Table of Contents

[II. MATLAB 2](#_Toc519573632)

[1.1 Allgemein 2](#_Toc519573633)

[1.2 Erstellen einfacher Variablen: 2](#_Toc519573634)

[1.3 Vektor 2](#_Toc519573635)

[1.4 Matrizen 2](#_Toc519573636)

[1.4.1 Multiplizieren von Matrizen: 3](#_Toc519573637)

[1.4.2 Zusammenführen von 2 Arrays: 4](#_Toc519573638)

[1.4.3 Einzelne Werte auslesen: 4](#_Toc519573639)

[1.5 Speichern und Laden: 5](#_Toc519573640)

[1.6 Array-Funktionen 5](#_Toc519573641)

[1.7 2D-Plot 6](#_Toc519573642)

[1.8 3D-Plots 7](#_Toc519573643)

[1.9 Schleifen 8](#_Toc519573644)

[1.9.1 For-Schleife: 8](#_Toc519573645)

[1.10 Funktionen 8](#_Toc519573646)

[1.11 Bildverarbeitung 8](#_Toc519573647)

[III. Quellen: 11](#_Toc519573648)

# MATLAB

## Allgemein

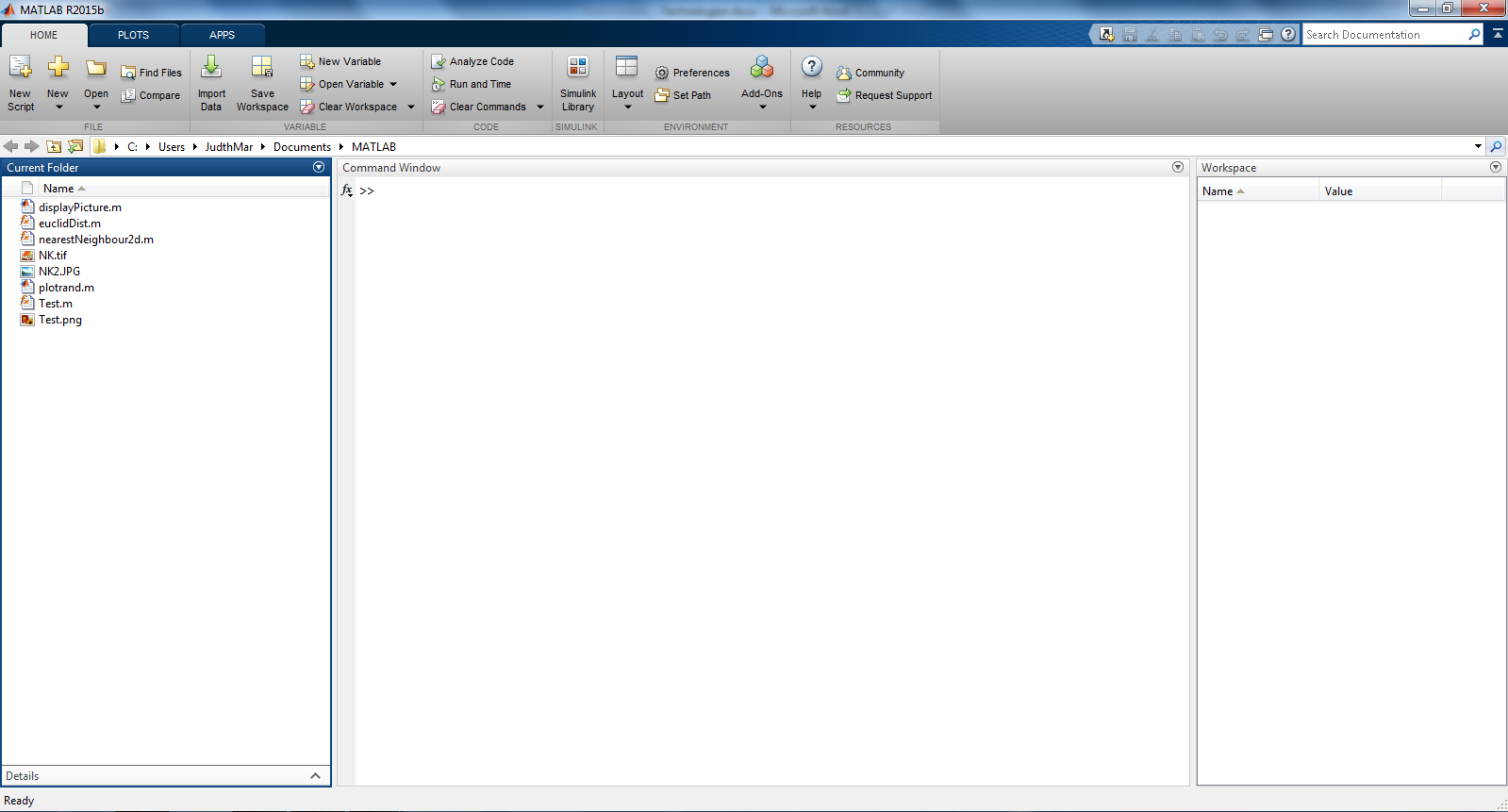


Abbildung 1: MATLAB default Layout

MATLAB enthält ein Command Window wo alle Befehle eingegeben werden und einen Workspace wo alle Variablen angezeigt werden. Außerdem befindet sich auf der linken Seit der Current Folder, wo alle Dateien angezeigt werden.

## Erstellen einfacher Variablen:

>>a = 3

Es wird automatisch a = 3 ausgegeben und um die automatische Ausgabe zu verhindern muss man ein Semikolon (‚;‘) einfügen.

Beispiel:

>>a=3;

🡪 Keine Ausgabe

>>b=4;

>>c=a+b

Ausgabe: c=7

## Vektor

Ein Vektor kann folgendermaßen realisiert werden:

>>arr=[1 2 3 4 5]

arr = a = *1×4*

1 2 3 4

## Matrizen

>>mat=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]

mat = 4×4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Sinus berechnen in einer Matrix:

>>sin(mat)

ans =

0.8415 0.9093 0.1411 -0.7568

-0.9589 -0.2794 0.6570 0.9894

0.4121 -0.5440 -1.0000 -0.5366

0.4202 0.9906 0.6503 -0.2879

Ausgabe einer Matrix mit „‘ “:

>> mat'

ans =

1 5 9 13

2 6 10 14

3 7 11 15

4 8 12 16

Jeden Wert der Matrix um eines erhöhen

>> mat + 10

ans =

11 12 13 14

15 16 17 18

19 20 21 22

23 24 25 26

### Multiplizieren von Matrizen:

>> erg = mat.\*mat

erg =

1 4 9 16

25 36 49 64

81 100 121 144

169 196 225 256

>> erg = mat \* inv(mat)

erg =

2 0 0 0

0 -4 0 8

0 -8 -4 0

8 -8 0 16

Diese Matrix wird statt Integer als floating-point value gespeichert. Mit „format“ kann der gewünschte Datentyp ausgewählt werden.

>> format long

>> P=mat \* inv(mat)

P =

2 0 0 0

0 -4 0 8

0 -8 -4 0

8 -8 0 16

### Zusammenführen von 2 Arrays:

* Vertikal anhängen:

>>mat2=[mat,mat]

* Senkrecht anhängen:

>>mat2=[mat;mat]

Mit magic(n) wird eine Matrix erstellt die die Werte 1 bis n2 beinhaltet und so angeordnet sind das jede Reihe, jede Spalte und die ersten Diagonalen dieselbe Summe bilden.

>>magic(3)

Mat=3x3

8 1 6

3 5 7

4 9 2

### Einzelne Werte auslesen:

* >>Mat(8)

ans = 4

* >>Mat(3,2)

ans = 9

* >>Mat(1:2,2)

ans = 2x1

1

5

* >>Mat(: , 2)

ans = 3x1

1

5

9

Mit „whos“ kann der Inhalt vom Workspace angezeigt werden und dadurch wird es im „Workspace-Pane“ angezeigt.

>> whos

Name Size Bytes Class Attributes

A 4x4 128 double

B 3x5x2 240 double

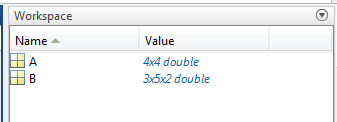


Abbildung 2: MATLAB Workspace whos

## Speichern und Laden:

Mit „save“ wird der Workspace im aktuellen Ordner gespeichert und kann wieder mit „load“ geladen werden.

>>save myFile.date

>>load myFile.date

Characters:

>>text = ‚Hello‘

text = ‚Hello‘

>>text2 = ‚World‘

text2 = ‘World’

>>comText = [text, '-', text2]

comText = ‘Hello-World’

Mit num2str(number) werden Nummern in Characters umgewandelt.

>> infoText = ['I am ', num2str(17), ' years old']

infoText = ‘I am 17 years old’

## Array-Funktionen

Das Maximum eines Vektors wird mit max(vektor) gezeigt.

>>arr1 = [2 4 5 1]

>>arr2 = [1 6 3 2]

>>max(arr1)

ans = 5

>>max(arr1, arr2)

ans = 2 6 5 2

>> [max,location] = max(arr1)

max = 5

Location = 3

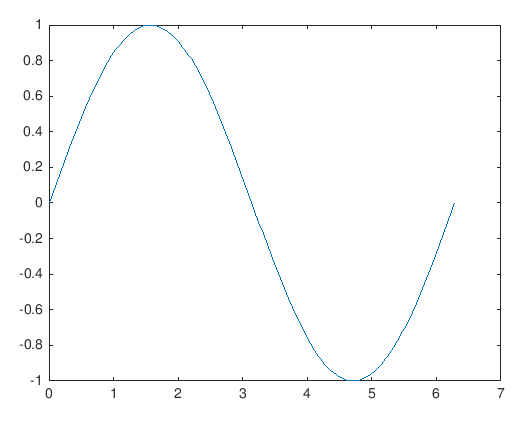
## 2D-Plot

Plots werden mit der „plot“ Fuktion gezeichnet. Dieser wird der X-Wert und der Y-Wert übergeben. Dieses Plot wird danach grafisch dargestellt.

>> x = 0:pi/100:2\*pi;

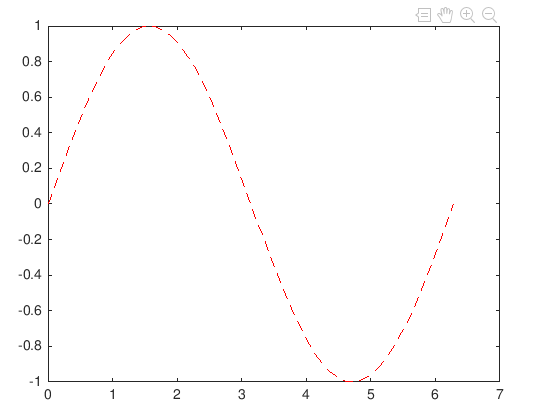
>> y = sin(x);

>>plot(x,y)



Außerdem kann die Linienart mit einem dritten Parameter verändert werden.

>> plot(x,y, 'r--')

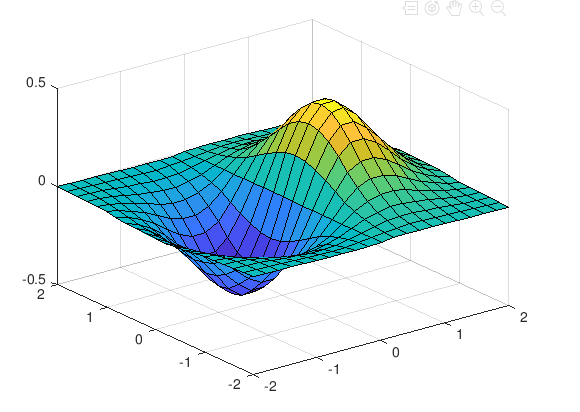


## 3D-Plots

>> [X,Y] = meshgrid(-2:.2:2);

>>Z = X .\* exp(-X.^2 - Y.^2);

>>surf(X, Y, Z)



Programm:

Um ein neues Script zu erstellen wird der Befehl „edit“ verwendet. Dieses Script kann über den Run-Button oder über die Kommando-Zeile ausgeführt werden.

## Schleifen

### For-Schleife:

Beispiel:

for i = 1:10

X = [num2str(i),'. Durchgang'];

disp(X);

end

## Funktionen

Neue Funktionen werden durch einen Rechtsklick im „Current Folder“ 🡪 New File 🡪 Function erstellt.

function y = Test(x)

y = x + 100;

end

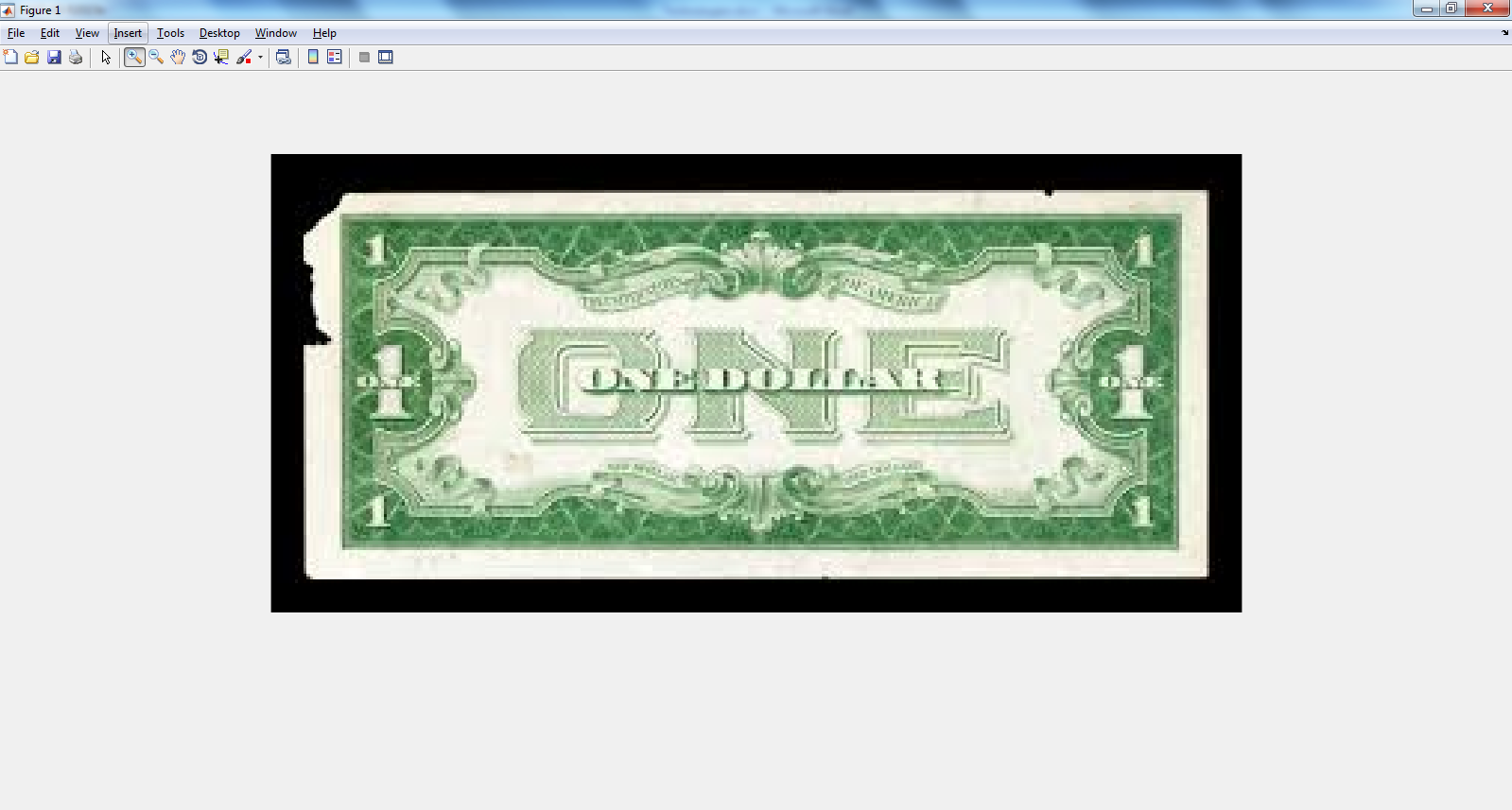
## Bildverarbeitung

Um ein Bild einzulesen wird die Funktion „imread“ verwendet und um dieses Bild danach auszugeben, wird „imshow“ verwendet. Beispiel:

img = imread('Test.png');

imshow(img);

Ausgabe:



Dadurch dass das Bild in eine Matrix gespeichert wird, kann man mit der Funktion „rgb2lab“ in verschiedene Graustufen gelapped werden. Dadurch könne n die Bilder besser verglichen werden.

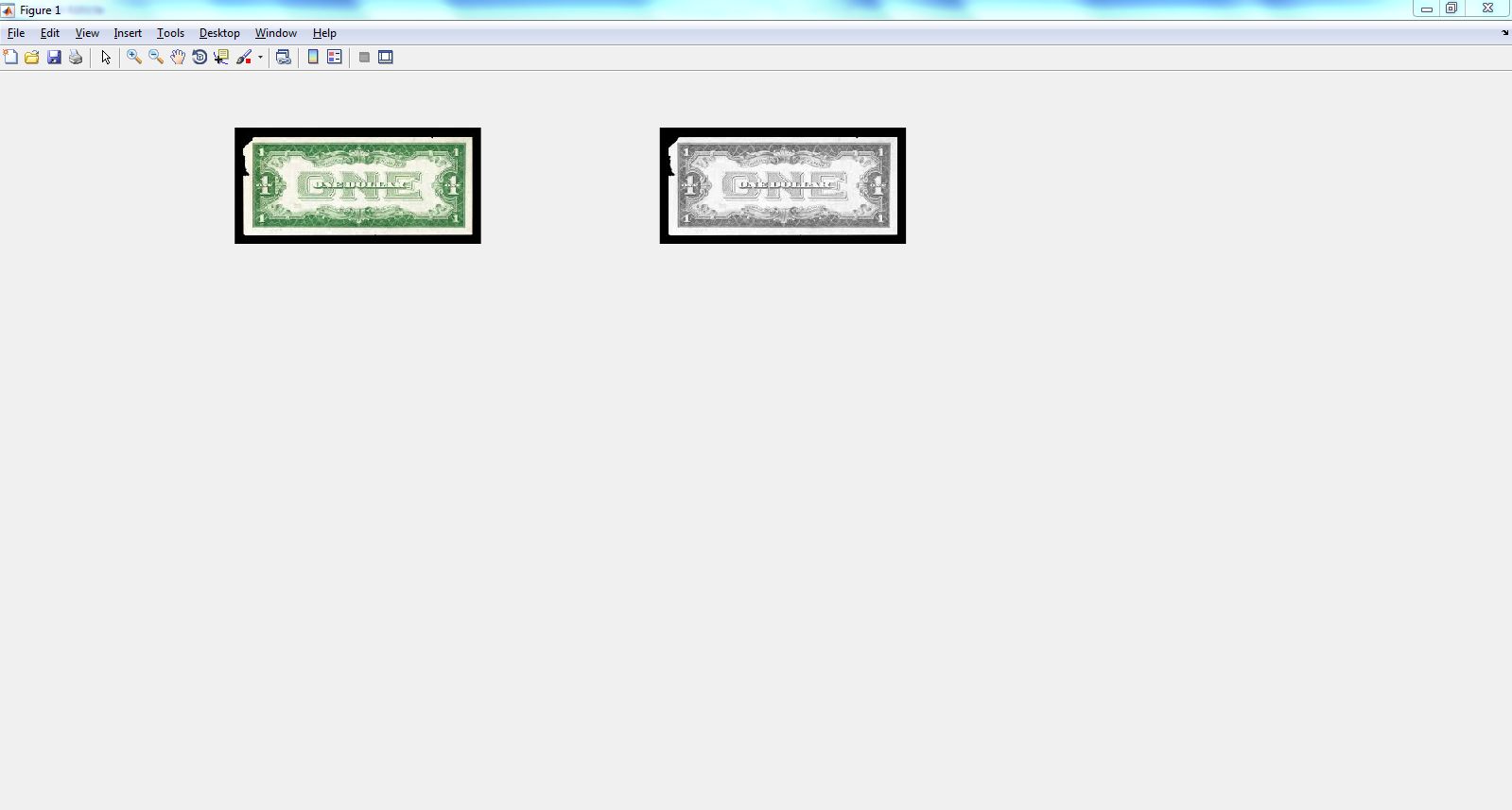
labimg = rgb2lab(img);

subplot(4,3,2);

imshow(labimg(:, :, 1), []);

Mit subplot können mehrere Bilder nebeneinander angezeigt werden. Der erste Parameter beschreibt die Anzahl der Bilder in einer Reihe und der zweite Parameter ist die Anzahl der Reihen. Dadurch können Bilder schnell verglichen werden.

Ausgabe:



Nach diesem wichtigen ersten Schritt wird das Bild klassifiziert und dafür werden die Koordinaten für die einzelnen Klassen benötigt. In diesem Beispiel gibt es einen Hintergrund, den grünen Farbbereich und den weißen Farbbereich. Durch die Koordinaten dieser Bereiche werden die Pixelwerte aus dem neu erstelltem Bild entnommen und in einem mehrdimensionalen Array abgespeichert.

white = labimg(72:82, 51:62, :);

gray = labimg(30:31, 88:91, :);

bgrd = labimg(5:10, 49:52, :);

classes = [[mean2(white(:, :, 2)), mean2(white(:, :, 3))]', [mean2(gray(:, :, 2)), mean2(gray(:, :, 3))]', [mean2(bgrd(:, :, 2)), mean2(bgrd(:, :, 3))]'];

Mit diesem Array „classes” wird im labimg überprüft, zu welcher Klasse jedes Pixel gehört und daraufhin werden die zugeteilten Klassen in ein neues Array gespeichert.

[height, width, channels] = size(labimg);

for i = 1:height

for j = 1:width

x = labimg(i, j, 2:3);

classimg(i, j) = nearestNeighbour2d(labimg(i, j, 2:3), classes);

end

end

function y = nearestNeighbour2d(x, classes)

dists = zeros(size(classes(1, :)));

for i = 1: size(classes(1, :), 2)

dists(i) = euclidDist(x(1), classes(1, i) + euclidDist(x(2), classes(2, i)));

end

[~, y] = ismember(min(dists), dists);

end

function d = euclidDist( x, y )

d = (x - y) ^2;

end

# Quellen:

1. „MATLAB A PRACTICAL INTRODUCTION TO PROGRAMMING AND PROBLEM SOLVING – SECOND EDITION“ von Stormy Attaway
2. Mathworks, <https://de.mathworks.com> 13.07.2018