Technische Universität Bergakademie Freiberg Wintersemester 2019/20

Dr. U. Prüfert
Dipl.-Math. M. Brändel 01.12.2019

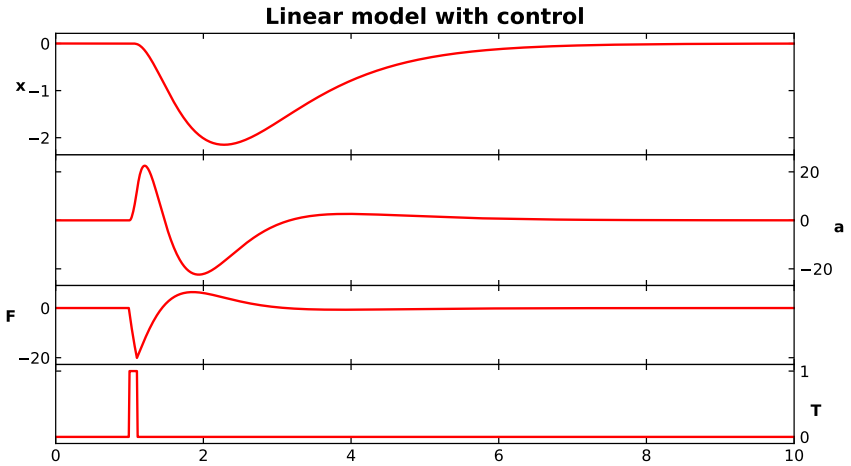
Introduction to Scientific Programming - Programming Project in C++

Description of the project

For the programming project you get an incomplete C++ code that simulates the behaviour of a self-regulating inverse pendulum - a segway.

Programming Task
Complete the C++ code, so that the results from the simulation are correctly displayed in the picture generated by the Python script!

Upon correct execution of the given task, the C++ code generates a text-file "segway.txt", that contains the simulated values for the four unknown entities. These values have then to be evaluated by the provided python script "plot.py", that saves the plot of the entites over time in a PDF file "segway.pdf" and a PNG file "segway.png".



Breaking the project down in parts and subparts

The amount of work to be done in order to complete the programming project can be divided in multiple parts:

Task 1.x Tasks that involve the allocation of variables and assigning correct values to them.

Task 2.x Generating output of variables from Task 1.x and comparing the results to the provided correct values.

Task 3.x Completing the classical Runge-Kutta method.

Task 4.x Implementing the pseudocode given in func_segway.cpp to obtain the (mathematical) function $f(\cdot)$ from the Runge-Kutta method.

Task 5.x Tasks that demand the implementation of algebraic operations. The functions you implement here, are needed for Task 3.x and Task 4.x!

Task 6 Execute the plot.py script to generate the desired picture from the segway.txt file.

Hints

- Complete the tasks in the order you see them in the main.cpp file. The code will not compile successfully, until you have completed at least Task 1.x and Task 2.x, due to usage of variable identifiers, that have not yet been declared!
- The background on the segway covers more than you explicitly need to know about the task in terms of pseudocode, so do not get demotivated. The slides that you will definitely need are 4,5,8 and 13.
- **Do not expect** to be able to fully accomplish all the tasks until you have reached the exercises about classes in the tutorial! Classes have nothing to do with this project, but everything else before might.
- **Do not neglect** the tutorial exercises about classes, they are relevant for the exam!
- **Do not change** anything that has not been highlighted as a task!

Processing

- You can work on this project either alone or with a partner. Groups of three students are not allowed.
- If you choose to work with a partner you need to write a PDF documentation, if you work alone, you do not need to.
- In that PDF documentation you need to
  - briefly explain the code you used to fill in the gaps and

– explain who did what (both partners need to program).

Submit the complete code (as well as the documentation, in case of partners) in one ZIP file. The name of the ZIP file must contain the immatriculation number(s).

List of files

List of provided files, (+) need to be edited, (-) do not need to be edited:

(+) main.c - main function file where you setup the environment for the simulation

(-) func.h - header file

(+) func.c - function file, code for vector manipulation, displayment, ...

(-) func_rk4.h - header file

(+) func_rk4.c - function file, code for classical Runge-Kutta method

(-) func_segway.h - header file

(+) func_segway.c - function file, code for creating the descent vector for the system of ordinary linear differential equations

(-) plot.py - Python script to create the figure from the segway.txt

(-) segway_backgroud.pdf - some information about the model

Details about the simulation

See "segway_background.pdf", a (certainly not complete) variable list:

- tsta, tend - values containing start and end time of the simulation
- framerate - value containing the increment of time steps (how many steps/second)
- n - # of time steps
- tspan - vector containing the time steps
- g,m1,m2,l - model parameters (as described in "segway_background.pdf")
- C - array of coefficients (as described in "segway_background.pdf")
- T - disturbance array, will be filled for you, contains zeros except for time steps $1 \leq t \leq 1.1$, there it contains ones
- A,b1,b2,K - system of ordinary linear differential equations (as described in "segway_background.pdf")
- y - matrix(!) containing the solution vector for the system of ordinary linear differential equations for every(!) timestep
- k1,k2,k3,k4 - temporary vectors needed for the classical Runge-Kutta method, for a more detailed look at the method consult Wikipedia