Aschaffenburg University of Applied Sciences
Faculty for Engineering
Prof. Dr. Michael Möckel

Email: Michael.Moeckel@th-ab.de
Telephone: +49 (0)6021 / 4206 - 507

Numerical Methods and Simulation
Module MT 03
Winter term 2022/23

`https://moodle.th-ab.de/course/view.php?id=624`

Problem Set 2

Programming in MATLAB

Exercise 1: Defining and plotting functions in 2D and 3D

MATLAB provides relatively simple ways to define and plot functions.

(1) Simple polynomials. Implement the function $f(x) = x^2 - 4x + 4$ as an anonymous function. The syntax for this is `f = @(x)` followed by the right hand side of the above equation. Evaluate the function at 200 points in the range [0:4] by creating a suitable discretization of the x-axis with `x=linspace(0,4,200)` and plot the function. Then repeat the evaluation with `[X,Y] = fplot(f, [0 4])` and plot the result using `plot`. Add the usual labels to the plot.

(2) Polynomials can also be defined by a vector of their coefficients. The number of elements of the vector must agree with the degree of the polynomial. Missing powers must be represented by coefficients of value 0. For instance, the polynomial $g(x) = x^2 + 1$ is represented by a vector `p=[1,0,1]`. The polynomial p can be evaluated at position x by `polyval(p, x)`. Now set up the vector p for the function $f(x)$ in (1), and evaluate and plot this function as a polynomial again.

(3) 3D Plotting. Plot the function $f(x, y) = \sin(2 * x) * \cos(y)$ in the range $x, y \in [0, 2\pi]$ using a surface plot (`surf(X,Y,Z)`). To do so, set up x and y as vectors of points within the given range and with a stepsize of 0.25. Create a meshgrid by `[X, Y] = meshgrid(x,y)`) and inspect the resulting matrices X and Y. What do they contain? Why are they packed with redundant information? Why can't you enter x and y directly into the `surf` function?

(4) Create a function file `hemisphere.m` for implementing the function `hemisphere(x,y,R)` of the positive hemisphere with radius R around the origin of the x-y plane (otherwise 0). The function describing the surface of a sphere around the origin is given by

$$z = S_R(x, y) = \begin{cases} \pm\sqrt{R^2 - x^2 - y^2} & : \quad x^2 + y^2 < R^2 \\ 0 & : \quad \text{otherwise} \end{cases}$$

First write a function which returns the correct result for a pair of numbers (x,y). Run it by calling `hemisphere(a,b,R)` for some values a,b and R=1 in the command window.

(5) Extend the function in `hemisphere.m` to also accept vectors and matrices as inputs. Consider what happens when you enter the output matrices of the meshgrid routine. Then pointwise operation of the equation must be implemented. To implement the condition of the equation one needs a little trick: Defining $L = (x.^2 + y.^2 < R^2)$ creates a matrix L with elements of '1' (true) and '0' (false). How could this help? Plot the function for R=1 in the interval $x, y \in [-1, 1]$ by creating a meshgrid, evaluating the function on the meshgrid and using a 3D surface plot (`surf(X,Y,Z)`).

Exercise 2: Simple programming (scripting) in MATLAB: Converting units of energy

Write a little MATLAB script program (filename EConv.m) which converts energy units into each other. The conversion factors are approximately 1 J(Joule) = 0.239 cal (calories) = 6.24 $10^{18} eV$ (electron volts). The program should take as inputs

(i) the unit of energy before conversion

(ii) the value of energy in this unit, and

(iii) the unit after conversion

The program should contain

(iv) a helpful error output if the given input does not match the possible choices

(v) a switch-case selection for processing the possible requests

(vi) and an output of the result

Exercise 3: Efficient programming in MATLAB

MATLAB provides access to the CPU time to measure the computational cost of an calculation. This is done by starting a clock with `tic` and ending it after the computation has been performed with `toc`.

(1) Approximate the infinite sum $\sum_{i=1}^{\infty}\left(\frac{1}{2^i}+\frac{1}{5^i}\right)$ by a numerical calculation replacing $\infty \to N = 100,000$.

(1.1) Which result do you expect?

(1.2) Calculate the result using a loop and measure the computational time

(1.2) Calculate the result using a vectorized approach based on the index vector $\vec{i}$ and measure the computational time

(2) A Hilbert matrix is a special matrix defined by $H_{ij} = \frac{1}{i+j-1}$. Measure the time for creating a Hilbert matrix of size 10,000 $\times$ 5 using different methods:

(2.1) By using an outer and an inner loop over the matrix indices for initializing all matrix elements separately (test both possible choices)

(2.2) By pre-allocating memory for all matrix elements initially. This can be done by starting with a suitable matrix of zeros or ones and replacing the matrix elements.

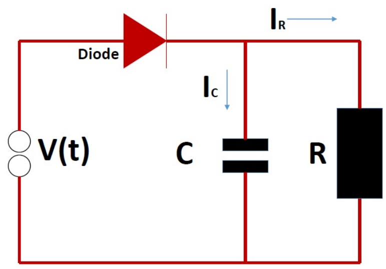
(2.3) By a combination of pre-allocated memory and vectorization.

Exercise 4: Modelling a simple AC to DC converter (by A. Gilat)

A simple half-wave rectifier can be realized from a diode, a capacitor (capacity C) and a resistor (resistance R). An AC voltage $V(t) = V_0 \sin(\omega t)$ with an amplitude V_0 is applied where $\omega = 2\pi f$ represents the frequency f of the sinusoidal oscillation. At a capacitor, voltage $V_C(t)$, charge $Q_C(t)$ and capacity C are related by $V_C(t) = Q_C(t)/C$. The current I_C through the capacitor is the time derivative of the charge Q_C. The rectifier operates in the following way:

Case 1: During the charging of the capacitor the external voltage $V(t)$ and V_C agree. Now the diode is *on* and the charge on the capacitor increases. The current through the resistor follows Ohm's law: $I_R = (V_0/R)\sin(\omega t)$.

Case 2: At some time t_A the external voltage drops below the voltage at the capacitor $V_C = Q_C(t)/C$. Then the direction of the current I_C is reversed. Without a diode, the charge on the capacitor would flow back into the voltage source. Yet the diode blocks the current and the capacitor discharges through the ohmic resistance. With charge flowing off the capacitor its voltage is also decreasing.



Repeat: At a later time t_B (almost a period later) the external voltage $V(t_B)$ increases beyond the then voltage at the capacitor $U_C(t_B)$. Then the capacitor is charged again and a new cycle begins.

1. Modeling: Understand the currents and voltages in the AC to DC converter qualitatively. Sketch the applied oscillating voltage $V(t)$ for ≥ 0 and include two suitable times t_A and t_B. We are interested in the behaviour of the voltage drop at the resistor in time $V_R(t)$. For a quantitative understanding a numerical simulation is helpful.
2. Mathematics: Derive the equation for the voltage at the resistor for any time in case 2. Result: $V_R(t > t_A) = V_0 \sin(\omega t_A) e^{-(t-t_A)/RC}$
3. Programming: Write a MATLAB program that calculates and plots the voltage across the resistor $V_R(t)$ and the external voltage $V(t)$ as a function of time. Use the parameters $0 \leq t \leq 70$ ms , $R = 1.8k\Omega$, $V_0 = 12V$, f = 60 Hz and C = 45 μF.
4. Observation: Compare with a simulation for C = 10 μF.
5. In practice, the value of the resistance depends on the load at the AC-DC converter. Observe how the quality of the DC voltage changes with reduced resistance (=increased load).

References:

[1] Amos Gilat, MATLAB, An Introduction with Applications, Wiley 2015