S. Mishra
K. O. Lye
L. Scarabosio

Autumn Term 2015

Computational Methods for Engineering Applications II

ETH Zürich
D-MATH

Homework Problem Sheet 1

Problem 1.1 Chemical concentrations

In this exercise we will study the evolution of chemical concentrations. Consider a chemical system consisting of four substances A, B, C and D. We have a reversible chemical reaction on the form

$$A + B \underset{k_2}{\overset{k_1}{\rightleftarrows}} C + D.$$

At time $t \geq 0$, we let $u_1(t), u_2(t), u_3(t)$ and $u_4(t)$ denote the concentration of A, B, C and D respectively. The kinetics are given by the following system

$$\begin{cases} u_1'(t) = -k_1 u_1(t) u_2(t) + k_2 u_3(t) u_4(t) \\ u_2'(t) = -k_1 u_1(t) u_2(t) + k_2 u_3(t) u_4(t) \\ u_3'(t) = -k_2 u_3(t) u_4(t) + k_1 u_1(t) u_2(t) \\ u_4'(t) = -k_2 u_3(t) u_4(t) + k_1 u_1(t) u_2(t) \end{cases} . \tag{1.1.1}$$

(1.1a) Write the system (1.1.1) in the form

$$\boldsymbol{u}'(t) = \vec{F}(\boldsymbol{u}(t)) \tag{1.1.2}$$

where $\boldsymbol{u} \in \mathbb{R}^4$ and $\vec{F} : \mathbb{R}^4 \to \mathbb{R}^4$.

(1.1b) We will use the *trapezoidal rule* (or implicit second order Runge-Kutta) to solve the system (1.1.2) numerically. Recall that the trapezoidal rule is given as

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \frac{\Delta t}{2}\Big(\vec{F}(\boldsymbol{v}_n) + \vec{F}(\boldsymbol{v}_{n+1})\Big) \qquad \text{for } n \geq 0, \tag{1.1.3}$$

$$\boldsymbol{v}_0 = \boldsymbol{u}_0, \tag{1.1.4}$$

where $\Delta t > 0$ is the step size.

Determine the truncation error of (1.1.3).

(1.1c) Notice that (1.1.3) is a non-linear equation in $\boldsymbol{v}_{n+1}$, therefore we will use the Newton method to solve for $\boldsymbol{v}_{n+1}$. Recall that the Newton method for solving

$$\vec{G}(\boldsymbol{x}) = 0$$

is formulated as

$$D\vec{G}(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k = -\vec{G}(\boldsymbol{x}_k) \tag{1.1.5}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k \qquad k \geq 0, \tag{1.1.6}$$

where $\boldsymbol{x}_0$ is some initial guess. Write the Newton step for (1.1.3), solving for $\boldsymbol{v}_{n+1}$. What is $\vec{G}$ and $D\vec{G}$ in this case?

HINT: You should *not* compute $\left[D\vec{G}(\boldsymbol{x}_k)\right]^{-1}$ by hand.

(1.1d) Write a function in C++ that computes $\boldsymbol{v}_{n+1}$ using the Newton method. The function should take the following parameters:

- The previous value $\boldsymbol{v}_n$.
- An initial guess.
- The constants k_1 and k_2.
- The step size Δt.
- A tolerance for when to exit the Newton iteration.
- A maximum number of iterations to use for the Newton method.

See the function `newtonSolve` in `template_code1/exercise1/trapezoid.cpp` for a template.

HINT: We strongly suggest that you use the Eigen library for solving the linear system $D\vec{G}(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k = -\vec{G}(\boldsymbol{x}_k)$. Refer to `https://www2.math.ethz.ch/education/bachelor/lectures/hs2015/..other/cmea2/series/eigen_example_dense.zip` for an introduction to using Eigen. Further documentation on Eigen can be obtained on the Eigen homepage `http://eigen.tuxfamily.org/`, and in particular the section `http://eigen.tuxfamily.org/dox/group__QuickRefPage.html`.

HINT: Use $\|\vec{G}(\boldsymbol{x}_k)\|$ as a measure on the error.

(1.1e) Implement the trapezoidal scheme (1.1.3), using the Newton function from the previous exercise. Compute the approximate solution up to time T, using the following constants:

- $\boldsymbol{u}_0 = \begin{pmatrix} 0.2 & 0.3 & 0.1 & 0.4 \end{pmatrix}$
- $k_1 = 0.4$, $k_2 = 0.1$
- $T = 20$
- $\Delta t = T/1000$

Plot u_1, u_2, u_3 and u_4 as a function of time. Also plot the sum $u_1 + u_2 + u_3 + u_4$ as a function of time.

See the function `main` in `template_code1/exercise1/trapezoid.cpp` for a template.

Problem 1.2 Explicit vs. Implicit Time Stepping

The universal oscillator equation with no forcing term is given by:

$$\ddot{x} + 2\zeta\dot{x} + x = 0, \quad t \in (0,T), \tag{1.2.1}$$

for $T > 0$. In (1.2.1), x denotes the position of the oscillator and $\dot{x}$ its velocity. The real parameter $\zeta > 0$ determines the damping behavior in the transient regime. For $\zeta > 1$ we have overdamping, for $\zeta = 1$ the so-called critical damping and for $\zeta < 1$ underdamping. In this exercise we consider the case $\zeta < 1$, $\zeta \neq 0$.

As initial conditions we impose

$$x(0) = x_0, \; \dot{x}(0) = v_0. \tag{1.2.2}$$

(1.2a) Equations (1.2.1) and (1.2.2) can be rewritten as a linear system of first order differential equations with appropriate initial conditions, i.e.:

$$\dot{\boldsymbol{y}} = \mathbf{A}\boldsymbol{y} \tag{1.2.3}$$

$$\boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{1.2.4}$$

Specify $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{y}, \boldsymbol{y}_0 \in \mathbb{R}^2$.

(1.2b) Compute the solution to (1.2.1) with initial conditions given by (1.2.2) with $x_0 = 1$ and $v_0 = 0$.

HINT: Starting from (1.2.3), use the matrix exponential technique.

HINT: Recall that $\zeta < 1$ and that, for a real number α, it holds that $e^{\alpha i} + e^{-\alpha i} = 2\cos\alpha$, where $i = \sqrt{-1}$ denotes the imaginary unit.

(1.2c) Recall the explicit Euler timestepping introduced in the lecture. Using the template file `harmonic_oscill.cpp` provided in the handout, implement the function `explicitEuler` to compute the solution $\boldsymbol{y} = \boldsymbol{y}(t)$ to (1.2.3) up to the time $T > 0$. The function should take as input the following parameters:

- The initial position x_0 and initial velocity v_0, stored in the 2×1 vector `y0`.
- The damping parameter ζ.
- The step size Δt, in the template called `dt`.
- The final time T, that we assume to be a multiple of Δt.

In output, the function returns the vectors `y1`, `y2` and `time`, where the i-th entry contains the particle position, the particle velocity, and the time, respectively, at the i-th iteration, $i = 1, \ldots, \frac{T}{\Delta t}$. The size of the output vectors has to be initialized inside the function according to the number of time steps.

(1.2d) Recall the implicit Euler timestepping introduced in the lecture. Using the template file `harmonic_oscill.cpp` provided in the handout, implement the function `implicitEuler` to compute the solution $\boldsymbol{y} = \boldsymbol{y}(t)$ to (1.2.3) up to the time $T > 0$. The input and output parameters are as in the function `explicitEuler` from subproblem (1.2c).

(1.2e) In the template file `harmonic_oscill.cpp`, complete the function `Energy` that, given in input a vector containing velocities at different time steps, returns the kinetic energy $E(t) = \frac{1}{2}v^2(t)$, where $v(t)$ denotes the velocity of the particle at time t.

(1.2f) We consider two time steps $\Delta_1 t = 0.1$ and $\Delta_2 t = 0.5$.

Using the `main` already implemented in the template file `harmonic_oscill.cpp`, plot the positions and the energies obtained with the explicit Euler time stepping and the implicit Euler for the two choices of time steps. For the position, plot the exact solution from subproblem (1.2b), too. What do you observe?

Problem 1.3 Multiple Choice: Basic concepts

Each question may have *multiple correct answers*.

(1.3a) Which of the following ODEs are *autonomous*?

1. $u'(t) = \cos(u(t))$
2. $u'(t) = u(t)^2 + t$
3. $u'(t) = \exp(t)u'(t)$
4. $u'(t) = \sin(u(t) + 2\pi t)$

(1.3b) Which of the following ODEs are *scalar*?

1. $u'(t) = 4u(t)$
2. $u'(t) = u(t)^2 + u(t)$
3. $\begin{pmatrix} u_1'(t) \\ u_2'(t) \end{pmatrix} = \begin{pmatrix} 3u_1(t) + u_2(t) \\ u_2(t) \end{pmatrix}$
4. $u'(t) = \cos(u(t))$

(1.3c) Which of the following ODEs are *linear*?

1. $u'(t) = 4u(t)$
2. $u'(t) = \exp u(t) + \sin(u(t))$
3. $\begin{pmatrix} u_1'(t) \\ u_2'(t) \end{pmatrix} = \begin{pmatrix} 3u_1(t) + u_2(t) \\ u_2(t) \end{pmatrix}$
4. $u'(t) = u(t)^2 + u(t)$

(1.3d) We are given the ODE

$$u'(t) = -u(t),$$

with initial value $u(0) = A > 0$. We solve the ODE using the *Forward-Euler* method with step size Δt. What will be the first value u_1?

1. Method will crash because of the minus sign
2. $(1 - \Delta t)A$
3. $A - \Delta t A + \Delta t^2 A$
4. $A \exp(-\Delta t)$

(1.3e) We are given the ODE

$$u'(t) = -u(t),$$

with initial value $u(0) = A > 0$. We solve the ODE using the *Backward-Euler* method with step size Δt. What will be the first value u_1?

1. Method will crash because of the minus sign
2. $(1 + \Delta t)A$
3. $A + \Delta t A + \Delta t^2 A$
4. $A/(1 + \Delta t)$

Published on September 29.

To be submitted on October 13.

Last modified on September 29, 2015