Out: January 28, 2019, 8:00
Deadline: February 10, 2019, 20:00

NTNU – Trondheim
Norwegian University of Science and Technology

Assignment 3
TTK4130 Modeling and Simulation

Problem 1 (The Modelica language, modelling of infectious disease outbreaks, simulation. 50 %)

NB: This is a computer exercise, and can therefore be solved in groups of 2 students. If you do so, please write down the name of your group partner in your answer.

The doomsday is upon us!

Zombies have begun to rise from the dead, and violently and indiscriminately kill and infest the living. In order to save humanity, you have to first model and simulate the zombie infestation using the little information available on these abominations.

There are 4 well-defined and non-overlapping populations:

- The healthy (H): Healthy people.
- The infected (I): People that have survived a zombie encounter, but have been infected.
- The zombies (Z).
- The dead (D): Recently deceased people and neutralized zombies.

The dynamics and interactions between these populations can be modeled like the classical predator-prey equations, i.e. the Lokta-Volterra equations, and they are given by the following laws:

- The birth-rate of the healthy is given by parameter $b > 0$. In addition, the birth-rate is damped by a quadratic term with parameter b_d.
- The healthy can either become dead by "natural" causes (non-zombie interaction) with death-rate $d > 0$, or can become infected due to interactions with zombies with parameter i.
- The healthy that become infected, remain infected for some time and then become zombies. This is model as a first order system with rate $a > 0$.
- Infected individuals can still die by "natural" causes before becoming a zombie with the same death-rate as healthy individuals. In such case, they become dead; otherwise, they become a zombie.
- Zombies rise from the dead with rate $r > 0$.
- Some zombies that interact with healthy individuals are neutralized with parameter $n > 0$. This zombies become dead, and may rise again.

(a) Model the dynamics of the populations H, I, Z and D.

Hint 1: The classical Lokta-Volterra equations are $\dot{x} = \alpha x - \beta xy$ and $\dot{y} = -\gamma y + \delta xy$, where x is the number of prey (e.g. tapirs) and y is the number predators (e.g. jaguars). In this model, $\alpha > 0$ is the birth-rate of the prey and $\gamma > 0$ is the death-rate of the predators. The interactions between these populations are modeled by the product xy, which is a measure of the number of encounters. Some encounters result in the death of prey, and enough eaten prey enable the predators to have offspring. How much the populations vary due to this interactions is described by the parameters $\beta > 0$ and $\delta > 0$.

Hint 2: The expressions for H', I', Z' and D' are given by polynomials in H, I, Z and D. Furthermore, $H' + I' + Z' + D'$ is equal to the net birth-rate $bH - b_d H^2$.

Solution:

$$\begin{aligned} H' &= (b-d)H - b_d H^2 - iHZ \\ I' &= -(a+d)I + iHZ \\ Z' &= aI + rD - nHZ \\ D' &= d(H+I) - rD + nHZ \end{aligned}$$

and $H, I, Z, D \geq 0$.

(b) Implement a Modelica model that represents the system $[H, I, Z, D]^T$ by completing the code shown below. Define a type for each population based on the *Real* type. These new types should have an in-built minimum value of 0 and the display of your choice. Use the parameter values $a = 1.4 \cdot 10^{-6}$, $b = 3.1 \cdot 10^{-8}$, $b_d = 5.6 \cdot 10^{-16}$, $d = 2.8 \cdot 10^{-8}$, $i = 2.6 \cdot 10^{-6}$, $n = 1.4 \cdot 10^{-6}$ and $r = 2.8 \cdot 10^{-7}$. Choose the start values for the variables that represent H, I, Z and D to be around 10 million.

```
model IncompleteZombieApocalypse "Incomplete zombie apocalypse model"
  // Define types, parameters and variables, as well as start values
  // ...
initial equation
  der(H) = 0;
  I=0;
  Z=0;
  D=0;
equation
  // The equations of the model
  // ...
  annotation(experiment(StartTime=0, StopTime=8640000, Tolerance=1e-6));
end IncompleteZombieApocalypse;
```

Simulate the model for 100 days. Add the Modelica model and a plot with the obtained results for all the populations to your answer. Comment on the results.
Are the initial values for the populations obtained in the simulations the same as the start values in the model code? Explain. What does the keyword *initial equation* mean?
What happens if one changes the start value for H to zero? Explain.
NB: The .mo file for the incomplete model has been uploaded together with this file.

Solution: One of infinite many correct solution is:

```
model ZombieApocalypse  "Zombie apocalypse model"
  type Healthy = Real(min=0, displayUnit="Healthy");
  type Infected = Real(min=0, displayUnit="Infected");
  type Zombie = Real(min=0, displayUnit="Zombie");
  type Dead = Real(min=0, displayUnit="Dead");
  parameter Real a = 1.4e-06  "Infected to zombie rate";
  parameter Real b = 3.1e-08  "Birth rate";
  parameter Real b_d = 5.6e-16  "Birth rate damping";
  parameter Real d = 2.8e-08  "Death rate";
  parameter Real i = 2.6e-06  "Healthy to infested parameter";
  parameter Real n = 1.4e-06  "Zombie to dead parameter";
  parameter Real r = 2.8e-07  "Dead to zombie rate";
  Healthy H(start = 5e+06) "Nr. of healthy people";
  Infected I(start = 1e+06) "Nr. of infected people";
  Zombie Z(start = 1e+06) "Nr. of zombies";
  Dead D(start = 1e+06) "Nr. of dead people";
initial equation
  der(H) = 0;
  I = 0;
  Z = 0;
  D = 0;
equation
  der(H) = (b - d)*H - b_d*H^2 - i*H*Z;
  der(I) = -(a + d)*I + i*H*Z;
  der(Z) = a*I + r*D - n*H*Z;
  der(D) = d*(H + I) - r*D + n*H*Z;
  annotation(experiment(StartTime=0, StopTime=8640000, Tolerance=1e-6));
end ZombieApocalypse;
```

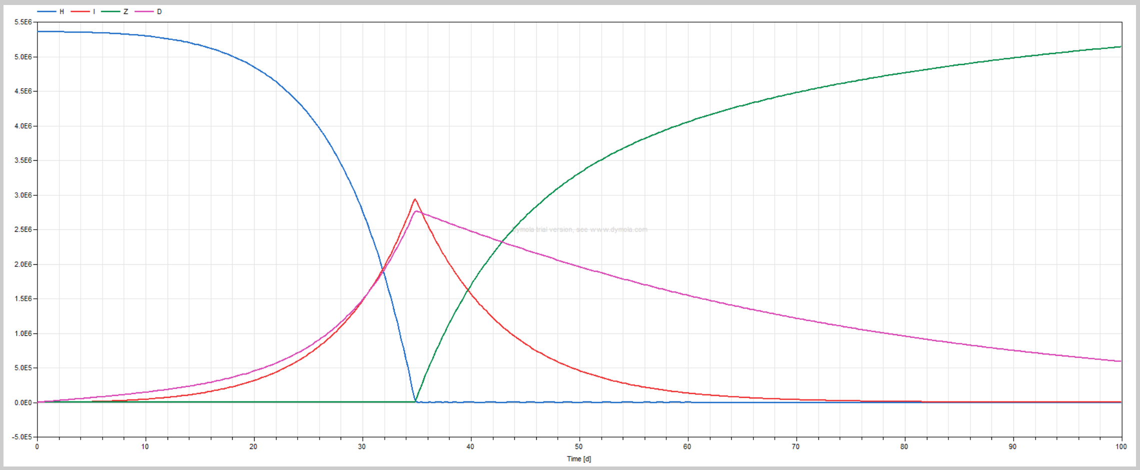


Figure 1: Simulation results: Doomsday at day 35.

The keyword *initial equation* defines the environment where the equations for the initial conditions are stated. These equations overwrite the start values of the variables involved. However, if the solution for these equations requires an iterative scheme, the (default) start values of the variables involved are used as the starting point for the scheme.
In this case, $H(0) = \frac{b-d}{b_d}$, $I(0) = 0$, $Z(0) = 0$ and $D(0) = 0$.
If one changes the start value for H to zero, then $H(0) = 0$, $I(0) = 0$, $Z(0) = 0$ and $D(0) = 0$.

The situation looks grim: All your simulations confirm that the outbreak of zombies will lead to $H \to 0$, i.e. the collapse of civilization.
In order to contain the outbreak, the authorities ask you to model the effects of a partial quarantine of zombies and infected individuals. Quarantined individuals are removed from their original populations and are placed on an special area, where they no longer can infect healthy individuals.
Consider the new population of quarantined individuals (Q), and add the following changes to the previous model:

- Infect individuals and zombies are quarantined with rates q_i and q_z, respectively.
- Some quarantined individuals will try to escape, and are immediately killed or neutralized, i.e. they moved to the dead population. This happens with rate d_q.
- The quarantined individuals that are moved to the dead population, may still rise as ”free roaming” zombies.

(c) Model the dynamics of the populations H, I, Z, D and Q.

Solution:

$$
\begin{aligned}
H' &= (b-d)H - b_d H^2 - iHZ \\
I' &= -(a+d+q_i)I + iHZ \\
Z' &= aI + rD - nHZ - q_z Z \\
D' &= d(H+I) - rD + nHZ + d_q Q \\
Q' &= q_i I + q_z Z - d_q Q
\end{aligned}
$$

and $H, I, Z, D, Q \geq 0$.

(d) Extend the Modelica model from part c. to represent the system $[H,\ I,\ Z,\ D,\ Q]^T$. Define a type for the population Q as done for the other populations. Do not change the parameter values for a, b, b_d, d, i, r and n, and do not change the start values for H, I, Z and D. Furthermore, use the parameter values $q_i = q_z = 2.7 \cdot 10^{-6}$ and $d_q = 2.8 \cdot 10^{-5}$, and add the initial equation $Q = 0$. Simulate the new model for 100 days. Add the Modelica model and a plot with the obtained results for all the populations to your answer. Comment on the results.

Solution: One of infinite many correct solution is:

```
model ZombieApocalypseQuarantine  "Zombie apocalypse model with quarantine"
  type Healthy = Real(min=0, displayUnit="Healthy");
  type Infected = Real(min=0, displayUnit="Infected");
  type Zombie = Real(min=0, displayUnit="Zombie");
  type Dead = Real(min=0, displayUnit="Dead");
  type Quarantined = Real(min=0, displayUnit="Quarantined");
  parameter Real a = 1.4e−06  "Infected to zombie rate";
  parameter Real b = 3.1e−08  "Birth rate";
  parameter Real b_d = 5.6e−16  "Birth rate damping";
  parameter Real d = 2.8e−08  "Death rate";
  parameter Real i = 2.6e−06  "Healthy to infested parameter";
  parameter Real n = 1.4e−06  "Zombie to dead parameter";
  parameter Real r = 2.8e−07  "Dead to zombie rate";
  parameter Real q = 2.7e−06  "Quarantine placement rate";
  parameter Real d_q = 2.8e−05  "Quarantine death rate";
  Healthy H(start = 5e+06) "Nr. of healthy people";
  Infected I(start = 1e+06) "Nr. of dead people";
  Zombie Z(start = 1e+06) "Nr. of zombies";
  Dead D(start = 1e+06) "Nr. of infected people";
  Quarantined Q(start = 1e+06) "Nr. of quarantined people";
initial equation
  der(H) = 0;
  I = 0;
  Z = 0;
  D = 0;
  Q = 0;
equation
  der(H) = (b − d)∗H − b_d∗H^2 − i∗H∗Z;
  der(I) = −(a + d + q)∗I + i∗H∗Z;
  der(Z) = a∗I + r∗D − n∗H∗Z − q∗Z;
  der(D) = d∗(H + I) + d_q∗Q − r∗D + n∗H∗Z;
  der(Q) = q∗(I + Z) − d_q∗Q;
  annotation(experiment(StartTime=0, StopTime=8640000, Tolerance=1e−6));
end ZombieApocalypseQuarantine;
```

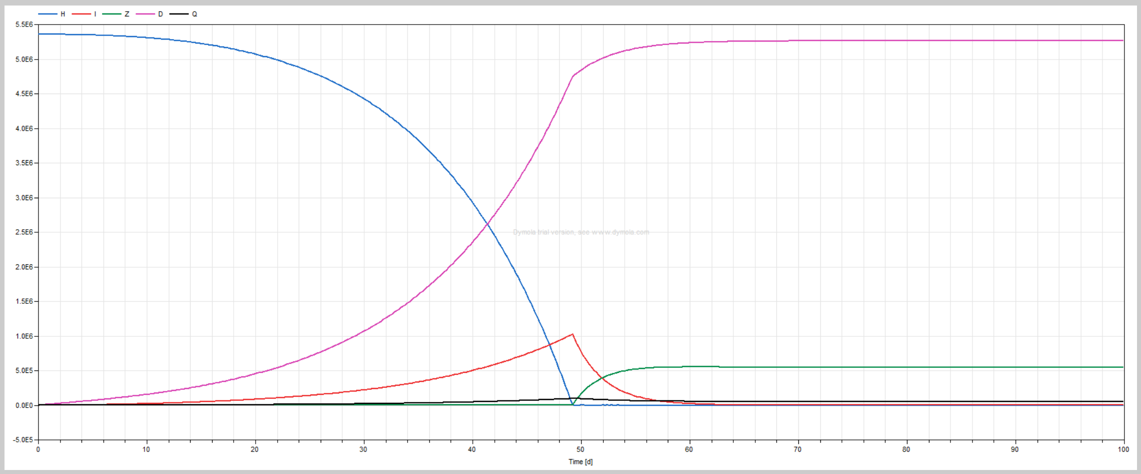


Figure 2: Simulation results: Doomsday at day 49.

The quarantined approach only seems to delay the inevitable. Luckily, news of a cure for ”zombie-ism” arrive. This treatment converts zombies and infected individuals to healthy individuals: If the individual was infected, it returns back to its healthy human form; and if it was resurrected from the dead, it returns to its healthy human form before death. Note however that this treatment does not provide immunity, i.e. individuals that are given the treatment may be converted to zombies in the future.

In order to model the effects of the cure, add the following changes to the previous model:

- Quarantine is no longer needed, i.e. the population Q is no longer part of the model.
- Zombies and infected individuals move to the healthy population with rate $c > 0$.

(e) Model the new dynamics of the populations H, I, Z and D.

Solution:

$$\begin{aligned} H' &= (b-d)H - b_d H^2 - iHZ + c(I+Z) \\ I' &= -(a+c+d)I + iHZ \\ Z' &= aI + rD - nHZ - cZ \\ D' &= d(H+I) - rD + nHZ \end{aligned}$$

and $H, I, Z, D \geq 0$.

(f) Extend the Modelica model from part c. to represent the new system $[H,\ I,\ Z,\ D]^T$. Do not change the parameter values for a, b, b_d, d, i, n and r, and do not change the start values for H, I, Z and D. Furthermore, use the parameter value $c = 2.7 \cdot 10^{-3}$.

Simulate the new model for 100 days. Add the Modelica model and a plot with the obtained results for all the populations to your answer. What is the stationary value of H? Comment on the results.

Hint: Not sure about the stationary value of H? Simulate the model for some years.

Solution: One of infinite many correct solution is:

```
model ZombieApocalypseCure  "Zombie apocalypse model with cure"
  type Healthy = Real(min=0, displayUnit="Healthy");
  type Infected = Real(min=0, displayUnit="Infected");
  type Zombie = Real(min=0, displayUnit="Zombie");
  type Dead = Real(min=0, displayUnit="Dead");
  parameter Real a = 1.4e-06  "Infected to zombie rate";
  parameter Real b = 3.1e-08  "Birth rate";
  parameter Real b_d = 5.6e-16  "Birth rate damping";
  parameter Real d = 2.8e-08  "Death rate";
  parameter Real i = 2.6e-06  "Healthy to infested parameter";
  parameter Real n = 1.4e-06  "Zombie to dead parameter";
  parameter Real r = 2.8e-07  "Dead to zombie rate";
  parameter Real c = 2.7e-03  "Cure rate";
  Healthy H(start = 5e+06) "Nr. of healthy people";
  Infected I(start = 1e+06) "Nr. of infected people";
  Zombie Z(start = 1e+06) "Nr. of zombies";
  Dead D(start = 1e+06) "Nr. of dead people";
initial equation
  der(H) = 0;
  I = 0;
  Z = 0;
  D = 0;
equation
  der(H) = (b - d)*H - b_d*H^2 - i*H*Z + c*(I+Z);
  der(I) = -(a + c + d)*I + i*H*Z;
  der(Z) = a*I + r*D - n*H*Z - c*Z;
  der(D) = d*(H + I) - r*D + n*H*Z;
  annotation(experiment(StartTime=0, StopTime=8640000, Tolerance=1e-6));
end ZombieApocalypseCure;
```

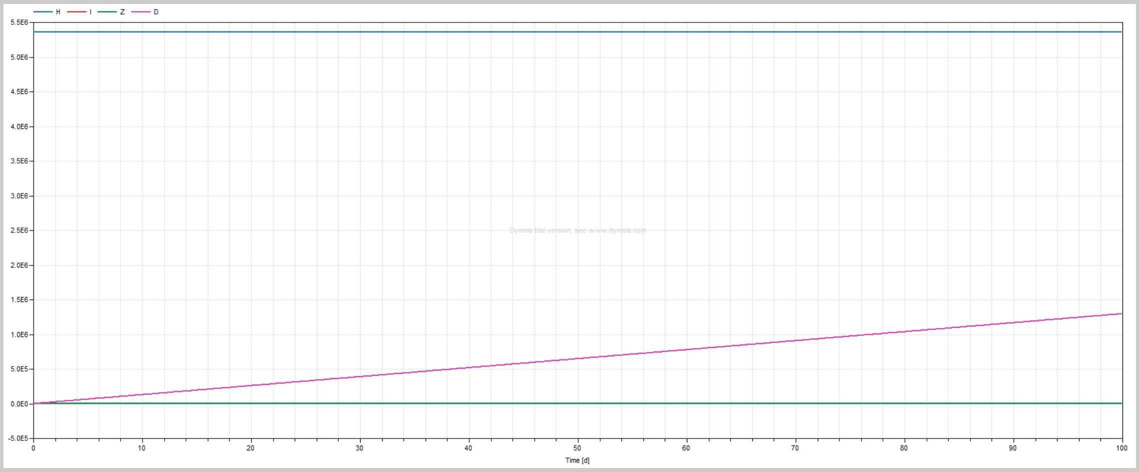


Figure 3: Simulation results: No doomsday.

The stationary value of *H* is around 2 million individuals.

Your are a champion of humanity! But before you can relax and think back with pride of the many lives your work saved, there is still one task to do: to clean up your code.

(g) Explain how you would use the language features of Modelica to reduce the amount of repeated code in the 3 models developed in parts (b), (d) and (f).

Feel free to rewrite the model codes so that the number of repeated code lines is minimized. If you do so, you only need to add the newest versions of the model codes to your answer. However, no code implementation is needed to solve this task.

Solution: One solution could involve the definition of a partial model with help variables that correspond to the interactions between populations that may differ from model to model. When the partial model is extended to a particular model, these help variables are set accordingly.
For example:

```
partial model ZombieApocalypsePartial "Zombie apocalypse partial model"
  type Healthy = Real(min=0, displayUnit="Healthy");
  type Infected = Real(min=0, displayUnit="Infected");
  type Zombie = Real(min=0, displayUnit="Zombie");
  type Dead = Real(min=0, displayUnit="Dead");
  type Quarantined = Real(min=0, displayUnit="Quarantined");
  parameter Real a = 1.4e-06  "Infected to zombie rate";
  parameter Real b = 3.1e-08  "Birth rate";
  parameter Real b_d = 5.6e-16  "Birth rate damping";
  parameter Real d = 2.8e-08  "Death rate";
  parameter Real i = 2.6e-06  "Healthy to infested parameter";
  parameter Real n = 1.4e-06  "Zombie to dead parameter";
  parameter Real r = 2.8e-07  "Dead to zombie rate";
  Healthy H(start = 5e+06) "Nr. of healthy people";
  Infected I(start = 1e+06) "Nr. of infected people";
  Zombie Z(start = 1e+06) "Nr. of zombies";
  Dead D(start = 1e+06) "Nr. of dead people";
  Quarantined Q_I(start = 0) "Quarantined infected";
  Quarantined Q_Z(start = 0) "Quarantined zombies";
  Dead D_Q(start = 0) "Dead quarantined";
  Healthy H_I(start = 0) "Cured infected";
  Healthy H_Z(start = 0) "Cured zombies";
initial equation
  der(H) = 0;
  I = 0;
  Z = 0;
  D = 0;
equation
  der(H) = (b - d)*H - b_d*H^2 - i*H*Z + H_I + H_Z;
  der(I) = -(a + d)*I + i*H*Z - Q_I - H_I;
  der(Z) = a*I + r*D - n*H*Z - Q_Z - H_Z;
  der(D) = d*(H + I) - r*D + n*H*Z + D_Q;
  annotation(experiment(StartTime=0, StopTime=8640000, Tolerance=1e-6));
end ZombieApocalypsePartial;
```

```
model ZombieApocalypse2  "Zombie apocalypse model v2"
  extends ZombieApocalypsePartial;
equation
  Q_I = 0;
  Q_Z = 0;
  D_Q = 0;
  H_I = 0;
  H_Z = 0;
end ZombieApocalypse2;
```

```
model ZombieApocalypseQuarantine2 "Zombie apocalypse model with quarantine v2"
```

```
  extends ZombieApocalypsePartial;
  parameter Real q = 2.7e-06  "Quarantine placement rate";
  parameter Real d_q = 2.8e-05  "Quarantine death rate";
  Quarantined Q(start = 1e+06) "Nr. of quarantined people";
initial equation
  Q = 0;
equation
  Q_I = q*I;
  Q_Z = q*Z;
  D_Q = d_q*Q;
  H_I = 0;
  H_Z = 0;
  der(Q) = Q_I + Q_Z - d_q*Q;
end ZombieApocalypseQuarantine2;
```

```
model ZombieApocalypseCure2 "Zombie apocalypse model with cure v2"
  extends ZombieApocalypsePartial;
  parameter Real c = 2.7e-03  "Cure rate";
equation
  Q_I = 0;
  Q_Z = 0;
  D_Q = 0;
  H_I = c*I;
  H_Z = c*Z;
end ZombieApocalypseCure2;
```

A zombie outbreak is of course an unrealistic scenario if taken literally. However, the structure of the models developed here can be used for real-life applications.

(h) Give some examples of such applications.

Solution: Diseases with dormant infection, allegiance to political parties, among others.

Problem 2 (Euler's method, linearization, stability. 25 %)

In this problem we will study the system

$$\ddot{x} + c\dot{x} + g\left(1 - \left(\frac{x_d}{x}\right)^{\kappa}\right) = 0, \tag{1}$$

where $\kappa = 2.40$ and $g = 9.81\,\mathrm{m/s^2}$.

(a) Write the system in state-space form.

Solution: Defining $y_1 = x$ and $y_2 = \dot{x}$, we can write

$$\begin{aligned}\dot{y}_1 &= y_2\,,\\ \dot{y}_2 &= -cy_2 - g\left(1 - \left(\frac{x_d}{y_1}\right)^{\kappa}\right).\end{aligned}$$

(b) Set up the Euler's method for the system.

Solution: Euler's method for this system becomes

$$y_{1,n+1} = y_{1,n} + hy_{2,n}\,,$$
$$y_{2,n+1} = (1 - ch)y_{2,n} - gh\left(1 - \left(\frac{x_d}{y_{1,n}}\right)^{\kappa}\right).$$

(c) Set up the modified Euler's method for the system.

Solution: We find $\mathbf{k}_1$ and $\mathbf{k}_2$ for the modified Euler's method

$$\mathbf{k}_1 = \begin{bmatrix} k_{1,1} \\ k_{1,2} \end{bmatrix} = \begin{bmatrix} y_{2,n} \\ -cy_{2,n} - g\left(1 - \left(\frac{x_d}{y_{1,n}}\right)^{\kappa}\right) \end{bmatrix},$$
$$\mathbf{k}_2 = \begin{bmatrix} k_{2,1} \\ k_{2,2} \end{bmatrix} = \begin{bmatrix} y_{2,n} + \frac{h}{2}k_{1,2} \\ -c\left(y_{2,n} + \frac{h}{2}k_{1,2}\right) - g\left(1 - \left(\frac{x_d}{y_{1,n}+\frac{h}{2}k_{1,1}}\right)^{\kappa}\right) \end{bmatrix}.$$

Hence, the modified Euler's method is

$$y_{1,n+1} = y_{1,n} + hk_{2,1},$$
$$y_{2,n+1} = y_{2,n} + hk_{2,2}.$$

(d) Linearize the system about its only equilibrium point.

Solution: The only equilibrium point is $x = x_d$, i.e. $[y_1,\, y_2]^T = [x_d,\, 0]^T$.
The linearized system about this stationary point is

$$\Delta\dot{y}_1 = \left.\frac{\partial f_1}{\partial y_1}\right|_{y_1=x_d, y_2=0} \Delta y_1 + \left.\frac{\partial f_1}{\partial y_2}\right|_{y_1=x_d, y_2=0} \Delta y_2$$
$$\Delta\dot{y}_2 = \left.\frac{\partial f_2}{\partial y_1}\right|_{y_1=x_d, y_2=0} \Delta y_1 + \left.\frac{\partial f_2}{\partial y_2}\right|_{y_1=x_d, y_2=0} \Delta y_2\,,$$

i.e. the system matrix at this point is

$$A = \left.\begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{bmatrix}\right|_{y_1=x_d, y_2=0} = \left.\begin{bmatrix} 0 & 1 \\ -\kappa g \frac{x_d^{\kappa}}{y_1^{\kappa+1}} & -c \end{bmatrix}\right|_{y_1=x_d, y_2=0} = \begin{bmatrix} 0 & 1 \\ -\frac{\kappa g}{x_d} & -c \end{bmatrix}.$$

The eigenvalues of the linearized system are given by

$$\det\left(A - \lambda I\right) = \det\begin{bmatrix} -\lambda & 1 \\ -\frac{\kappa g}{x_d} & -c - \lambda \end{bmatrix} = \lambda^2 + \lambda c + \frac{\kappa g}{x_d} = 0.$$

Solving this, we find

$$\lambda_{1,2} = -\frac{c}{2} \pm \sqrt{\frac{c^2}{4} - \frac{\kappa g}{x_d}}$$

(e) Assume $x_d = 1.32$. For which step-lengths h will the Euler's method be stable, if

i) $c = 0$?

ii) $c = 8.927$?

Solution:

i) For $c = 0$ the eigenvalues are

$$\lambda_{1,2} = \pm 4.2233i,$$

In particular, they are purely imaginary (physically: when $c = 0$, there is no damping in the system).

Euler's method is stable when

$$|1 + h\lambda| \leq 1$$

which is impossible to fulfill with purely imaginary eigenvalues for any h. In other words, Euler's method is unstable for all systems with purely imaginary eigenvalues.

Direct computation gives of course the same result:

$$|1 + h\,(\pm 4.2233i)| = \sqrt{1 + 17.8363h^2} > 1, \quad \text{for all } h > 0.$$

ii) For $c = 8.927$ the eigenvalues are

$$\lambda_1 = -3.0190 \quad \text{and} \quad \lambda_2 = -5.9080.$$

Euler's method is stable if and only if

$$|1 + h\lambda| \leq 1\,,$$

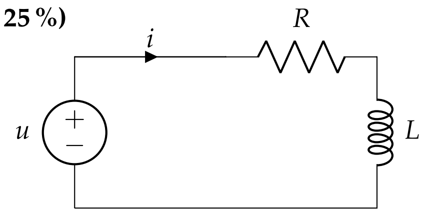
for all eigenvalues in the system. In our case, this is equivalent to

$$h \leq -\frac{2}{\lambda_2} = 0.3385.$$

Note that h is a positive number by definition.

Problem 3 (Modeling, control and simulation of RL-circuit. 25 %)

The figure on the right shows an electrical RL circuit, where $L > 0$ is the inductance, $R > 0$ is the resistance, i is the current through the circuit and i_0 is the initial current. The voltage u of the voltage source can be adjusted, and will be used as a control input. The circuit is modeled by



$$L\frac{d\,i}{d\,t} + Ri = u, \quad t \geq 0 \tag{2a}$$

$$i(0) = i_0 \tag{2b}$$

(a) Let $u = 0$. Show that the system is asymptotically stable about the equilibrium $i^* = 0$, using energy-based methods. What happens with the current $i(t)$ when $t \to \infty$? Make a sketch.
Hint: Remember that the energy stored in an inductor is given by $E = \frac{1}{2}Li_L^2$, where i_L is the current through the inductor. Find a differential equation for this energy function.

Solution: We define the positive definite energy function

$$E = \frac{1}{2}Li^2 \succ 0.$$

We differentiate this function along the solution of (2a):

$$\dot{E} = \frac{\partial E}{\partial i}\frac{d\,i}{d\,t} = Li\left(-\frac{R}{L}i\right) = -Ri^2 \prec 0.$$

A positive definite energy function with negative definite derivative, implies that the system is asymptotically (exponentially) stable.
Moreover, in this particular case, it is easy to verify that

$$\dot{E}(t) = -\frac{2R}{L}E(t)$$

which implies that $E(t) = E(0)\mathrm{e}^{\frac{-2R}{L}t} \to 0$ as $t \to \infty$, which again implies that $i(t) \to 0$ by the definition of $E(t)$. Furthermore, it is also very easy to check the asymptotic stability of this linear system by confirming that the eigenvalue $-R/L$ is negative.

(b) Assume that we desire to keep a constant current $i_{\text{ref}} > 0$. Design a controller u such that $i(t) \to i_{\text{ref}}$ when $t \to \infty$.
Hint: Define $e = i - i_{ref}$, and use the energy function $E = \frac{1}{2}Le^2$. What happens if the controller u is such that $\dot{E} < 0$ for $e \neq 0$?

Solution: We find that

$$L\dot{e} + Re = u - Ri_{\text{ref}}.$$

Differentiating the energy function, we obtain

$$\dot{E} = Le\dot{e} = -Re^2 + e\left(u - Ri_{\text{ref}}\right).$$

If we let

$$u = Ri_{\text{ref}} - K_p e$$

for $K_p > -R$, then

$$\dot{E} = -(R + K_p)e^2 \prec 0\,,$$

which shows that $e(t) \to 0$, which again means that $i(t) \to i_{\text{ref}}$.

(c) Let $i_0 = 0\,\text{A}$, $i_{\text{ref}} = 1\,\text{mA}$, $L = 3\,\text{mH}$ and $R = 2.5\,\text{k}\Omega$.
Implement a Matlab code that simulates (2) from $t = 0$ to $t = 10\,\mu\text{s}$ by using the (classical) fourth order ERK-method, with constant step-length $h = 10\,\text{ns}$.
Attach a code printout to your answer.

Solution:

```
% Simulate RL circuit
t = [0 1e-5]; h = 1e-8;
i0 = 0; L = 3e-3; R = 2.5e3; Kp = 0; iref = 1e-3;

% Number of iterations
N = round(t(2)/h,0);

% Define Butcher-array (book p. 528)
A = diag([0.5 0.5 1]);
b = [1/6 2/6 2/6 1/6]';
c = [0 0.5 0.5 1]';

% Order of ERK method
sigma = size(A,1) + 1;

% RL-circuit model
RL = @(i) ( -R/L * i + 1/L * ( R*iref - Kp*(i - iref)) );

% Initialize storage
y = zeros(N+1,1); y(1) = i0;

% We use numerical scheme from book p. 526
k = zeros(sigma,1);
for n = 1:N
    for j = 1:sigma
        A_diag = diag(A);
        k(j) = RL(y(n) + h * sum(A_diag(1:j - 1) .* k(1:j - 1)));
    end
    y(n + 1) = y(n) + h * sum(b .* k);
end

plot(1e6*(t(1):h:t(2)),1e3*y); xlabel('Time [\mu s]'); ylabel('Current [m A]');
```

(d) Implement a Modelica/Dymola code that simulates (2). This could be done using components from the Modelica Standard Library, but instead write in the parameter definitions and equations. When you simulate, choose the solver Rkfix4 ('Simulation' $\to$ 'Setup'), which is the same solver you implemented in part (c). Furthermore, use the same parameter values as in part (c), and compare the obtained results.

Solution: One solution is:

```
model CircuitRL
  import SI = Modelica.SIunits;
  SI.Current i;
  SI.Voltage u;
  parameter SI.Resistance R = 2.5e+03;
  parameter SI.Inductance L = 3.0e−03;
  parameter SI.Current i_ref = 1.0e−03;
  parameter Real K=0;
equation
  u = R*i_ref − K*(i − i_ref);
  L*der(i) + R*i = u;
end CircuitRL;
```

The simulation results should be identical to the ones from the Matlab script.

Problem 4 (Euler's method, convergence, stability. Optional)

Consider the first order linear differential equation with constant coefficients

$$\dot{x} = ax + b\,, \quad x(0) = x_0\,, \tag{3}$$

where $a, b, x_0 \in \mathbb{R}$.

(a) Find the solution $x(t)$, $t \geq 0$.

Solution:

$$x(t) = \begin{cases} x_0 e^{at} + \frac{b}{a}(e^{at} - 1) & \text{, if } a \neq 0 \\ x_0 + bt & \text{, if } a = 0 \end{cases}$$

(b) Set up Euler's method for this linear system. This gives a recursive expression for the numerical solution x_n. In order to analyze the properties of the numerical solution, we will first find an explicit expression for x_n. Show using induction that the numerical solution is given by

$$x_n = x_0\,(1 + ah)^n + hb \sum_{j=0}^{n-1} (1 + ha)^j \tag{4}$$

for $n = 0, \cdots, N$, where h is the step-length and N is the number of intervals.
Finally, show that (4) can be rewritten as

$$x_n = \begin{cases} x_0 + nhb & , a = 0 \\ x_0\,(1 + ah)^n + \frac{b}{a}\left((1 + ah)^n - 1\right) & , a \neq 0 \end{cases} \tag{5}$$

NB: Use the formula for a geometric sum in order to simplify the expression for x_n in (4).

Solution:

$$x_{n+1} = x_n + h(ax_n + b) = (1 + ha)x_n + hb.$$

The prove by induction is standard. Finally, use the formula for a geometric sum.

(c) Under which conditions on a does x_n converge as $n \to \infty$ for all x_0 and b.
Comment on the stability of the method.

Solution: The condition is $|1 + ah| < 1$. However, the integration method is stable if and only if $|1 + ah| \leq 1$.

(d) We will now prove the pointwise convergence of the Euler's method for this particular system. Therefore, assume that $h = h(n)$ is chosen such that the discretization point $t_n = nh$ is constant for all $n \in \mathbb{N}$. Show that

$$\lim_{n\to\infty} x_n = x(t_n) \tag{6}$$

Hint: Use the result $\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x$ *for all* $x \in \mathbb{R}$.

Solution: If $a \neq 0$, then

$$\begin{aligned} x_n &= \left(1 + \frac{at_n}{n}\right)^n + \frac{b}{a}\left(\left(1 + \frac{at_n}{n}\right)^n - 1\right) \\ &\to x_0 e^{at_n} + \frac{b}{a}(e^{at_n} - 1) = x(t_n). \end{aligned}$$

If $a = 0$, then

$$x_n = x_0 + nhb = x(t_n).$$

(e) Find an expression for the global error $E_n = x_n - x(t_n)$ based on the results from part (b).
Show that $E_n = o(h)$, which is stronger than the general result $E_n = O(h)$ for the Euler's method.
NB: $f(x) = o(x)$ *if and only if* $\lim\limits_{x\to 0} = \frac{f(x)}{x} = 0$.

Solution: If $a = 0$, then

$$E_n = 0 = o(h).$$

If $a \neq 0$, then

$$\begin{aligned} E_n &= \left(x_0 + \frac{b}{a}\right)\left(e^{at_n} - \left(1 + \frac{at_n}{n}\right)^n\right) \\ &= \left(x_0 + \frac{b}{a}\right)\left(e^{anh} - (1 + ah)^n\right). \end{aligned}$$

Since

$$\lim_{h\to 0^+} \frac{|E_n|}{h} = \lim_{h\to 0^+} \left|x_0 + \frac{b}{a}\right| \frac{|e^{anh} - (1 + ah)^n|}{h} = 0\,,$$

it follows that $E_n = o(h)$.

(f) In the context of numerical integration methods, what is the difference between stability and accuracy?

Solution: An integration method is stable if the values it generates are bounded. On the other hand, an integration method is accurate if the values it generates are a good approximation of the true values. One concept does not imply the other.