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`

Organizational Matters

(1) **Moodle:** Please register on moodle (`https://moodle.th-ab.de/`) with the course *Mathematik III für MT 3*. Materials will be provided via the moodle platform.

(2) **Getting in touch:** For questions related to the course material there is a user forum on the moodle site. Check out if someone else inquired your question there already. For other questions please get in touch (`Michael.Moeckel@th-ab.de`).

(3) **Books and online ressources:** Recommendations are available on the Moodle course site.

(4) **Prerequisites:** This course aims at implementing mathematical structures in high level computer code. This is wonderful as the computer will solve tedious math problems! However, familiarity with some (higher) mathematics will be required. I strongly recommend to review the essential terms of linear algebra (vectors, matrices, special matrices, matrix calculus, diagonalization of matrices, etc.) and analysis (polynomials, trigonometric functions, curve sketching, differential equations, etc.) whenever needed.

(5) **Hands-on example classes:** Example classes will take place in the computer room 40-140. Please log into the machines before the lecture starts. All computer rooms are accessible for doing exercises during daytime whenever there is no other course. I strongly recommend to install MATLAB on a personal notebook or computer if available. The campus license for MATLAB also covers student owned machines.

(6) **Exercises:** Problem sets will be handed out every week. They will partially be discussed in the example classes. It is recommended to store your code in your file system. Create a new directory for every problem set (e.g. Probs1, Probs2, etc) and start a new m-file for every problem.

(7) **Exam:** There will be one written exam at the end of term (90 minutes). No access to computers, handbooks, lecture notes etc. will be allowed. The exam may contain a small programming task, so you should memorize essential coding techniques. This is best done by doing the exercises continuously. It is not considered sufficient to start preparing for the exam after the Christmas break.

Problem Set 1

Getting familiar with MATLAB

Coursework

Exercise 1: MATLAB as a luxury calculator

1. Screen appearance. Start MATLAB and, firstly, identify the command window, the workspace window and the current folder manager. Find the tab *Default* in the *Layout* menu. Whenever your windows get messed up this will quickly return you to the default screen.
2. Interrupt. You can interrupt any calculation using Strg + C (except in parallel mode –which we will not use–). When working on two screens there can be tricky issues. The command window has to be on screen one to accept interrupts.
3. Logging. Start logging your session such that you can review the course later. Enter into the command window

   ```
   diary Lecture_1_LOG.m
   ```

   Then all your input will be stored in this file. You can switch off the logging using *diary off*. Then you can switch to a different file (e.g. a proper script) by ending the old and starting a new log file, e.g. for creating your own set of helpful examples and remarks

```
diary Lecture_1_SCRIPT.m
```

4. Variables. Variables do not need to be declared. Just initialize them at first use.
5. Basics. Perform some simple calculations: Enter into the command window a=1 (enter) then b=2*a; (enter) then b; (enter). What does the semicolon? What can you read off in the workspace window? Click on the variable symbols!
6. Domain. Define a vector of equidistant points on the x axis by `x = 0:0.1:3;`. This creates a vector. Check its entries in the workspace window. Double the distance between the points by overwriting x with a new definition. Then reduce it by a factor of 5. Extend the upper limit to π (`pi`).
7. Built-in trigonometric functions. Calculate sin(a), cos(pi), atan(1) and `y = sin(x);`. Check y. Note that the last command operates pointwise on the vector x. Pointwise operation on vectors is an important concept in MATLAB.
8. Call the command line help for the sin function using `help sin` (enter). Check out the help page linked to the command line output. Try to structure the given information. Learn how to plot the sin(x) curve from the given example.

Exercise 2: MATLAB as a matrix lab

1. Enter a matrix $M = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \end{pmatrix}$ and a vector $b = \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$ as a list using the following commands (take care with commas and semicolons, commas could also be replaced by blancs.)

```
M = [11, 12, 13; 21, 22, 23; 31, 32, 33; 41, 42, 43]
b = [7; 8; 9]
```

   - Write out individual matrix elements by stating their indices in round brackets, e.g. `M(2,3)`, and change it by `M(2,3) = 0`.
   - Write out all matrix elements of a row using the unspecified index range : operator, e.g. r1 = M(1,:) and define the three column vectors c1, c2, c3 in analogy to the example c2 = M(:,2).
   - Reconstruct a new matrix N = [c1,c3,c2] from the columns of M. What should happen?
   - Construct a square matrix K by cutting off the last row of M using `K=M(1:3,:)`. Compare with the definition of a domain (exercise 1) and explain the syntax of this command.
   - In the command window, use the *up* key to recall the original definition of M. Change the semicolons to commas. What changes? Carefully check the variable M on the workspace window. What does the command `size(M)` tell you? Recall the first definition of M and repeat.

2. The `reshape(old object, new number of rows, new number of columns)` command goes to the heart of the MATLAB philosophy and is a powerful tool in MATLAB. It re-casts a set of numbers into different vector, matrix (or in general tensor) *shapes*. Define alternative representations of the data present in M as M2 (2x6), M3 (3x4), v (12x1), z (1x12).
3. In linear algebra calculus is only defined for vectors and matrices with corresponding shapes. Make sure that M is a 4 by 3 matrix. Then calculate the product of the matrix M and the vector b as S = M * b. What is the shape of S?
4. Define the transpose of the matrix M as `T=M'` (or `T = transpose(M)`). Check the size of T and M. Calculate the matrix products `TM = T * M;` and `MT = M * T;` and compare.
5. Some high level operations are only defined for a subset of matrices, e.g. the determinant for a square matrix S (implemented by `det(S)`) or the inverse for an invertible (nonsingular) matrix (implemented by `inv(A)`). Try to compute both objects for the square matrix K defined by `K=M(1:3,:)`. Explain the problem.

Exercise 3: MATLAB's collection of special matrices

MATLAB provides a large set of routines to initialize special matrices or vectors (socalled constructors). The size of the matrix or vector is declared in round brackets. Try the following examples:

- The zero matrix (containing only zeros) `zeros(2,3)`. What can it be used for?
- A matrix containing only ones `ones(4,4)`. How could you make a matrix which contains 4 at every matrix element?
- The unit matrix (i.e. the diagonal matrix with ones on the diagonal) of size 3 `eye(3)` or `eye(3,3)` or `identity(3)`
- A diagonal matrix with the entries of the vector b on the diagonal (i.e. casting the vector as a diagonal matrix) `diag(b)`. The opposite also works: `w = diag(M)`. What is contained in w?
- Random matrices, e.g. `rand(5,6)`
- Magic matrices, e.g. `magic(3)` Why is it magic? Calculate the sums for each row and column. More complicated matrices (pascal, hadamard, hilbert, toeplitz, etc.) have been implemented, too.

Exercise 4: Notation and some ambiguities

Unfortunately, there are various conventions and some ambiguities in linear algebra.

- Vectors have been denoted by various symbols $\vec{a}, \bar{a}, \mathbf{a}, \mathfrak{a}$. Their components can be addressed by a single index and are often denoted by a_i or a^i. Vectors can be represented as row or column vectors which are related to each other by transposition[1]. In MATLAB, any *name*, e.g. `v, vector1, vectorlong` etc. can represent a vector if it is initialized as a vector (cf Exercise 2). No type control is enforced. Make a list corresponding notations for matrices yourself!
- Aside: Internally, MATLAB treats vectors as degenerate matrices with only one row or column. Therefore, there is no strict distinction between matrices and vectors in most routines. This can be seen by applying the command `ndims()`, which usually returns the number of dimensions (or indices needed to label an object). For matrices M_{ij} two indices i and j are always needed. For vectors, `ndims(v)` returns 2 as well (not 1 as expected).
- Some operations between vectors or matrices can be performed either element-wise or with respect to the full object, e.g. multiplication. Compare the results for `d1 = b.*b`, `d2 = b*b`, `d3=(b')*b`, `d4=dot(b,b)` and `d5=b*(b')`. Which operation(s) represent the scalar product $\vec{b} \cdot \vec{b}$? What do the others?
- Absolute value. Some operations can be easily confused: When applied to vectors, the notation $|\vec{a}|$ takes the absolute values of individual elements, i.e. $(|a_1|, |a_2|, ..., |a_n|)$. The MATLAB command is `abs`. Yet the same MATLAB command applied to a complex number gives what is -correctly- known as the absolute value of a complex number. This is NOT the absolute value of the real part and the imaginary part but the distance of the number from the origin (radius). Compare the vector result `abs(b)` with the `abs(z)` for the complex number `z = 3 - 4i`.
- Norm. On the other hand, the *(2-)norm* (i.e. the length) of a vector is $||\vec{a}||_2 = \sqrt{\sum_i a_i^2}$. Compute `norm(b)` in MATLAB. What is the result of `norm(z)`?

Exercise 5: MATLAB as a tool with limits – roundoff errors

All numerical approaches have their inherent limits. It is important to be aware of this simple fact and to develop some feeling for it to create and run reliable practical applications.

1. Delete all variables and start with a new workspace using `clear all`. If you want to delete a file in the current folder you need to use the right button of your mouse.
2. Floating point representations limit the minimum and maximum range of available numbers and their accuracy. Use `realmin` and `realmax` to find the range of valid numbers (absolute values, the sign is an extra information). Change the output format to `format long e` and repeat. Now all significant digits are displayed (the inverse operation is `format short`). What is the second smallest and the second largest number in MATLAB? What is their distance from the smallest and the largest number, respectively? Note that zero is a special case.
3. The *relative* error can be displayed with `eps`. What is the *absolute* error of the smallest number and of the largest number? Note that both errors are not equal at all!

[1]Mathematically, a row vector is called the *dual* of a column vector.

4. Numerical wipeout of significant digits is a problem when very small and very large floating point numbers are added. For all $x \neq 0$ the following equality holds: $(1+x)-1)/x = 1$. Set `x = 1.0e-15` and calculate the left hand side in MATLAB. What is the relative error?

5. The seriousness of the roundoff error depends on how a calculation is done. Mathematical equivalent routes can lead to different computational errors! Check this for the identity

$$(x-1)^7 = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

Generate a vector of points closely around $x_0 = 1$ by `x = 1-2.0e-8:1.0e-10:1+2.0e-8;`. Consult the `help power` reference for element-wise (pointwise) exponentiation of the elements of a vector x and use this on the vector (x-1) to implement the left hand side as a new vector $y = (x-1)^7$. Note that - implies an element-wise subtraction here! Implement the right hand side as z accordingly. Plot y with `plot(x,y)` and study the scale of the axis carefully. Then insert a second graph into the plot by the commands `hold on` and `plot(x,z)`. Again, study the scale of the axis. What happend?

Additional Homework

Exercise 6: Practising MATLAB thinking

1. Create a matrix M with the following values: $M = \begin{pmatrix} 1.1 & -5.0 & 4.7 \\ -4.0 & 4.7 & -22.3 \\ -4.1 & 2.3 & 19.2 \end{pmatrix}$
2. Create a row vector r with the values 1.6 -16.5 10.5
3. Create a column vector c with the values 0.7 -5.1 -4.8
4. Print the rank, size and array dimensions (`ndims`) of these objects
5. Save the objects to the MATLAB-file Ex6sub1.mat
6. print the element at row 2, column 3
7. Calculate r times M times c using matrix multiplication
8. Calculate c times M times r with suitable transposed objects and compare results
9. Calculate the determinant of M
10. Create a (7,7) Hilbert matrix and define B as the middle three rows (3-5) and last three columns (5-7)
11. Set F to be the second column of B as a row vector
12. Set FM to be the diagonal matrix with the same entries as F
13. Set V to be a row vector of length 5 with all elements 0 using a standard constructor
14. Delete the matrix M from the workspace
15. Load the matrix M from the file.
16. Print the maximum value of each row and the minimum value of each column of M
17. Sum up all matrix elements in M without using loops

Exercise 7: Plotting sets of functions in MATLAB

There is a large selection of predefined plots in MATLAB. Write and run a script *Firstplot.m* with the following features:

- Create a vector x with 1000 values in the interval [0,10].
- Consider the three parameter values $m = 1, m = \pi/2, m = \pi$ and compute the function $f(x) = sin(m * x)$ for all three values of m.
- Plot the function for all three values of m in the given interval into a single two-dimensional plot;
- Add labels on the x and y axis, a title and a legend labeling the three curves with the corresponding values of m.
- Repeat the exercise and treat all values of m simultaneously by creating a vector $[1, \pi/2, \pi]$. Calculate the function w.r.t. an argument of two vectors x and m. This can be done naively. What is the result? Plot the curves. Note the simplicity of array operations in MATLAB!