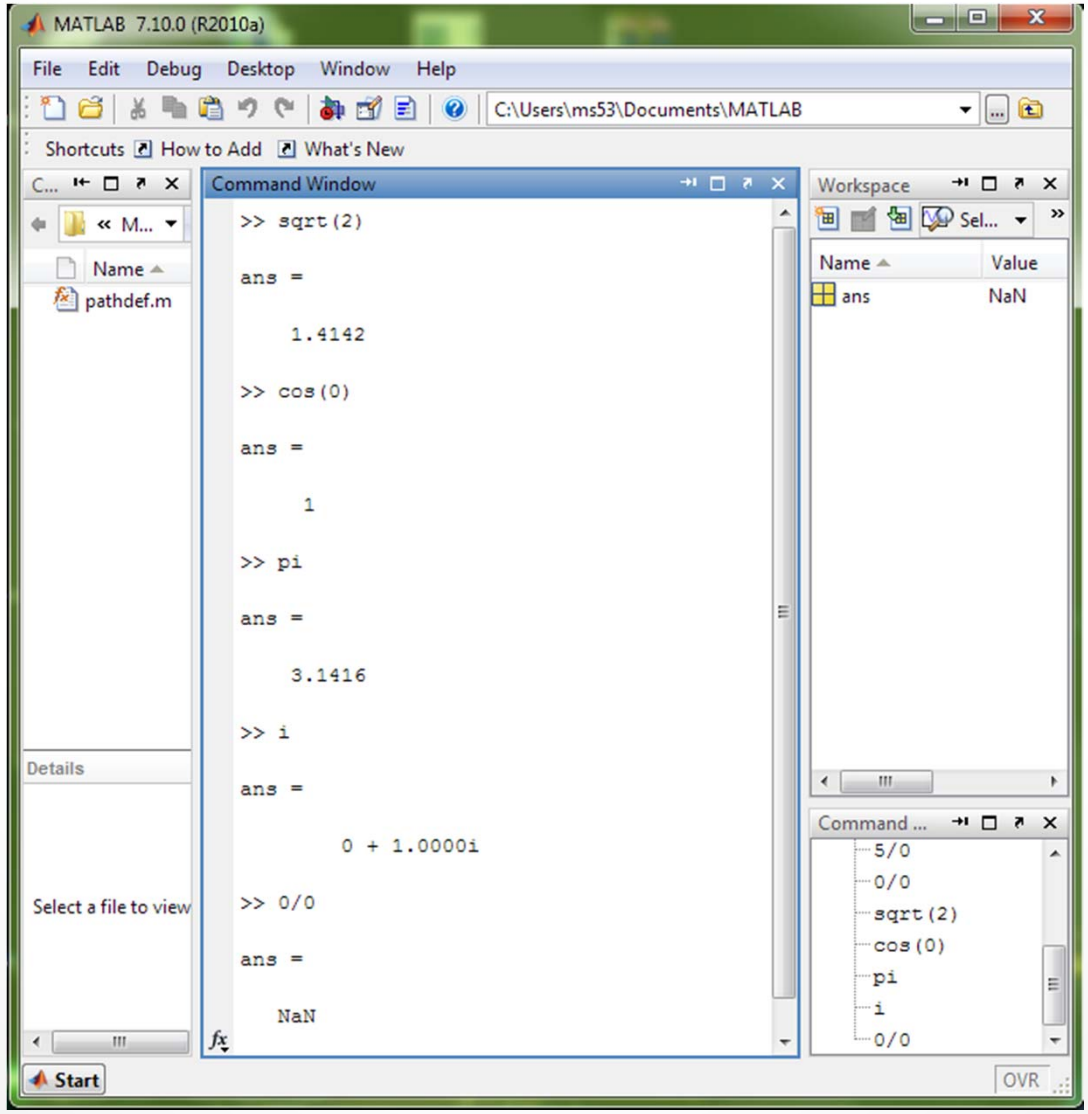
Kommandos


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Current Folder: C:\Users\ms53\Documents\MATLAB
Shortcuts How to Add What's New
Command Window
>> 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
>>
Workspace
Name Value
ans 25
name 'hallo welt'
variable_2 7
x 5
y 2.5000
Command History
name = 'hallo welt'
clear x
y = 2.5;
x = 5;
x+y
x*y
x^2
Name
pathdef.m
Details
Select a file to view det
Start
OVR

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 benutze Konstanten sind vordefiniert:
    - pi = 3.1415…
    - i = imaginärer Teil bei komplexen Zahlen
    - inf = unendlich
    - nan = mathematisch nicht definierte Zahl, z.B. 0/0 = nan

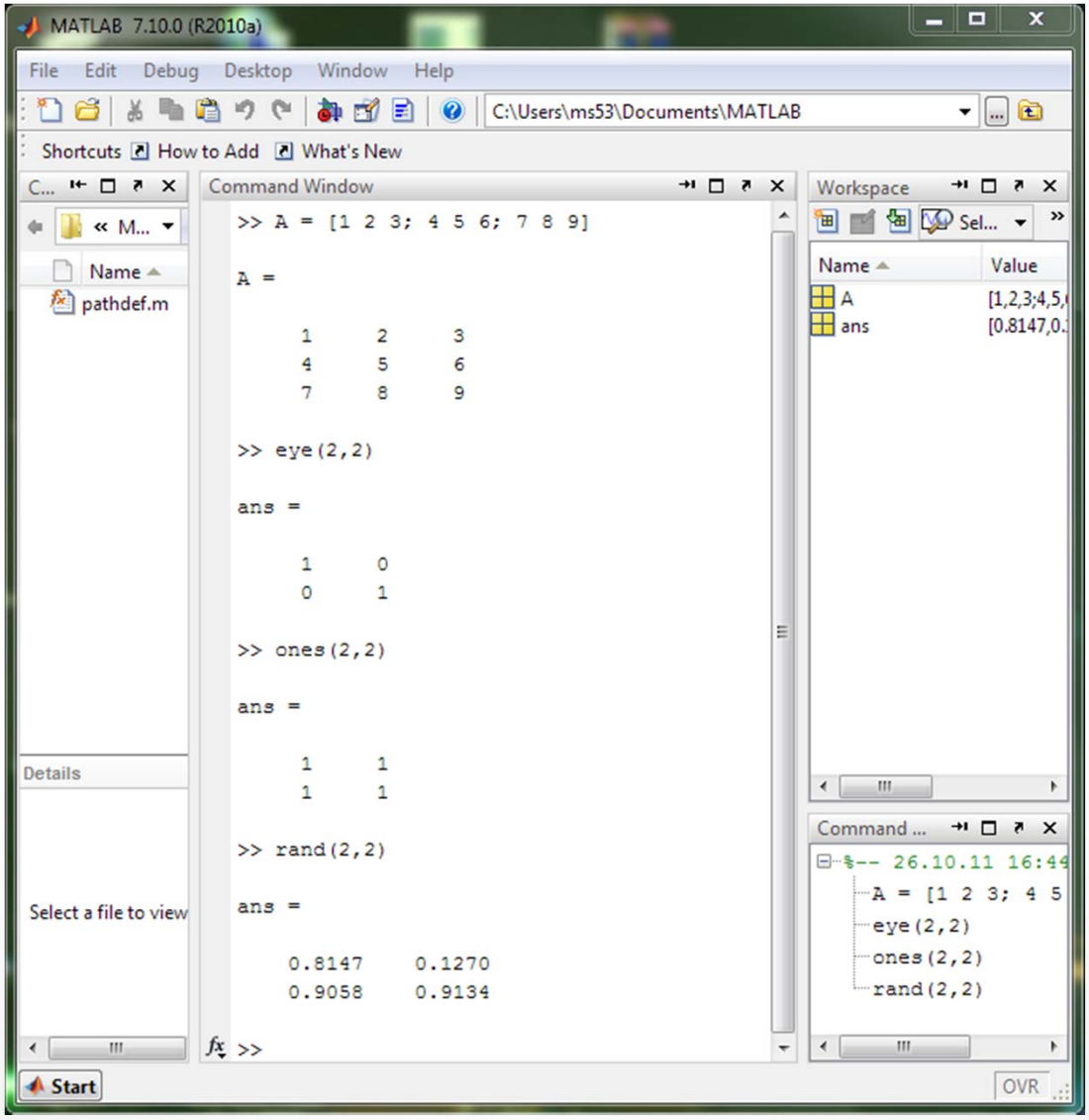
Kommandos (2)


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
C:\Users\ms53\Documents\MATLAB
Shortcuts How to Add What's New
Command Window
>> sqrt(2)
ans =
1.4142
>> cos(0)
ans =
1
>> pi
ans =
3.1416
>> i
ans =
0 + 1.0000i
>> 0/0
ans =
NaN
Workspace
Name Value
ans NaN
Command ...
5/0
0/0
sqrt(2)
cos(0)
pi
i
0/0
Name
pathdef.m
Details
Select a file to view
Start
OVR

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

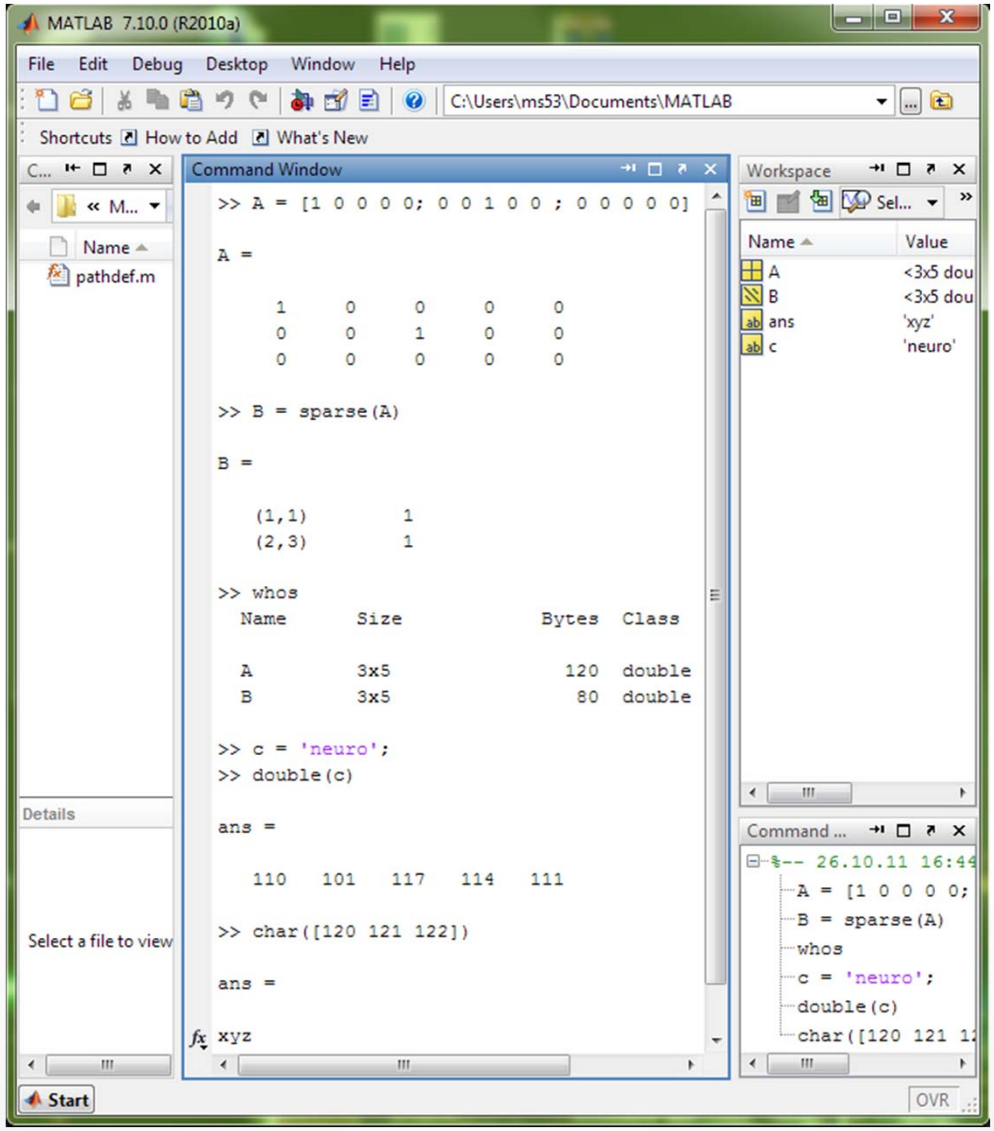
Datentypen


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
C:\Users\ms53\Documents\MATLAB
Shortcuts How to Add What's New
Command Window
>> 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
>>
Workspace
Name Value
A [1,2,3;4,5,
ans [0.8147,0.
Command ...
%-- 26.10.11 16:44
A = [1 2 3; 4 5
eye(2,2)
ones(2,2)
rand(2,2)
Name
pathdef.m
Details
Select a file to view
Start
OVR

Datentypen (2)

- Dünn besetze 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“

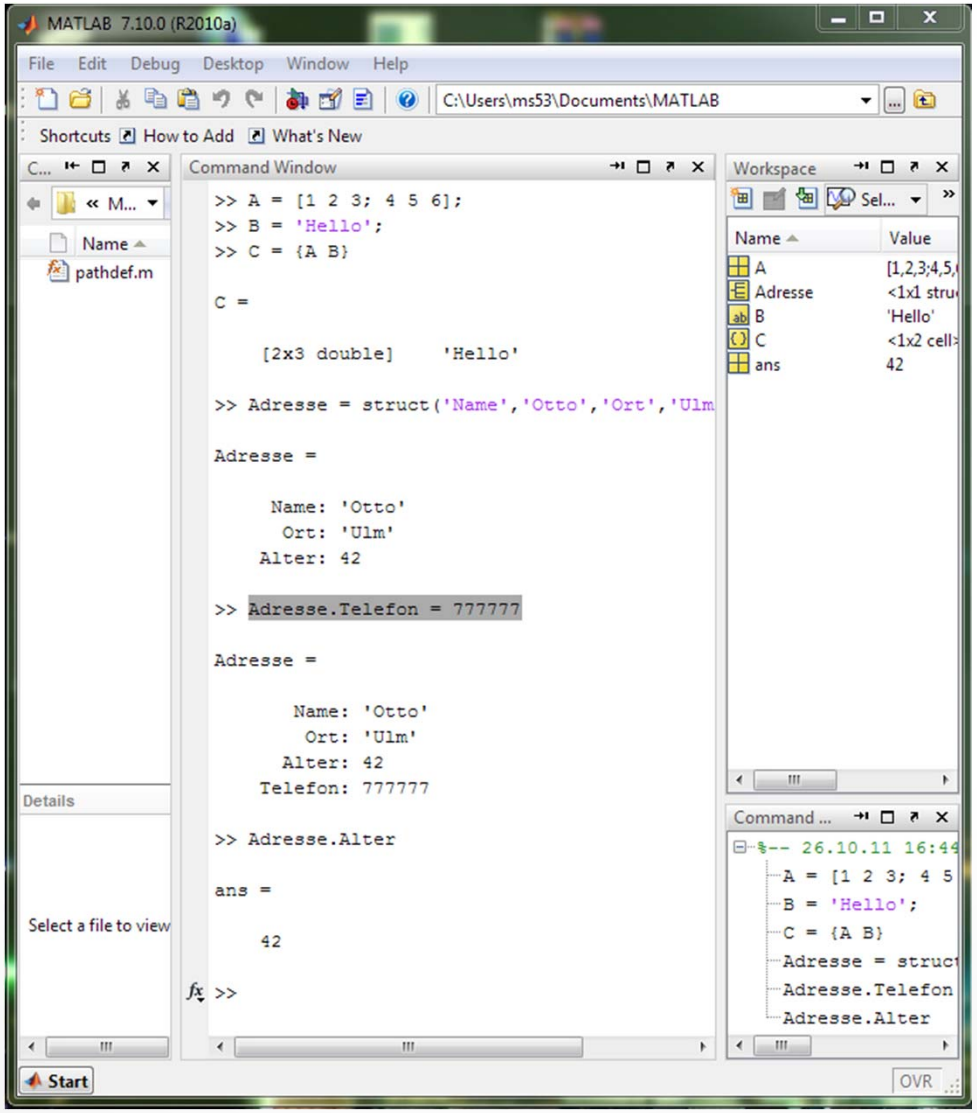
Datentypen (2)



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.neuesElelement` hinzugefügt werden
  - Bestehende Felder können so auch abgefragt werden

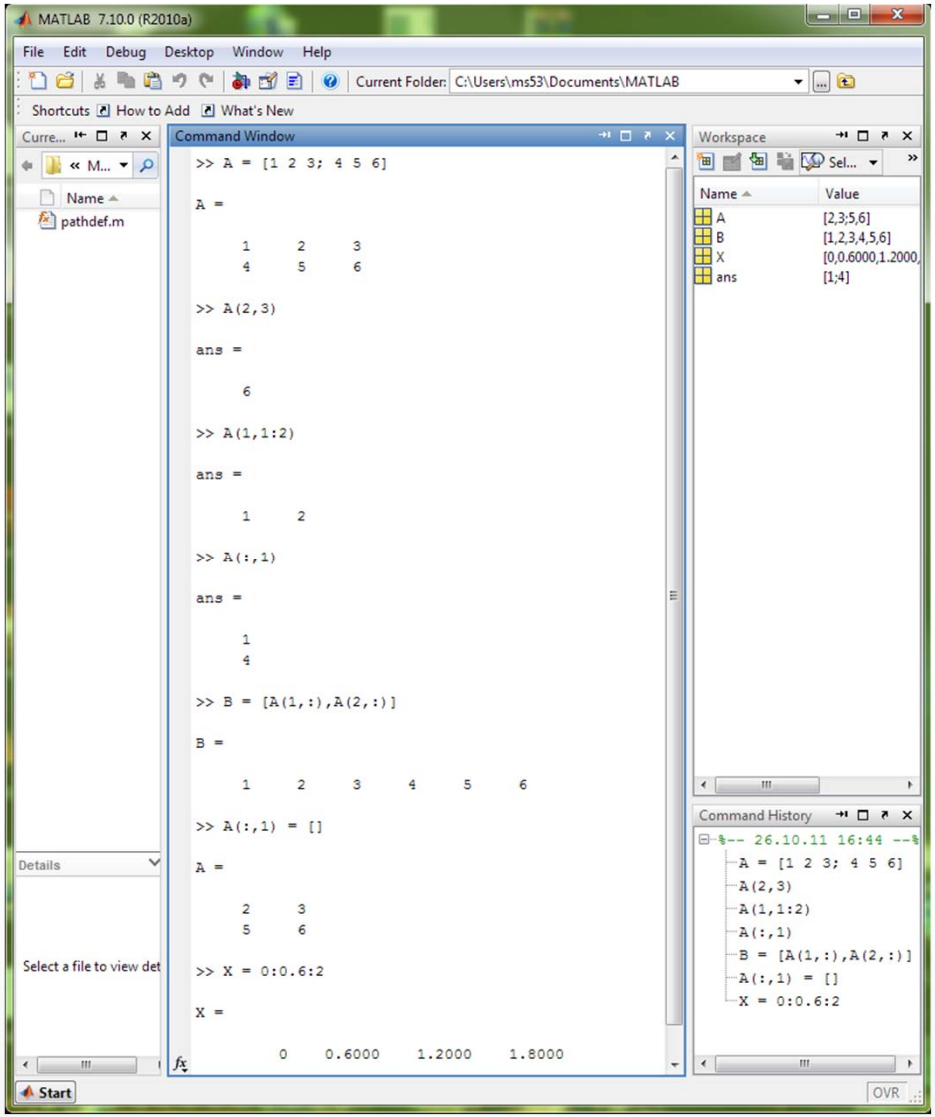
Datentypen (3)



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



Indexierung (2)

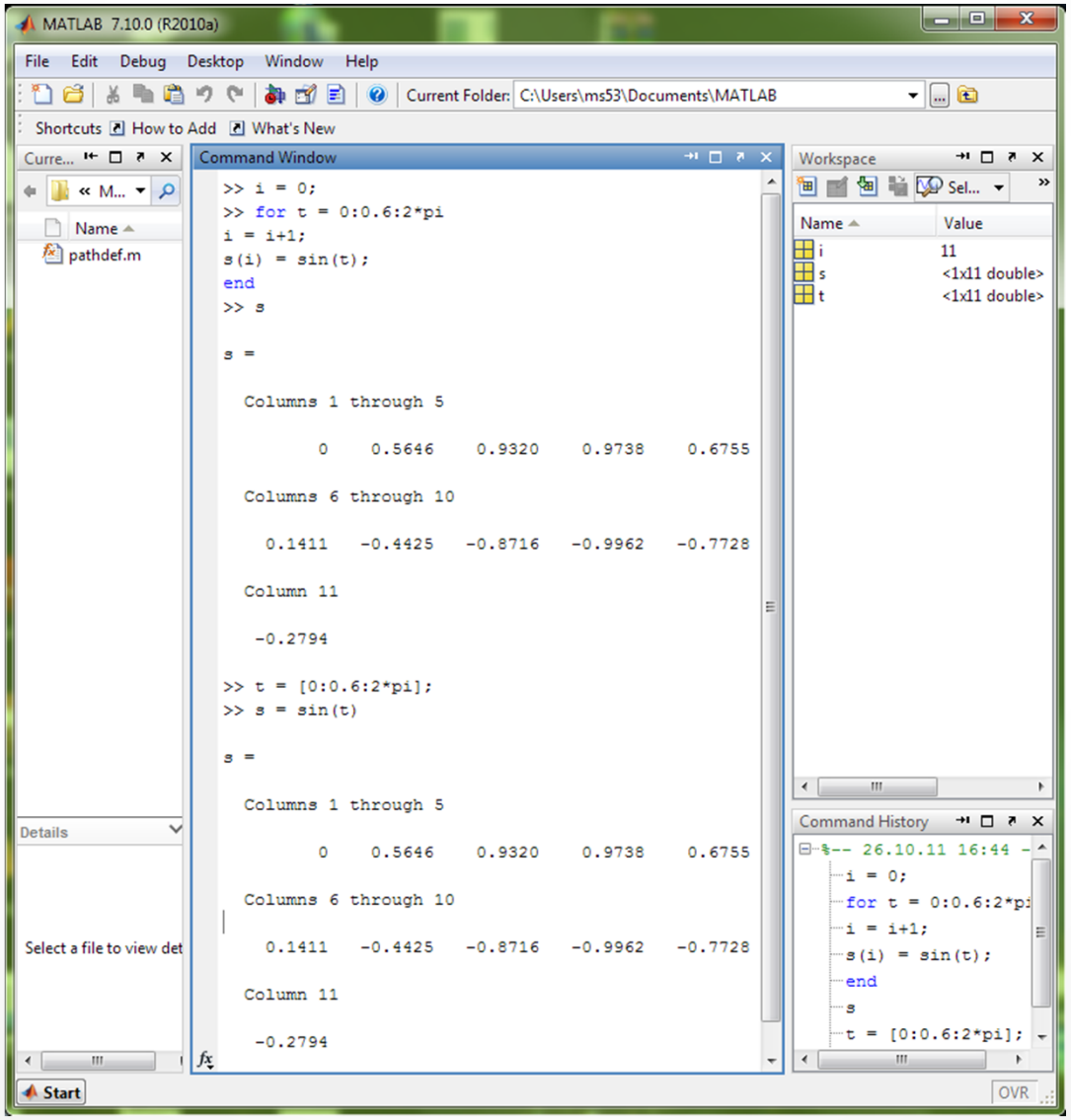
- Vektorisierung und Schleifen
  - Um einen Vektor zu besetzen, können alle Elemente der Reihe nach gefüllt werden
  - Vektor s mit Sinus-Werten von 0 bis 2*pi mit Schrittweite 0.6

```
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)



Indexierung (3)

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

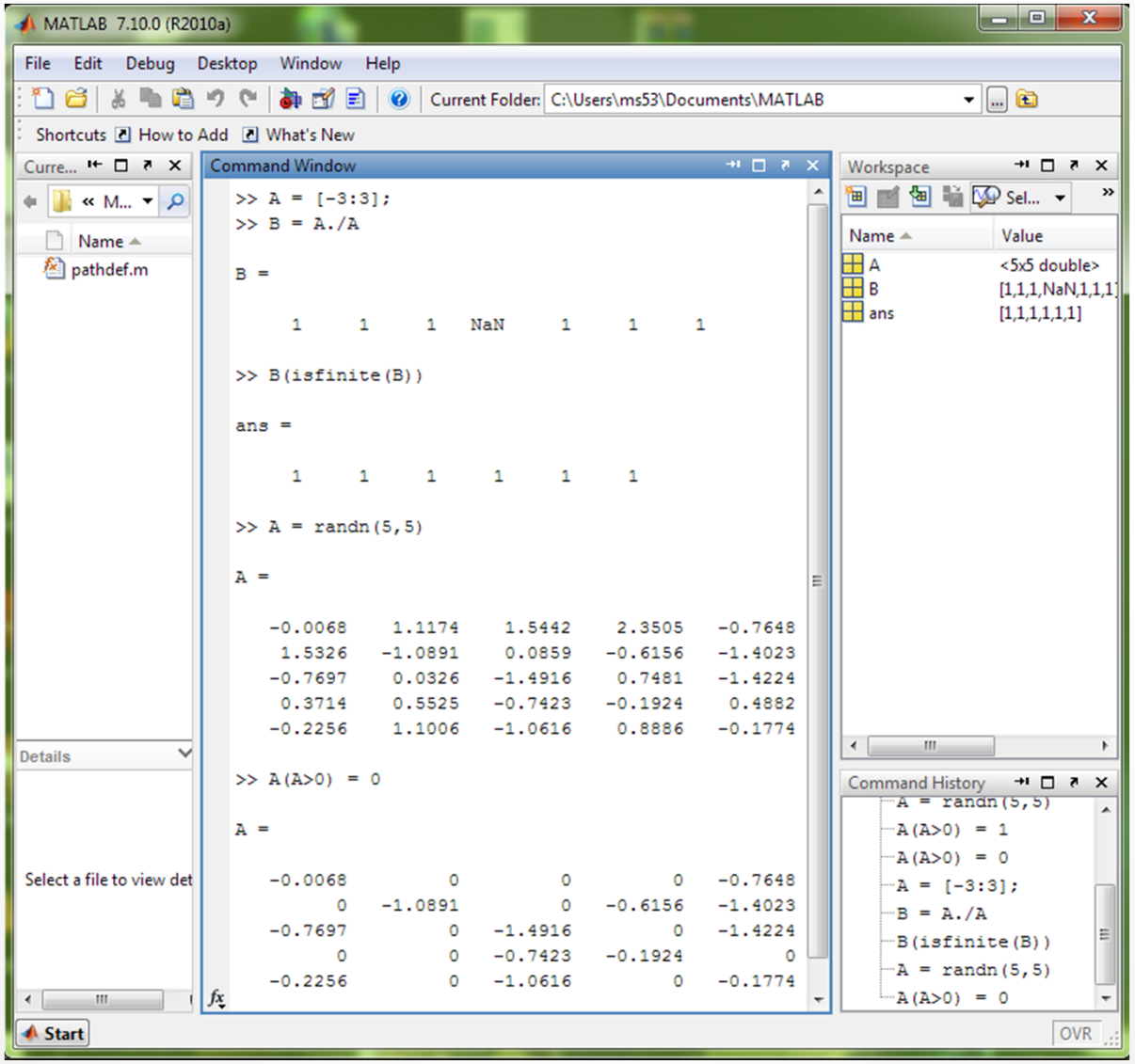
    ```
    A = [-3:3];
    ```

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

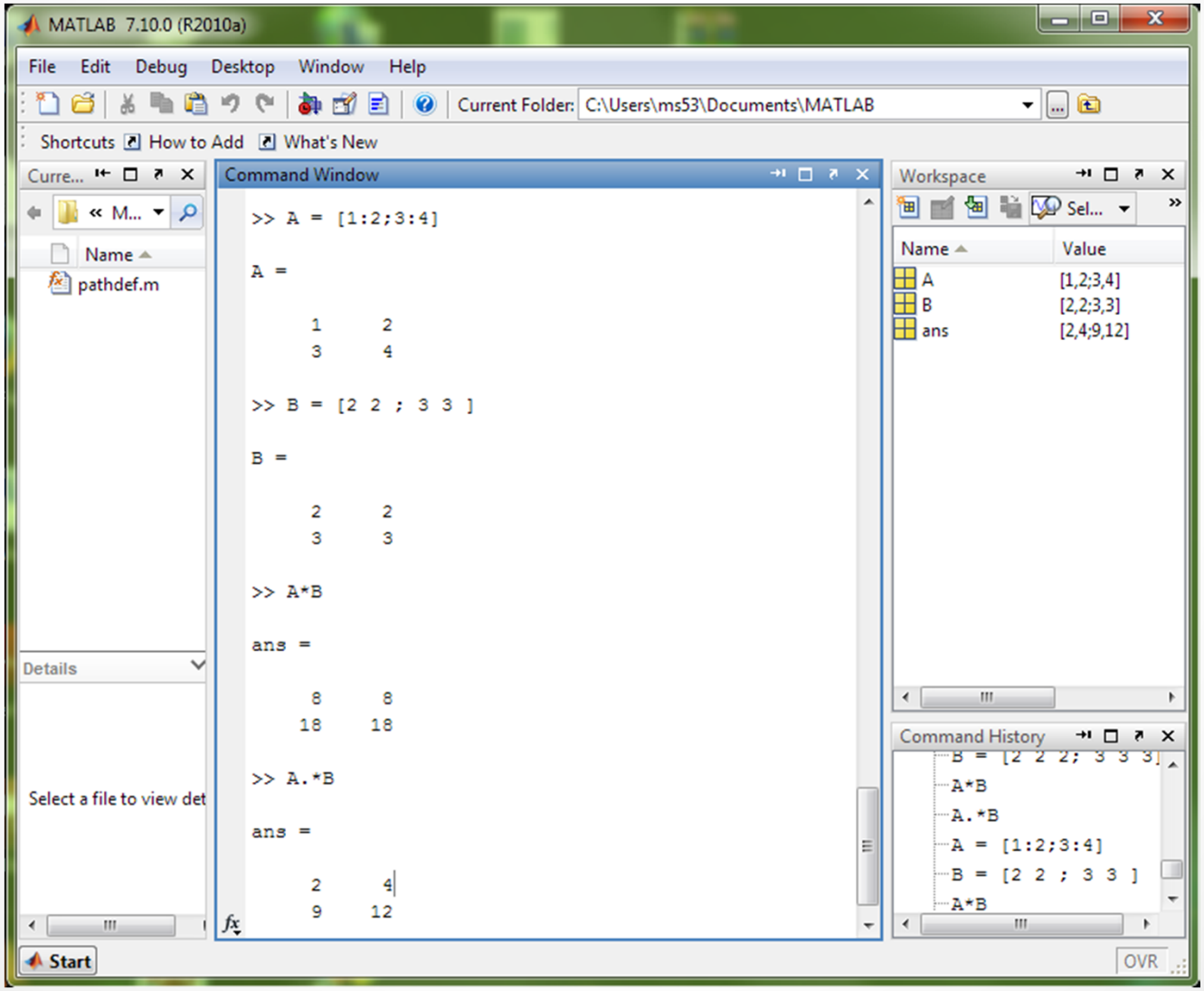
    ```
    B(isfinite(B))
    ```
  - Aus einer zufälligen normalverteilten 5x5 Matrix alle Elemente auf Null setzen, die größer als `0` sind:

    ```
    A = randn(5,5)
    A(A>0)=1
    ```
- Operator auf Elemente in Matrizen
  - .* und ./ elementweise Multiplikationen und Division
  - .^ elementweise Exponentation

Indexierung (3)



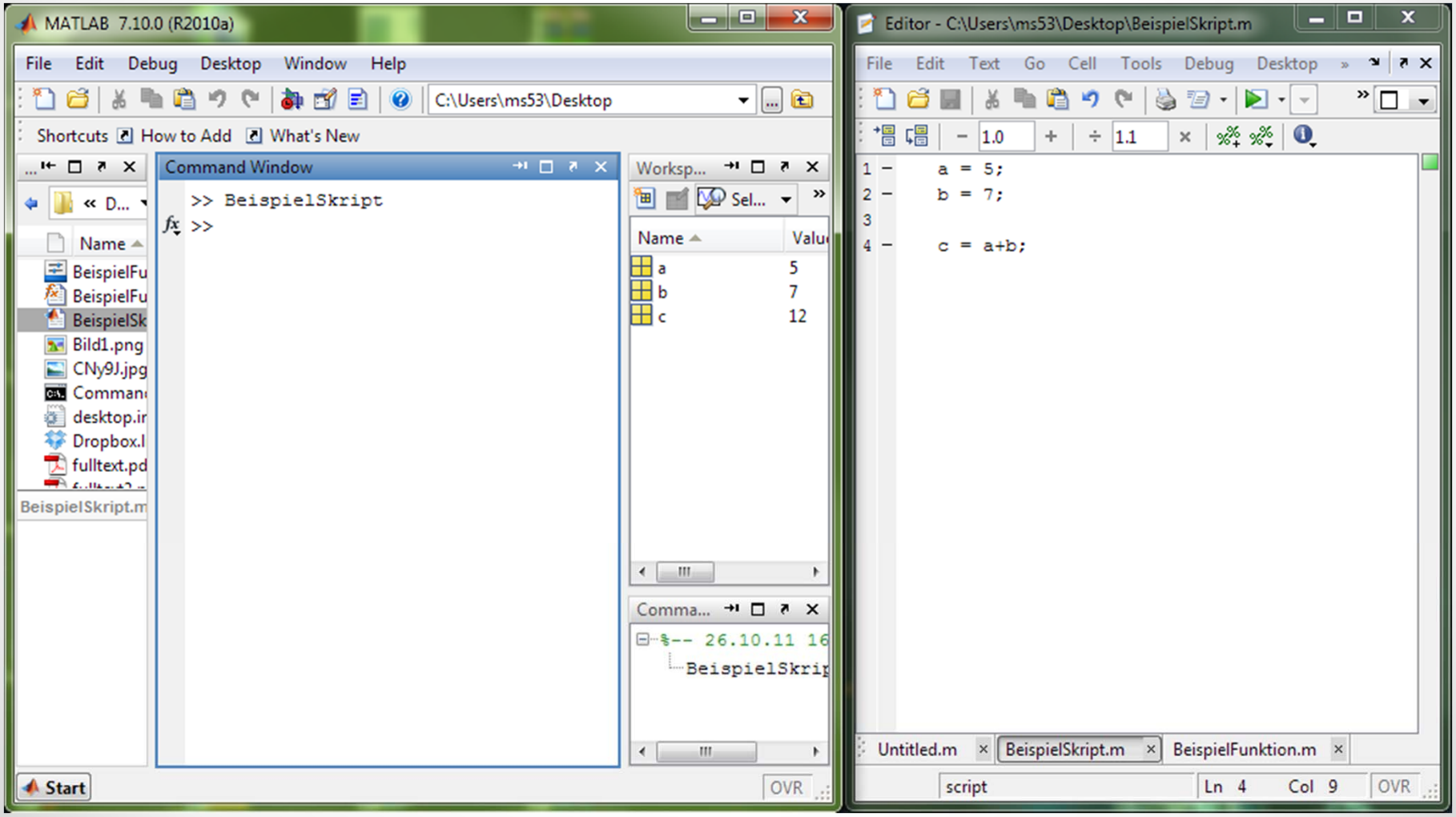
Indexierung (3)



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.

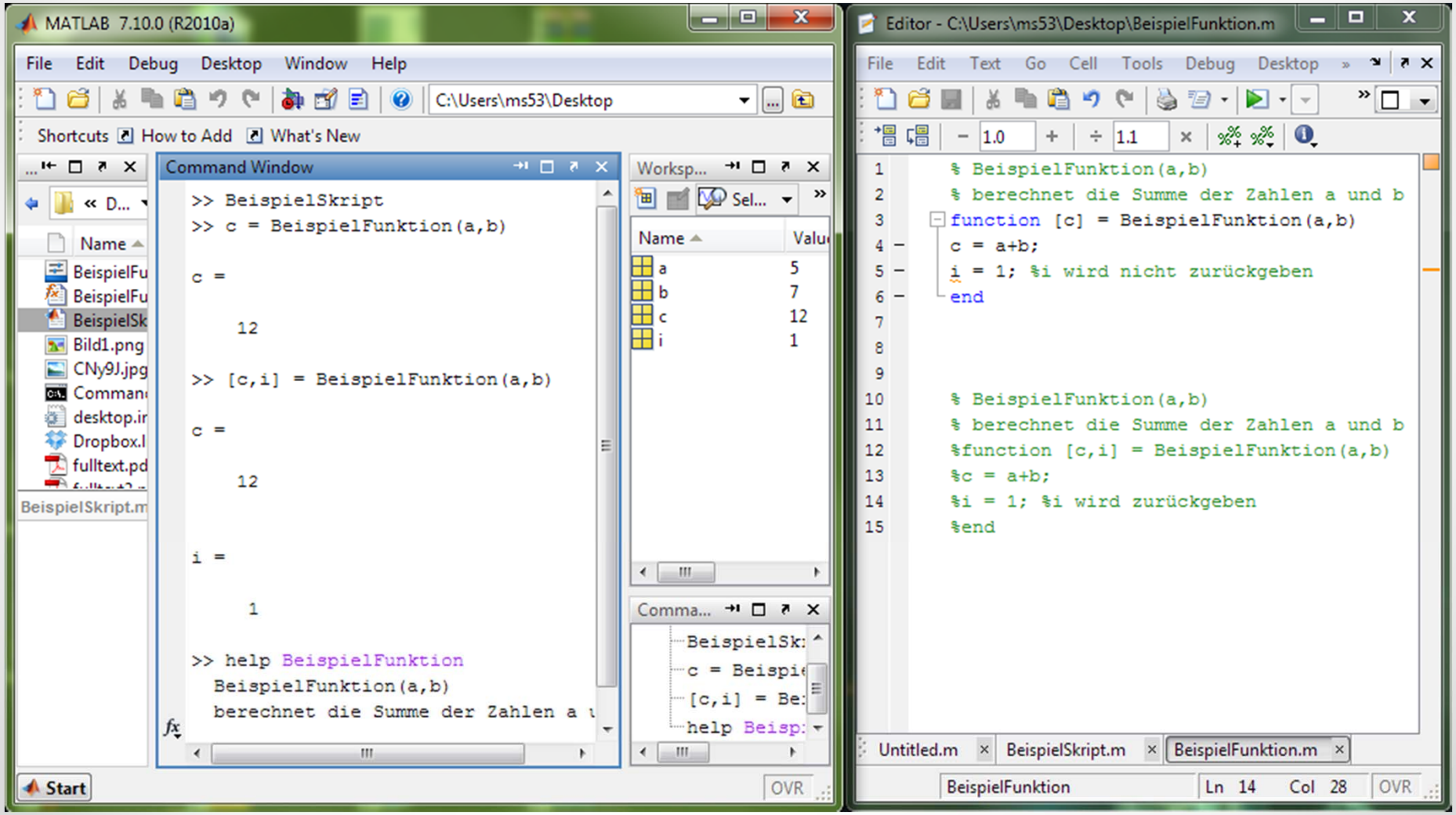
Programmierung



Programmierung (2)

- Funktionen
  - werden wie Skripte erzeugt, abgespeichert 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)



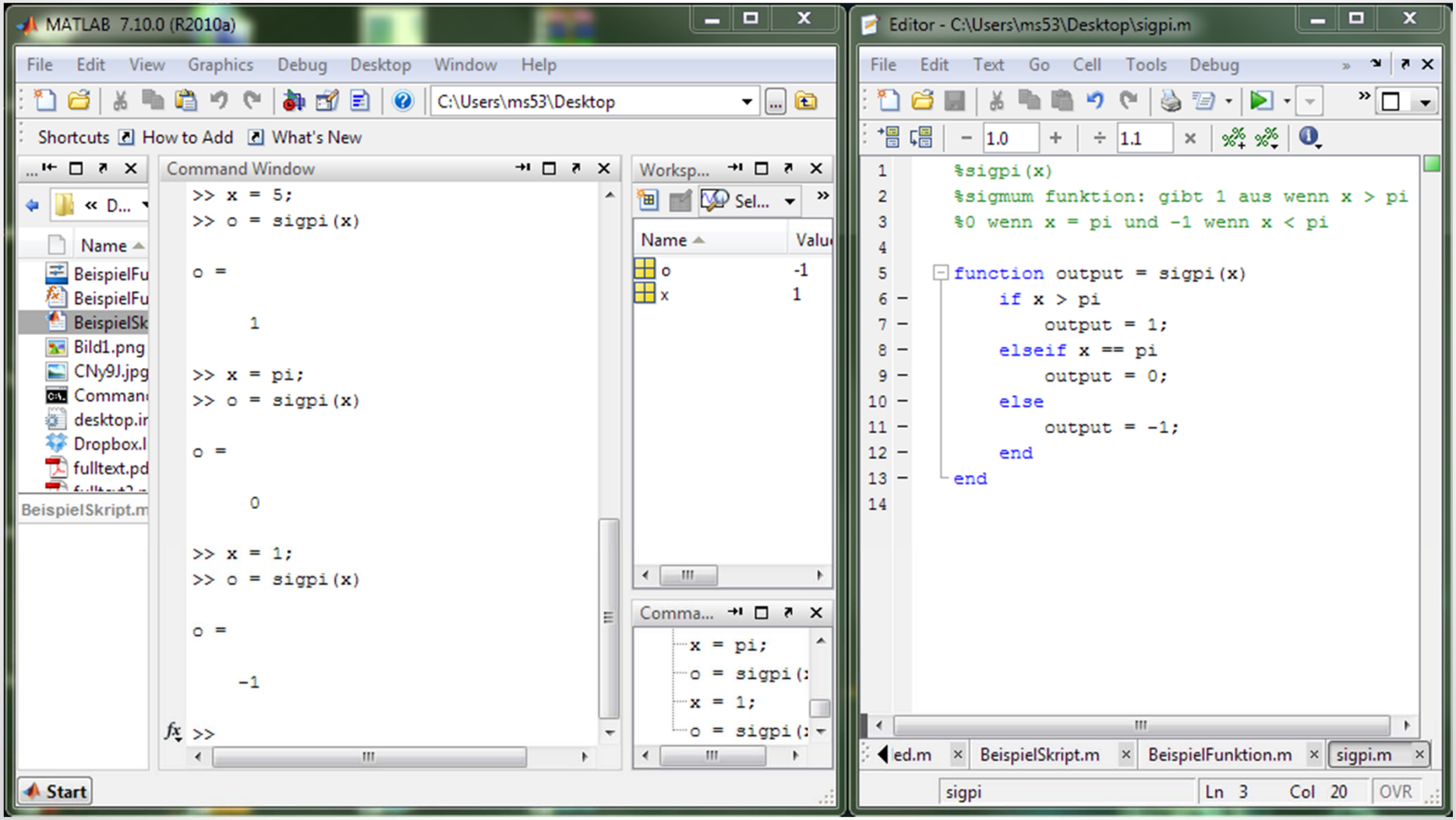
Programmierung (3)

- Flusskontrolle durch:
  - `if i == 1  …  end; (elseif, else)`
  - `switch i …`

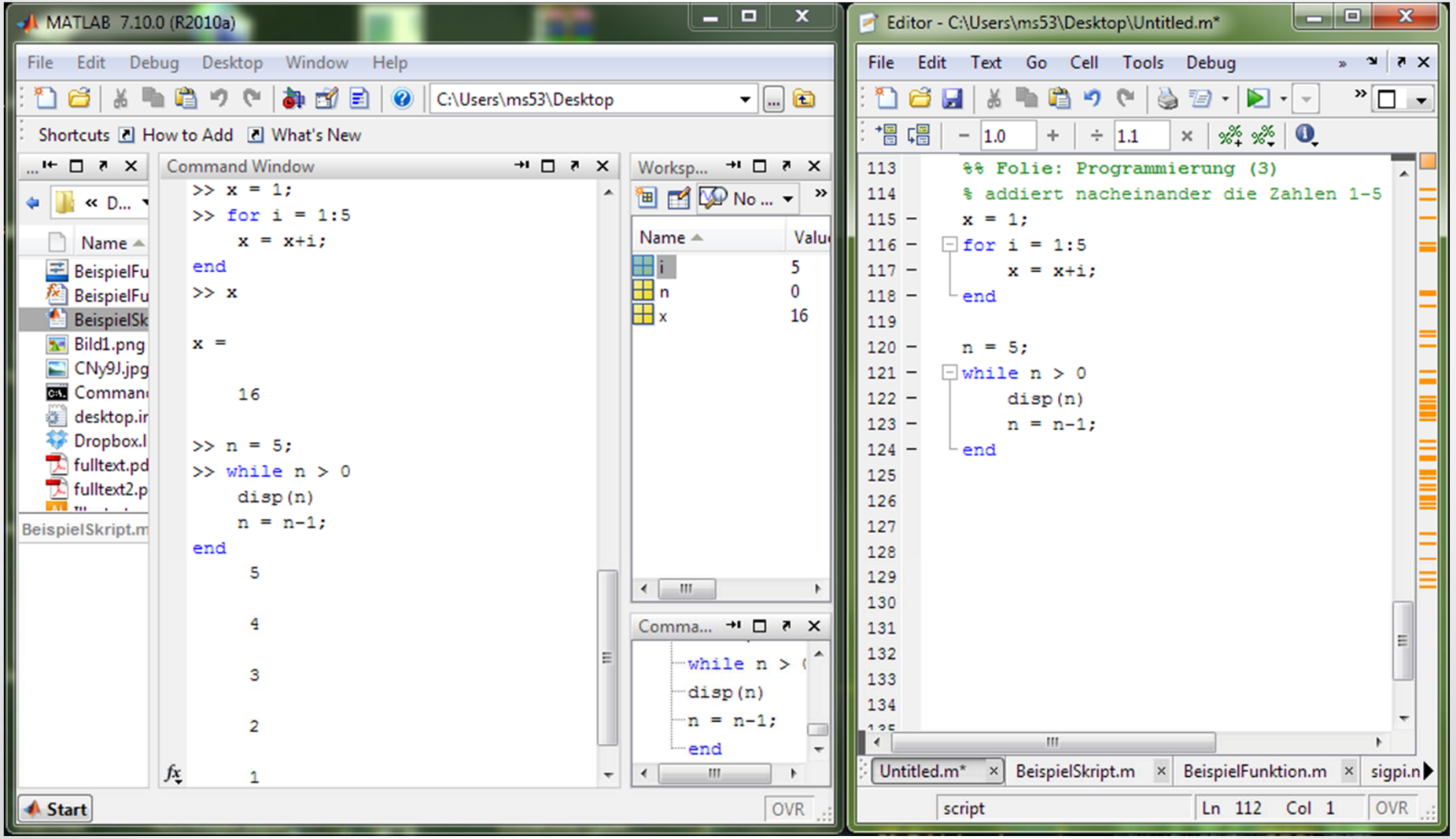
```
        case 1 …
        case 2 …
        otherwise …
   end;
```

  - `for i=1:10 … [break]  end;`
  - `while i > 0  … [break]  end;`

Programmierung (3)



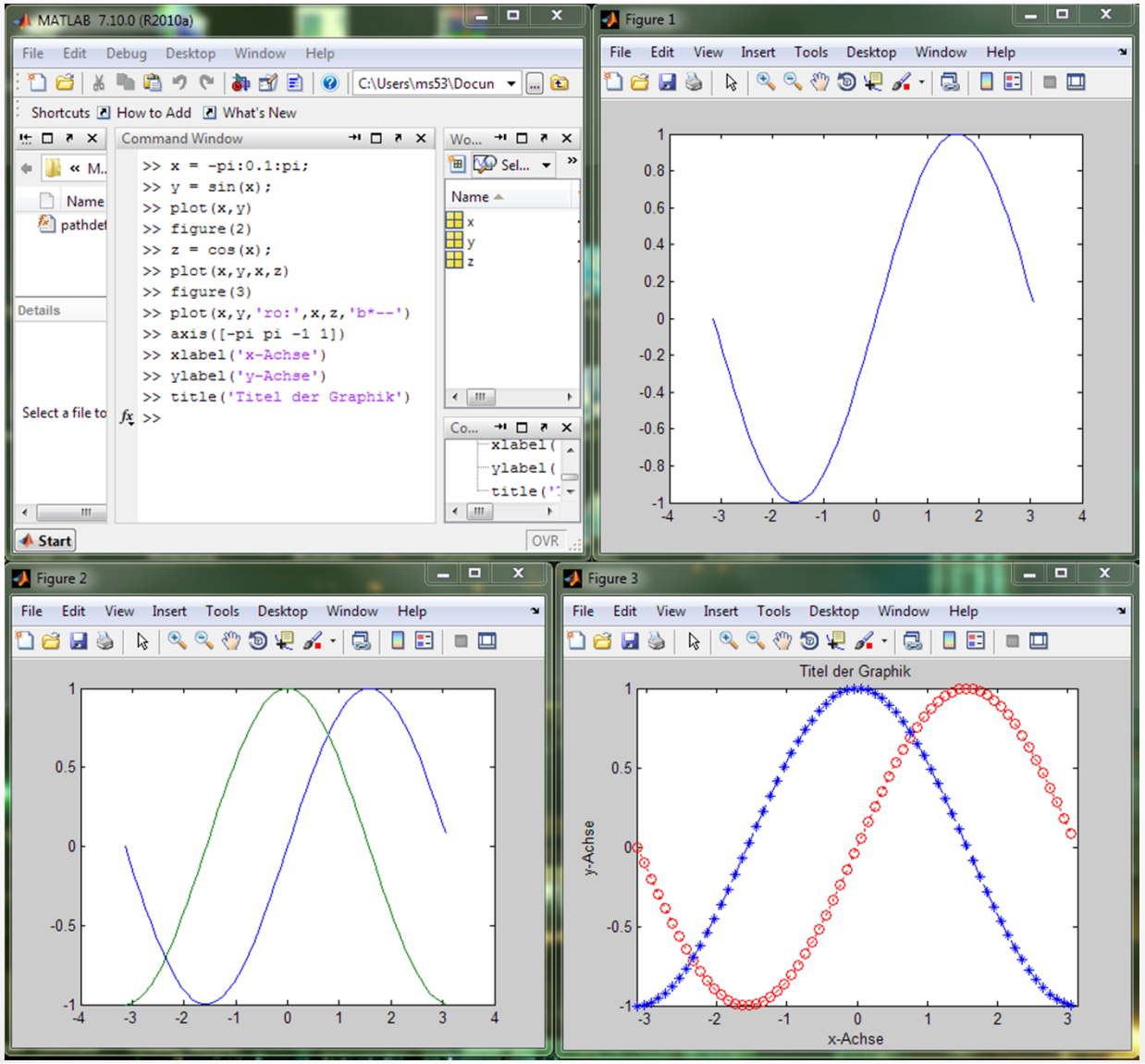
Programmierung (3)



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

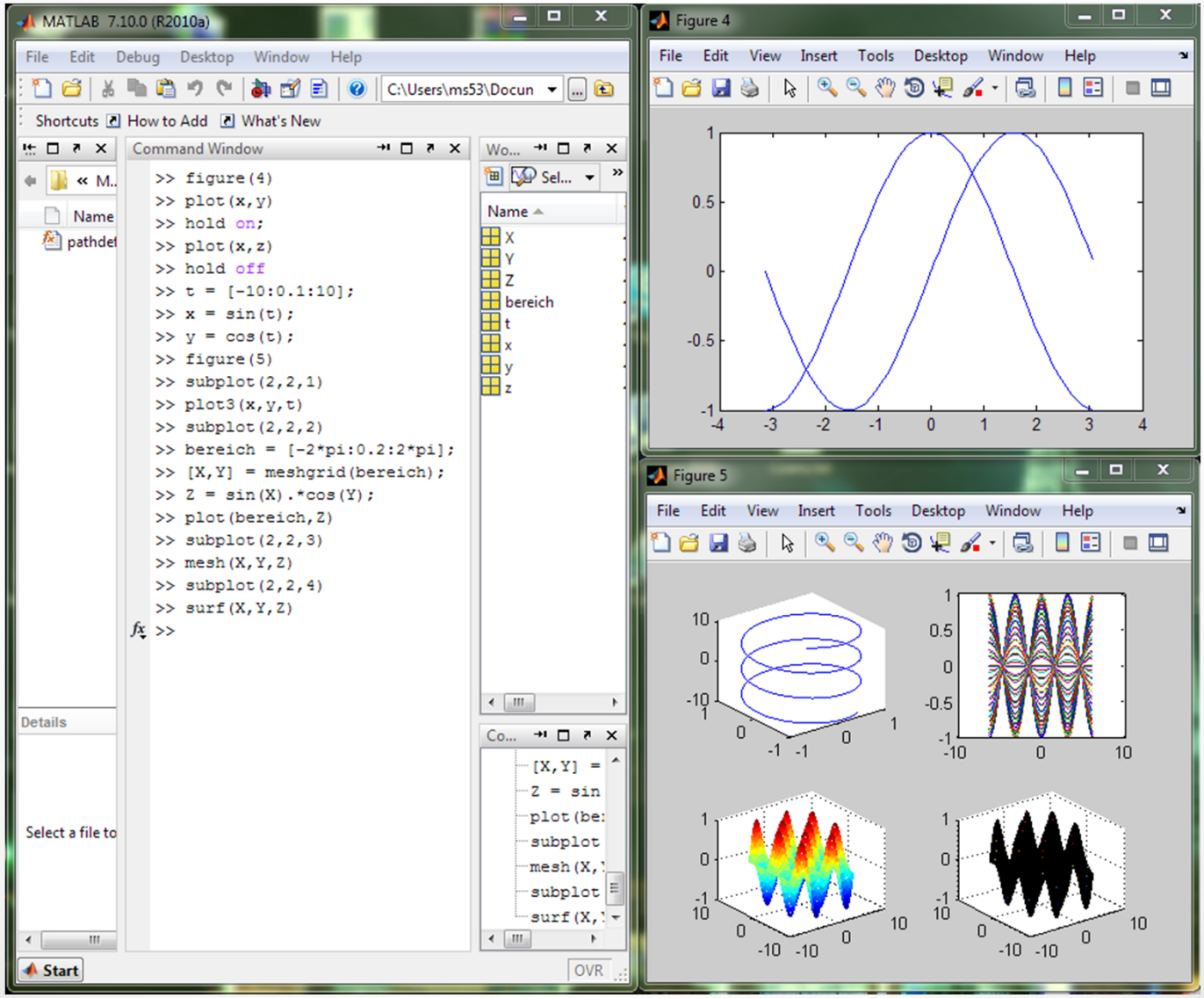

MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Command Window
>> x = -pi:0.1:pi;
>> y = sin(x);
>> plot(x,y)
>> figure(2)
>> z = cos(x);
>> plot(x,y,x,z)
>> figure(3)
>> plot(x,y,'ro:',x,z,'b*--')
>> axis([-pi pi -1 1])
>> xlabel('x-Achse')
>> ylabel('y-Achse')
>> title('Titel der Graphik')
>>
Name
x
y
z
Details
Select a file to
Start
OVR
Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help
Figure 3
File Edit View Insert Tools Desktop Window Help
Titel der Graphik
y-Achse
x-Achse

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`

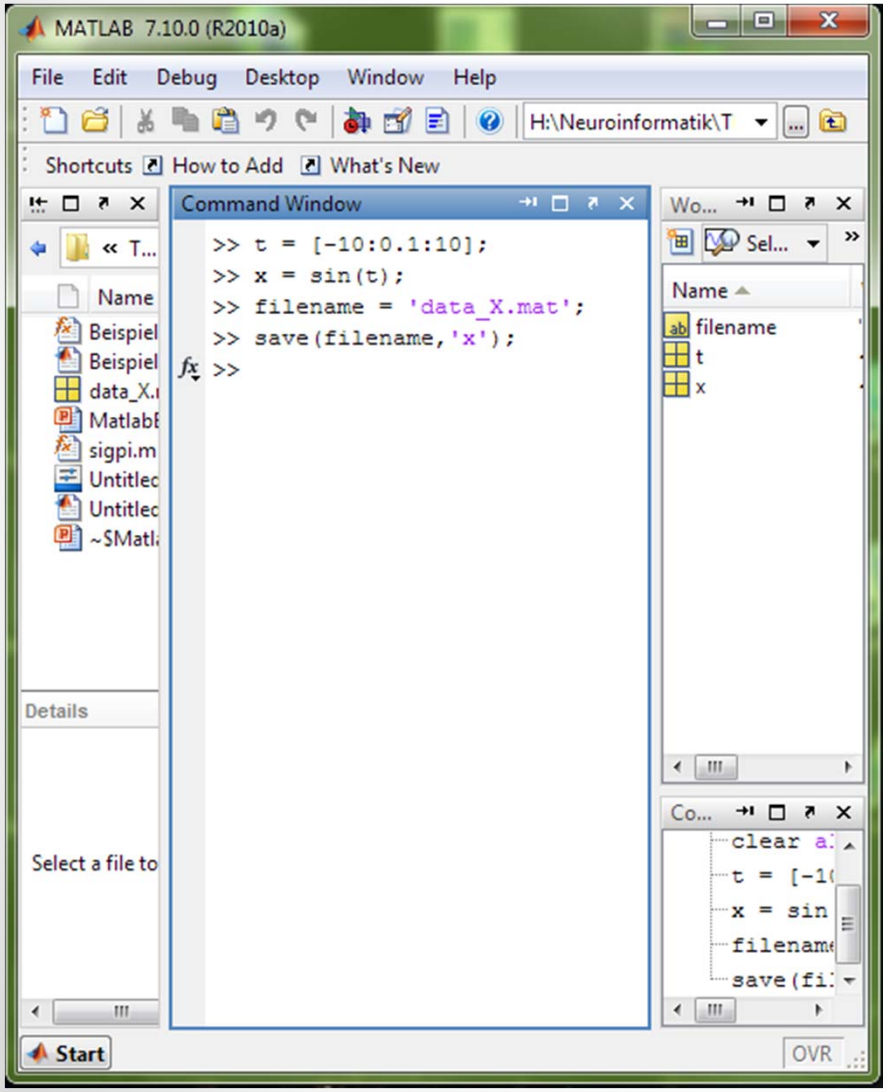
Grafik (2)

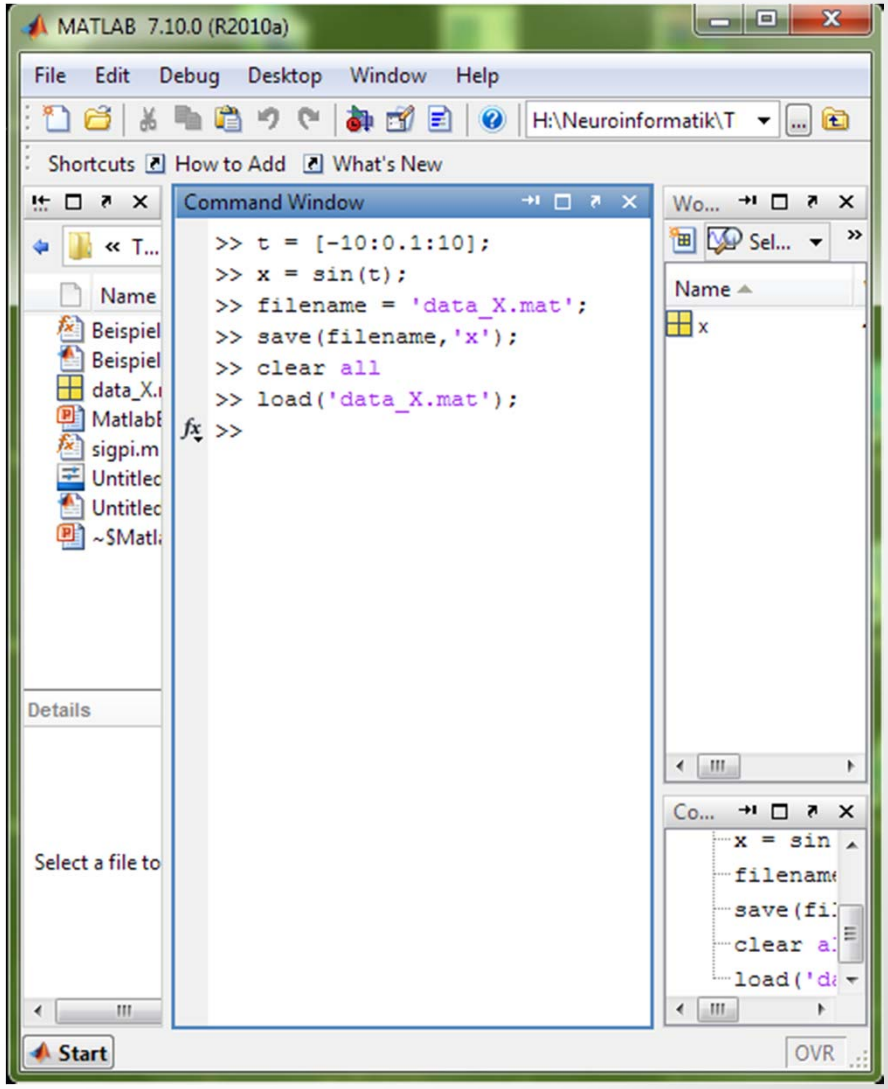

MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Command Window
>> figure(4)
>> plot(x,y)
>> hold on;
>> plot(x,z)
>> hold off
>> t = [-10:0.1:10];
>> x = sin(t);
>> y = cos(t);
>> figure(5)
>> subplot(2,2,1)
>> plot3(x,y,t)
>> subplot(2,2,2)
>> bereich = [-2*pi:0.2:2*pi];
>> [X,Y] = meshgrid(bereich);
>> Z = sin(X).*cos(Y);
>> plot(bereich,Z)
>> subplot(2,2,3)
>> mesh(X,Y,Z)
>> subplot(2,2,4)
>> surf(X,Y,Z)
>>
Name
X
Y
Z
bereich
t
x
y
z
Details
Select a file to
Start
OVR
Figure 4
File Edit View Insert Tools Desktop Window Help
Figure 5
File Edit View Insert Tools Desktop Window Help

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





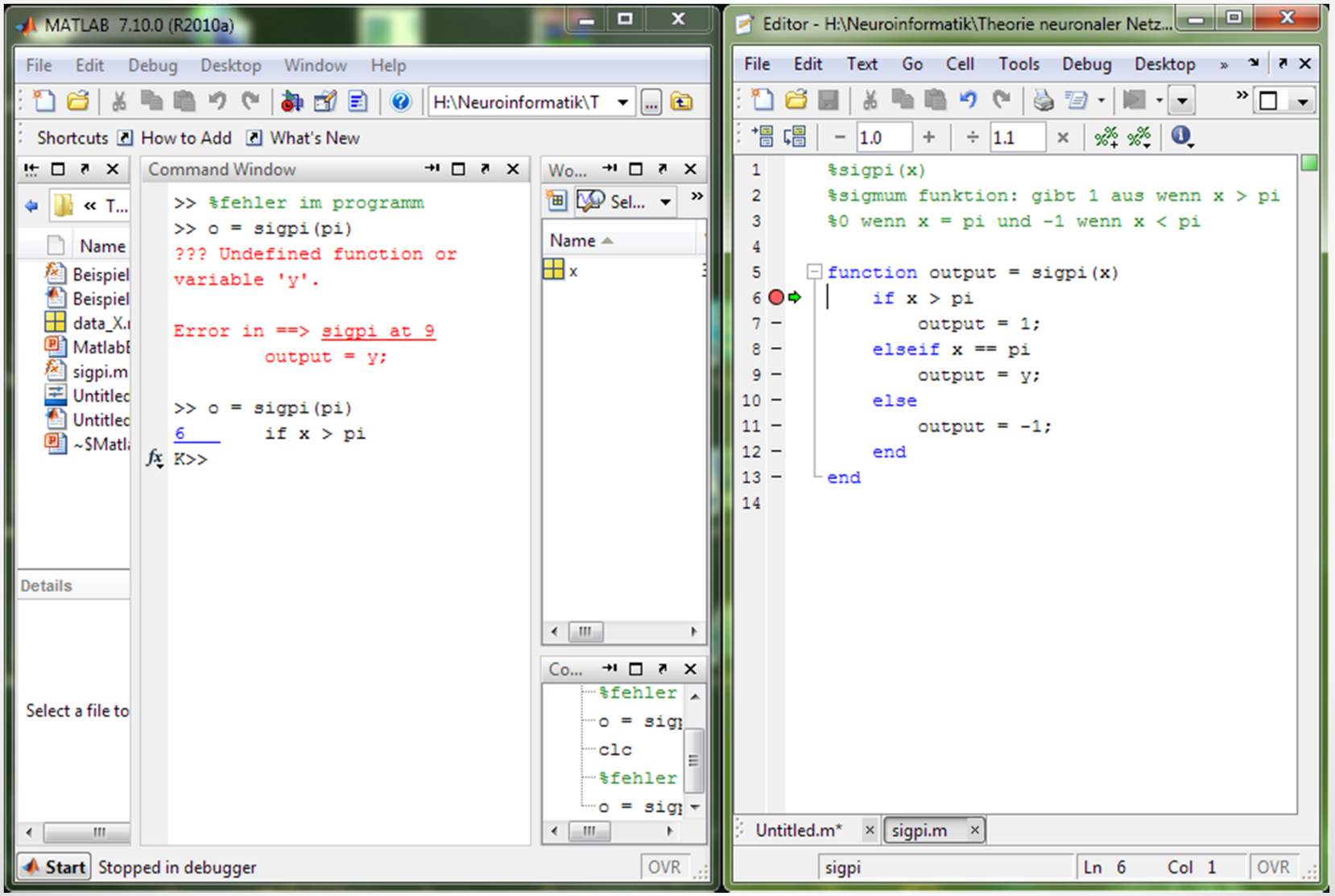
Weitere nützliche Kommentare

- `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


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
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
K>>
Name
x
Details
Select a file to
Start
Stopped in debugger
OVR
Editor - H:\Neuroinformatik\Theorie neuronaler Netz...
File Edit Text Go Cell Tools Debug Desktop
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
Untitled.m*
sigpi.m
sigpi
Ln 6 Col 1
OVR