Technische Universität München
Institut für Informatik
Prof. Dr. H.-J. Bungartz
Dipl.-Tech. Math. S. Schraufstetter
Benjamin Peherstorfer, M.Sc.

WT 2010/11

October 22nd, 2010

Introduction to Matlab

Engineering Informatics I

— *Exercises marked with * are for already advanced programmers.* —

1) Getting Started...

a) Enter in the command window the following lines step by step:

```
>> doc
>> help plot
>> 3+4;
>> 3+4
>> x=ans
>> y=3*x^2
>> f = inline('3*x^2')      % let's define a function
>> f(7)
>> z =[1:7]
>> f(z)                     % don't worry about the result!
>> f = inline('3*x.^2')
>> f(z)
>> clc
```

b) Now, open a new script (*file menu* → *new* → *script* or button "*new script*").

Enter some commands from the slides of the lecture, store the script, and execute it (*debug menu* → *save and run* or button "*save and run*"). It is recommend to omit the semicolons at the end of the commands for once to get an output at the command window when executing the script.

2) Writing Functions

a) Write a Matlab function

```
function c =  fahrenheit2celsius(f)
    ...
```

that converts a temperature from Fahrenheit scale to Celsius scale and store it as an m-file `fahrenheit2celsius.m`.

The formula for converting an Fahrenheit value f to a Celsius value c is given by

$$c = \frac{5}{9} \cdot (f - 32).$$

b) Try to evaluate the function written in a) for $f = 33.8$ in two different ways:

1. The simplest way is directly calling the Matlab function: `fahrenheit2celsius(33.8)`.
2. Another possibility is to use the command `feval`. There exist two possibilities:
   (i) Pass the function name as a string `'fahrenheit2celsius'` as input parameter to the `eval` function.
   (ii) Construct a function handle for the function `fahrenheit2celsius` using the `@` operator and pass this function handle as input parameter to the `eval` function.

   For details, please refer to the documentation of the `eval` command, entering `doc eval` in the command window. Both ways might apper inconvenient, but for more complicated programs, where you have to pass function names, the `eval` function has to be used.

c) Write a Matlab function

```
function f =  celsius2fahrenheit(c)
   ...
```

stored as `celsius2fahrenheit.m` that computes the inverse conversion from Celsius to Fahrenheit scale. Call the nested expression

`celsius2fahrenheit(fahrenheit2celsius(f))`

for some arbitrary values f. What do you expect to get as result?

3) Programming Structures: Loops and If-Statements

a) Write a Matlab function

```
function mean =  arithmeticMean(x)
   ...
```

stored as `arithmeticMean.m` that computes the arithmetic mean value

$$\bar{x} = \frac{1}{n} \cdot (x_1 + x_2 + \ldots + x_n)$$

of a vector $x = (x_1, x_2, \ldots, x_n)$. The length n of a vector can be determined by the command `length`.

Test the function with some arbitrary input vectors `x`. You can compare the results using the Matlab command `mean`.

b) Add an if-statement at the begin of the function which sets the result to 0 and returns an error message, if the length of the input vector is 0. Otherwise, the mean value shall be computed and returned as result.

Messages can be displayed to the command window with `disp('This is a message.')`, for error messages use the command `error('Ups, there occured an error!')`.

4) Vectors and Matrices

a) Warm-up:

1. Create a 5×5 matrix A with random numbers between 0 and 1.
2. Create a 5×5 matrix B with random numbers between e and π.
3. Subtract B from A and store the result in C.
4. Take the square root of all entries in C.
5. Clear the three matrices A, B, and C.

b) Create a vector x with 100 entries between 1 and 100 and dimension 100×1.

Check the dimension with the command `size` which returns the dimension of a matrix. You can transpose (i.e. "flip") a vector `x` to get the desired dimension by applying the transpose-operator: `x'`.

c) Use the command `magic` to define A as the *magic matrix* of dimension 10×10.

1. Store in a matrix B the elements of the 5×5 submatrix of A which consists of the elements a_{ij} with $1 \leq i \leq 5$ and $1 \leq j \leq 5$.
2. Store in a matrix C the first and last column of A.
3. Can you delete the third column of A?

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,5} & \cdots & a_{1,10} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,5} & \cdots & a_{2,10} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{5,1} & a_{5,2} & \cdots & a_{5,5} & \cdots & a_{5,10} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{10,1} & a_{10,2} & \cdots & a_{10,5} & \cdots & a_{10,10} \end{pmatrix}$$

$$B = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,5} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,5} \\ \vdots & \vdots & & \vdots \\ a_{5,1} & a_{5,2} & \cdots & a_{5,5} \end{pmatrix}, \qquad C = \begin{pmatrix} a_{1,1} & a_{1,10} \\ a_{2,1} & a_{2,10} \\ \vdots & \vdots \\ a_{10,1} & a_{10,10} \end{pmatrix}$$

d) Solve the following system of linear equations:

$$\begin{array}{rcrcrcr} 30x & + & 14y & - & 10z & = & 112 \\ 222x & - & 21y & + & 41z & = & -5 \\ -2x & + & \frac{1}{3}y & - & z & = & 3 \end{array}$$

First, define a matrix A and a vector b that correspond to the given linear system. Then, use the backslash operator to solve the system $Ax = b$.

e)* Compute the sum of the elements on the *antidiagonal* of the 4×4 magic matrix. The antidiagonal is the diagonal going from the lower left to the upper right element. Use a `for`-loop.

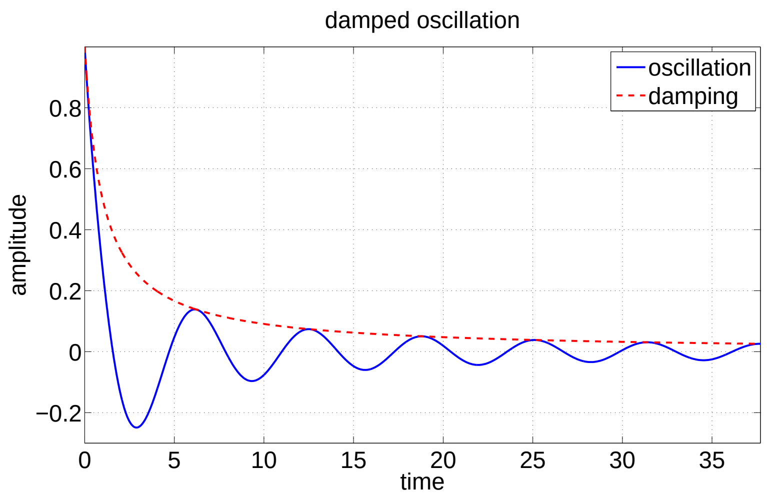


Figure 1: Visualization 2D: Damped Oscillation

5) Visualization 2D: Damped Oscillation

a) Plot the function $f(x) = \frac{\cos(x)}{x+1}$ on the interval $[0, 24\pi]$:

1. Generate a vector `x` with values between 0 and 12π and a resolution of 0.02 with the colon operator '`:`'.
2. Define a vector `y` containing the function values of $f(x)$ that correspond to the values of `x`. For element-by-element operations, do not forget to use '`.`' before operators.
3. Use the command `plot` to visualize the function.

b) Add the function $g(x) = \frac{1}{x+1}$ to the plot:

1. Compute a vector `z` containing the function values of $g(x)$ that correspond to the values of `x`.
2. Visualize this function with a red dashed line. Do not forget to hold the previous plot before! With the command `help plot` you get information about the formatting parameters.
3. Restrict the current view to the x-interval $[0, 12\pi]$ and the y-interval $[-0.3, 1.0]$ using the command `axis` and add grid lines with the command `grid`.

c) Add title, labels and a legend as shown in Fig. 1).

6) Visualization 3D: Monkey Saddle

Plot the function $f(x, y) = x^3 - 3xy^2$ for $(x, y) \in [-1, 1] \times [-1, 1]$ (cf. Fig. 2):

a) Define axis vectors `x` and `y` with a resolution of 0.05 and generate a meshgrid `[X,Y]`.

b) Compute the function values `Z` on the meshgrid `[X,Y]`. You could also use the command `inline` first to define function `f` and then evaluate the function with the simple expression `Z=f(X,Y)`. Anyway, do not forget to use '`.`' before operators for element-by-element operations.

c) Plot the function. Try different commands such as `surf`, `plot3`, and `mesh`. With the button "*rotate 3D*", you can rotate 3D figures.

d) Estimate the saddle point and mark it with a cross in the 3d plot. To be able to see the cross, you can add the options `'Linewidth',5,'MarkerSize',20` at the end of the plot command.

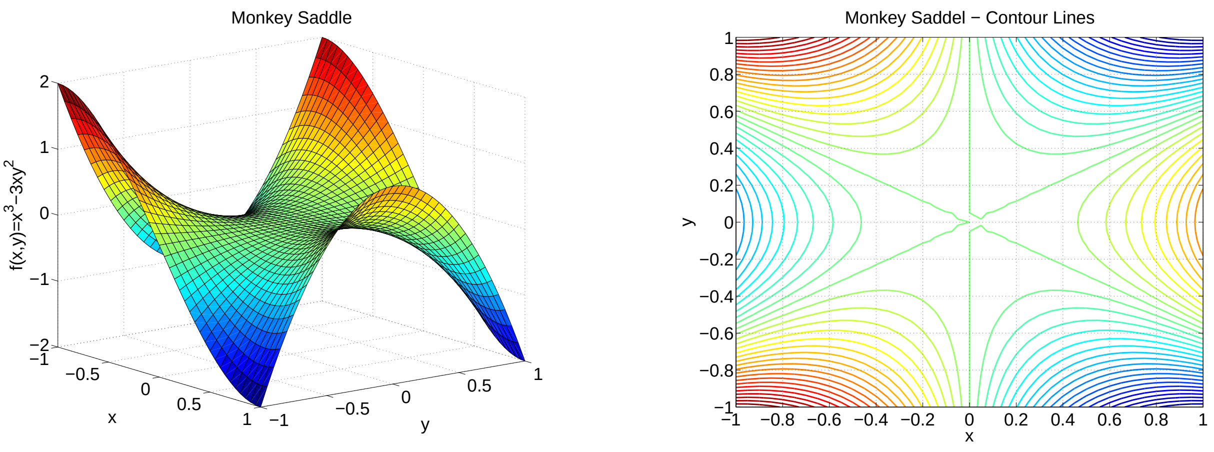


Figure 2: Visualization 3D: Monkey Saddel

e) Open another plot without closing the figure of c) and plot the contour lines with the command `contour`. Try to modify the contour-plot command such that the contour lines for the values `-2:0.1:2` are plotted.

7) Working with Built-in Matlab Functions

a) Use the Matlab documentation to find commands for solving the following problems:

1. Find the greatest common divisor of 1664 and 1792.
2. Find the prime factors of the number 3120.
3. Define a string `s='I can not do string replacement.'` and replace the string `'not'` by the string `'now'`.

b) Consider the polynomial

$$p(x) = x^3 - 6x^2 + 11x - 6.$$

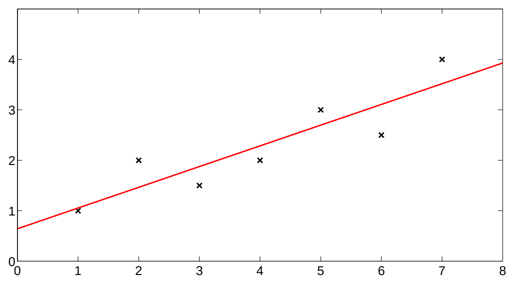
1. Find the roots (“Nullstellen”) of $p(x)$ using the command `roots`.
2. Compute the values of the polynomial $p(x)$ at $x = -1, 0, 1, 2, 3, 4$. Use the function `polyval`.
3. Integrate the polynomial $p(x)$ analytically with `polyint`. What is the result? Write down the result on your sheet!
4. Compute the derivative of $p(x)$ analytically with the help of the command `polyder`. What is the result? Write down the result!

8)* Application: Regression

In this exercise, we want to determine and visualize the line of best fit ("Lineare Ausgleichsgerade") for the following points (x, y):

x	1	2	3	4	5	6	7
y	1	2	1.5	2	3	2.5	4

The final result will be as shown in the following figure:



The line of best fit can be determined by solving the linear system of equations

$$B\,z = b \tag{1}$$

with

$$B = A^T \cdot A \text{ and } b = A^T y \tag{2}$$

and

$$A = \begin{pmatrix} | & 1 \\ x & \vdots \\ | & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ \vdots & \vdots \\ 6 & 1 \\ 7 & 1 \end{pmatrix}. \tag{3}$$

The result vector `z` contains the slope m and the axis intercept t ("Steigung und Achsenabschnitt") of the line of best fit

$$f(x) = mx + t. \tag{4}$$

Remark: A^T is the transposed of A which means to mirror all matrix elements across the diagonal. In Matlab, use the command `A'` to get the transposed matrix A^T. This Matlab operator also changes horizontal vectors to vertical vectors and vice versa.

Now, find the line of best fit:

1. Open a new Matlab script and define vectors `x` and `y` that contain the values of the table.
2. Plot all points (x, y) in a 2d plot using the command `plot`.
3. Define the matrix A as given in Eq. (3).
4. Compute the matrix B and the vector b of Eq. (2).
5. Solve the linear system of equations (1) with the backslash operator $\backslash$ and store the result in a new vector `z`.
6. Define the slope `m` as the first component of the resulting vector `z` and the intercept `t` as its second component.
7. Finally, add a plot of the line of best fit (4) to the plot with the given points generated in subtask 2.

9)* Application: Interpolation

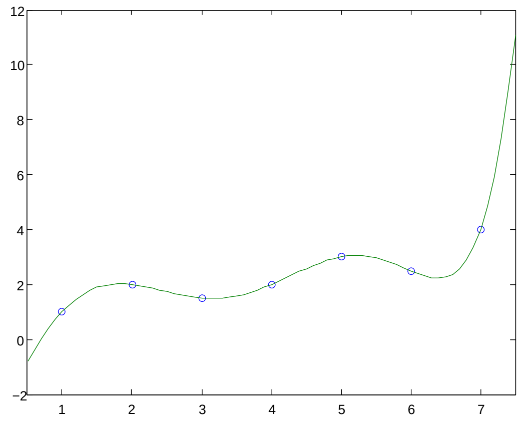
We want to interpolate a given data set by a polynomial, i.e. to find a polynomial

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{5}$$

for some given points $(x_0, y_0), (x_2, y_2), \ldots, (x_n, y_n)$ such that $p(x_0) = y_0, p(x_1) = y_1, \ldots, p(x_n) = y_n$. Consider again the data set

x	1	2	3	4	5	6	7
y	1	2	1.5	2	3	2.5	4

.



a) Polynomial interpolation using the built-in function `polyfit`

1. Create vectors $x = (x_0, \ldots, x_6)$ and $y = (y_0, \ldots, y_6)$ with x_i, y_i from the table above.
2. Use `polyfit` to compute the coefficients $(a_0, \ldots, a_6)$ of the interpolation polynomial (5).
3. Plot the polynomial in the interval $[0.5, 7.5]$. Use the command `polyval` for evaluation.
4. Add all data points $(x_0, y_0), \ldots, (x_6, y_6)$ to the plot and check if the polynomial goes exactly through these points.

b) Polynomial interpolation with the *Vandermonde* matrix
For a given data set $(x_0, y_0), \ldots, (x_n, y_n)$, we have to find $n + 1$ coefficients a_i of the polynomial (5), such that $p(x_i) = y_i$. This means, we have to solve a system of $n+1$ linear equations. The corresponding matrix is called *Vandermonde matrix.*

$$\begin{pmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0^1 & x_0^0 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1^1 & x_1^0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n^1 & x_n^0 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{6}$$

1. Create a function with parameters x and n which returns the Vandermonde matrix:
   a) Create a new function with two parameters and the return value A.
   b) The dimension of the Vandermonde matrix is $(n+1) \times (n+1)$. Initialize a matrix A of this dimension, e.g. with the command `zeros`.
   c) Use nested `for`-loops to fill the matrix with the values as shown in (6).
2. Rewrite the function with only one parameter x and use the command `length` to obtain the dimension n of the Vandermonde matrix.
3. Use the function to compute the Vandermonde matrix for our data set and solve the system of linear equations with the backslash operator.
4. Plot the result.

10)* Application: Approximation

If a function f satisfies certain conditions, it can be represented by its Taylor series in the neighborhood of a point. The Taylor series of $\sin(x)$ around the point $x = 0$ is given by

$$\sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots.$$

In this exercise, we want to compute values of $\sin(x)$ by using this Taylor series. Since we cannot compute an infinite sum, we only take finitely many terms into account.

Write a function `approxsin` which computes this sum:

1. The evaluation point x and the maximal exponent n are parameters of the function.
2. The approximated value of the sum is returned as result.
3. To construct a `for`-loop which iterates over the sequence $1, 3, 5, 7, 9, \dots$, you can use the colon operator : with step size 2.
4. To construct the alternating sign, you can define a variable s with initial value 1 and multiply it with -1 after every iteration.
5. The factorial denoted by $m!$ can be computed with the built-in function `factorial(m)`.

Compute some values with `approxsin` and compare them with the values of the built-in function `sin`. What do you observe? Maybe a plot helps?