Out: January 20, 2014
In: January 29–February 7, 2014

NTNU – Trondheim
Norwegian University of Science and Technology

Exercise 4
TTK4130 Modeling and Simulation

Problem 1 (Taylor expansions and order conditions)
The Butcher array for an explicit Runge-Kutta method with two stages is

$$\begin{array}{c|cc} 0 & & \\ c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array}$$

Assume (for simplicity) that the problem is scalar, that is, the dimension of $\mathbf{y}$ is 1. The stage computations are

$$\begin{aligned} k_1 &= f(y_n, t_n), \\ k_2 &= f(y_n + ha_{21}k_1, t_n + hc_2). \end{aligned}$$

Recall that the Taylor expansion of a function of two variables can be written

$$f(y+\Delta, t+\delta) = f(y,t) + \Delta\frac{\partial f(y,t)}{\partial y} + \delta\frac{\partial f(y,t)}{\partial t} + O(\Delta^2) + O(\delta\Delta) + O(\delta^2).$$

(a) Derive a first order taylor expansion of k_2, assuming that $a_{21} = c_2$, and using that

$$\frac{\mathrm{d}f(y_n,t_n)}{\mathrm{d}t} = \frac{\partial f(y_n,t_n)}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial f(y_n,t_n)}{\partial t} = \frac{\partial f(y_n,t_n)}{\partial y}f(y_n,t_n) + \frac{\partial f(y_n,t_n)}{\partial t}.$$

Solution: By letting

$$\begin{aligned} \Delta &= ha_{21}k_1 = ha_{21}f(y_n,t_n), \\ \delta &= hc_2, \end{aligned}$$

we can write

$$\begin{aligned} k_2 &= f(y_n + ha_{21}k_1, t_n + hc_2) \\ &= f(y_n,t_n) + ha_{21}f(y_n,t_n)\frac{\partial f(y_n,t_n)}{\partial y} + hc_2\frac{\partial f(y_n,t_n)}{\partial t} + O(h^2) \\ &= f(y_n,t_n) + ha_{21}\left(\frac{\partial f(y_n,t_n)}{\partial y}f(y_n,t_n) + \frac{\partial f(y_n,t_n)}{\partial t}\right) + O(h^2) \\ &= f(y_n,t_n) + ha_{21}\frac{\mathrm{d}f(y_n,t_n)}{\mathrm{d}t} + O(h^2). \end{aligned}$$

(b) Derive the conditions on b_1, b_2 and $c_2 = a_{21}$ for the method to be of order 2.

Solution: Inserting the above result into $y_{n+1} = y_n + h(b_1k_1 + b_2k_2)$, we get

$$\begin{aligned} y_{n+1} &= y_n + hb_1k_1 + hb_2k_2 \\ &= y_n + hb_1f(y_n,t_n) + hb_2\left(f(y_n,t_n) + ha_{21}\frac{\mathrm{d}f(y_n,t_n)}{\mathrm{d}t} + O(h^2)\right) \\ &= y_n + h(b_1+b_2)f(y_n,t_n) + h^2b_2a_{21}\frac{\mathrm{d}f(y_n,t_n)}{\mathrm{d}t} + O(h^3). \end{aligned}$$

Comparing this to the Taylor expansion of the exact solution starting at y_n (the local solution),

$$\begin{aligned} y(t_n+h) &= y(t_n) + h\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{h^2}{2!}\frac{\mathrm{d}^2y}{\mathrm{d}t^2} + O(h^3) \\ &= y(t_n) + hf(y_n,t_n) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n,t_n)}{\mathrm{d}t} + O(h^3), \end{aligned}$$

we see that the local error is $O(h^3)$ (the order of the method is 2) if

$$\begin{aligned} b_1 + b_2 &= 1, \\ b_2 a_{21} &= \frac{1}{2}. \end{aligned}$$

Remark 1: From the above, we can conclude that if the order conditions are satisfied, the order of the method is at least two. To see that it cannot be higher, one must calculate the second order Taylor expansion of k_2, and see it is impossible to match the third-order terms. This is a nice exercise for those who like to differentiate!

Remark 2: This procedure (matching the expansions of the stage computations with the expansion of the exact solution) is (as we have seen) fairly easy using first order expansions, doable (but involved) for second order expansions but practically impossible (by hand) for higher order expansions. Special computer programs for this have been developed to find order conditions for higher order Runge-Kutta methods.

Problem 2 (Implementing solvers for the pneumatic spring)

This problem continues with the pneumatic spring, described by

$$\ddot{x} + g\left[1 - \left(\frac{1}{x}\right)^{\kappa}\right] = 0,$$

where $\kappa = 1.40$ and $g = 9.81$. Defining $y_1 = x$ and $y_2 = \dot{x}$, this is on state-space form

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

Recall (from the book or previous exercises) that since there is no damping, the physical solution is that the position oscillates around the equilibrium position $y_1 = 1$.

(a) Implement the explicit Euler method (or an explicit two-stage Runge-Kutta method) in Matlab and simulate from $t = t_0 = 0$ to $t = 10$ s, with step length $h = 0.01$ s. Use initial condition $y_0 = (2,0)^\top$. Plot the position, and comment.

Solution: The position is shown in Figure 1. Since there is no supply or dissipation of energy in the system, we would expect standing oscillations. However, this explicit Euler solution is unstable (the energy is, unphysically, increasing). This is as expected for this method: In a previous problem we saw that the eigenvalues of the linearization of this model is on the imaginary axis, and Euler's method cannot be stable for purely imaginary eigenvalues.

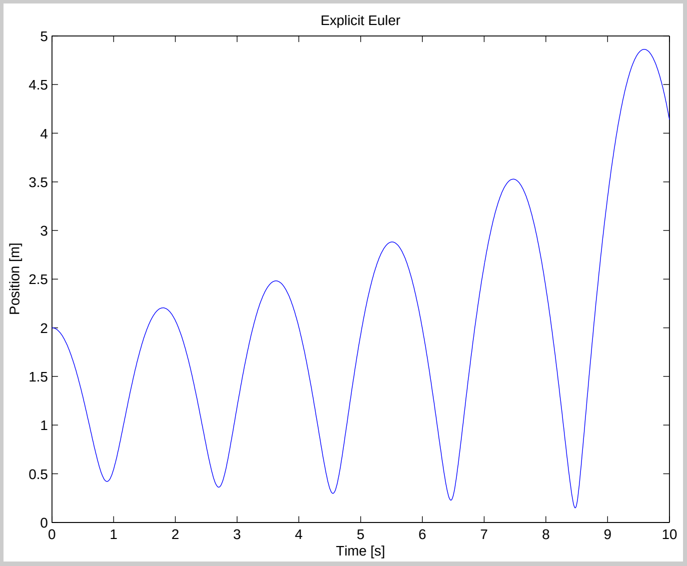

Figure 1: Simulation using explicit Euler

The code in the below listing contains code for solving all part of this problem.

```
% Script to simulate pneumatic spring using explicit Euler, implicit Euler
% and implicit midpoint rule (Gauss order 2).

%% Parameters
g = 9.81; K = 1.4;
y10 = 2; y20 = 0; y0 = [y10;y20]; % Initial values
h = 0.01; % Step size
t0 = 0; tstop = 10; % Time start and stop
time = t0:h:tstop; % Generate time vector
nstep = ((tstop-t0)/h)+1;
opt = optimset('Display','off','TolFun',1e-8); % Options for fsolve

%% Create storage
y_EE = zeros(size(y0,1),size(time,2)); % Explicit Euler
y_IE = zeros(size(y0,1),size(time,2)); % Implicit Euler
y_IM = zeros(size(y0,1),size(time,2)); % Implicit Midpoint rule

%% Function y' = f(y,t)
f = @(y,t) [ y(2); -g*(1-(1/y(1))^K) ];

%% EXPLICIT EULER
y_EE(:,1) = y0; % Initial value
for i = 1:nstep-1,
    y_EE(:,i+1) = y_EE(:,i) + h*feval(f,y_EE(:,i),time(i));
end
% Plots the results for Explicit Euler
figure(1); plot(time,y_EE(1,:));
xlabel('Time [s]'); ylabel('Position [m]'); title('Explicit Euler')

%% IMPLICIT EULER
y_IE(:,1) = y0; % Initial value is set.
```

```
for i = 1:nstep-1,
    r = @(y) (y_IE(:,i) + h*feval(f,y,time(i+1)) - y); % Root function
    [y_IE(:,i+1),fval,exitflag,output] = fsolve(r,y_IE(:,i),opt);
end
% Plots the results for Implicit Euler
figure(2); plot(time,y_IE(1,:));
xlabel('Time [s]'); ylabel('Position [m]'); title('Implicit Euler')

%% IMPLICIT MIDPOINT RULE
y_IM(:,1) = y0; % Initial value is set.
for i = 1:nstep-1,
    r = @(y) (y_IM(:,i) + h*feval(f,(y_IM(:,i) + y)/2,time(i)+h/2) - y);
    [y_IM(:,i+1),fval,exitflag,output] = fsolve(r, y_IM(:,i), opt);
end
% Plots the results for Implicit Midpoint Rule
figure(3); plot(time,y_IM(1,:));
xlabel('Time [s]'); ylabel('Position [m]'); title('Implicit Midpoint Rule')

%% Plot energies
p0 = 200000; A = 0.01; g =10; m = 200;
E_EE = 1/(K-1)*p0*A*y_EE(1,:).^(-(K-1)) + m*g*y_EE(1,:) + 1/2*m*y_EE(2,:).^2;
E_IE = 1/(K-1)*p0*A*y_IE(1,:).^(-(K-1)) + m*g*y_IE(1,:) + 1/2*m*y_IE(2,:).^2;
E_IM = 1/(K-1)*p0*A*y_IM(1,:).^(-(K-1)) + m*g*y_IM(1,:) + 1/2*m*y_IM(2,:).^2;
figure(4); plot(time,E_EE); hold on;
plot(time,E_IE,'m--'); plot(time,E_IM,'c:'); hold off;
legend('Explicit Euler','Implicit Euler',...
    'Implicit Midpoint Rule (Gauss 2)','Location','NorthWest');
xlabel('Time [s]'); ylabel('Energy [J]');
```

(b) Implement the implicit Euler method for the same problem. Use fsolve from the optimization toolbox (or implement a Newton-type algorithm yourself) to solve the nonlinear equation. For example: Define the model as

```
    f = @(y,t) [ y(2); -g*(1-(1/y(1))^K) ];
```

Then, in each iteration of the for-loop, define the function to be solved ($r(y_{n+1}) = y_n + hf(y_{n+1}, t_{n+1}) - y_{n+1} = 0$), and call fsolve.

```
    r = @(ynext) (y(:,i) + h*feval(f, ynext, time(i+1)) - ynext);
    y(:,i+1) = fsolve(r, y(:,i), opt);
```

To get this to work, it is important to set small tolerances for the solution:

```
    opt = optimset('Display','off','TolFun',1e-8); % Options for fsolve
```

Remark: Using fsolve for this is not particularly efficient. If you want, you can provide the Jacobian of r to fsolve to speed up the solutions (the Jacobian of f was calculated in an earlier exercise). You may also experiment by using the procedure outlined in Chapter 14.8.1 in the book (not required).

Solution: See listing above. A simulation is shown in Figure 2. We see that the solution now is stable, but that energy is removed from the simulation. This is in accordance with the method being L-stable.

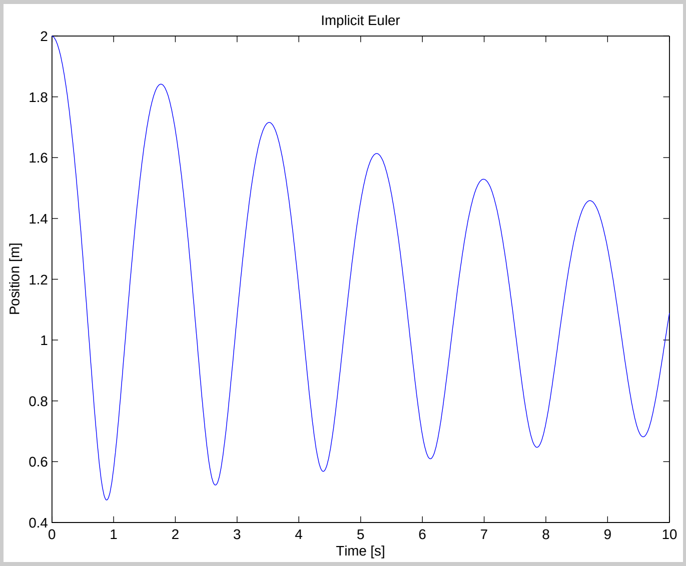

Figure 2: Simulation using implicit Euler

(c) Implement the implicit midpoint rule (Gauss order 2) for this problem, $y_{n+1} = y_n + hf((y_n + y_{n+1})/2, t_n + h/2)$.

Solution: See listing above, and simulation in Figure 3. We see that this method does not remove energy from the simulation, in accordance with the method being A-stable but not L-stable.

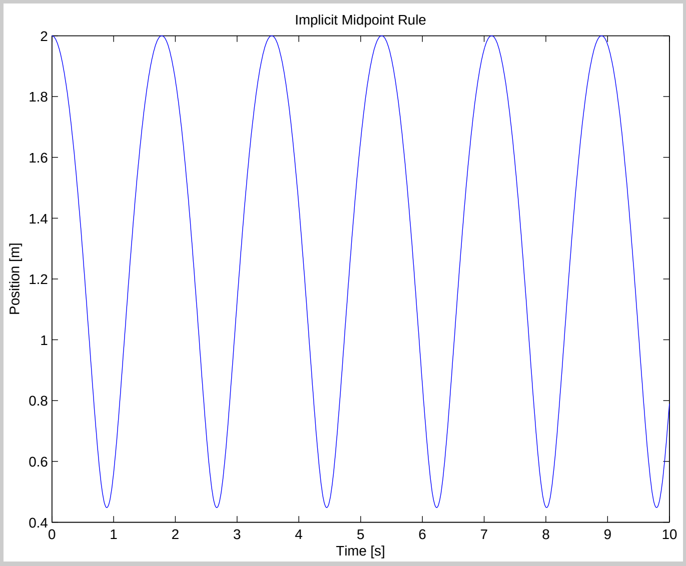

Figure 3: Simulation using implicit midpoint rule (Gauss order 2)

(d) The energy for the system is

$$E = \frac{1}{\kappa - 1} p_0 A x^{-(\kappa-1)} + mgx + \frac{1}{2} m v^2$$

Plot the energy for all three solutions above. Assume $A = 0.01$ m^2, $m = 200$ kg og $p_0 = 2 \cdot 10^5$ N/m^2.

Solution: See Figure 4. This plot agrees with the insights obtained above.

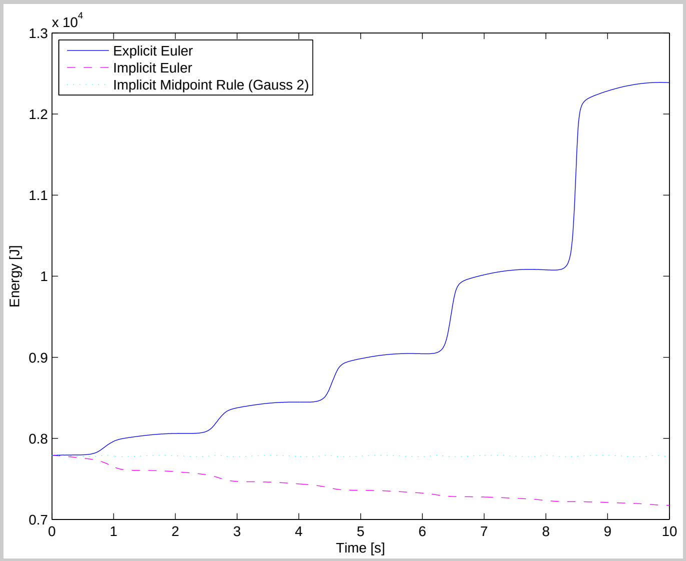

Figure 4: Energies

Problem 3 (Passivity of DC motor)

A voltage controlled DC motor can be described by the model

$$L_a \frac{\mathrm{d}i_a}{\mathrm{d}t} = -R_a i_a - K_E \omega_m + u_a \tag{1a}$$

$$J_m \frac{\mathrm{d}\omega_m}{\mathrm{d}t} = K_T i_a - T_L \tag{1b}$$

Let $K_E = K_T$.

(a) Let $T_L = 0$ and $u_a = 0$. Show that the system is stable via use of the energy function

$$E = \underbrace{\frac{1}{2} L_a i_a^2}_{\text{energy in coil}} + \underbrace{\frac{1}{2} J_m \omega_m^2}_{\text{kinetic energy}} . \tag{2}$$

Solution: Differentiate (2) along the the solutions of (1) to obtain

$$\begin{aligned}\dot{E} &= L_a \frac{di_a}{dt} i_a + J_m \dot{\omega}_m \omega_m \\ &= (-R_a i_a - K_E \omega_m + \underbrace{u_a}_{=0}) i_a + (\underbrace{K_T}_{=K_E} i_a - \underbrace{T_L}_{=0}) \omega_m \\ &= -R_a i_a^2 - K_E \omega_m i_a + K_E \omega_m i_a \\ &= -R_a i_a^2 \leq 0 \quad \forall (i_a, \omega_m) .\end{aligned}$$

This implies stability of (1).

(b) Let $\mathbf{u} = \begin{bmatrix} u_a & -T_L \end{bmatrix}^\top$ og $\mathbf{y} = \begin{bmatrix} i_a & \omega_m \end{bmatrix}^\top$. Show that the system is passive for this choice of inputs and outputs. Hint: Use (2) as storage function.

Solution: We differentiate the storage function E along the the solutions of (1) to obtain

$$\begin{aligned}\dot{E} &= -R_a i_a^2 + i_a u_a + (-T_L)\omega_m \\ &= -R_a i_a^2 + [u_a \quad -T_L][i_a \quad \omega_m]^\top \\ &= \underbrace{-R_a i_a^2}_{\leq 0} + \mathbf{u}^\top \mathbf{y}\end{aligned}$$

From (2.149) in the book, we can conclude that the DC motor with input $\mathbf{u}$ and output $\mathbf{y}$ is passive.

This is also seen by integrating the previous equation,

$$\begin{aligned}\int_0^T \dot{E}(t)\,\mathrm{d}t &\leq \int_0^T \mathbf{u}(t)^\top \mathbf{y}(t)\,\mathrm{d}t \\ E(T) - E(0) &\leq \int_0^T \mathbf{u}(t)^\top \mathbf{y}(t)\,\mathrm{d}t\end{aligned}$$

Since $E(T) > 0$,

$$E(T) - E(0) \leq -E(0) \leq \int_0^T \mathbf{u}(t)^\top \mathbf{y}(t)\,\mathrm{d}t$$

which implies passivity by definition 1, page 53.

(c) Suggest an input $\mathbf{u}$ which can control the system to a given setpoint.

Solution: Since the system is passive, we can use any two PID controllers that couples these inputs with these outputs, see 2.4.17 in book for "proof". This illustrates the power of knowledge of passivity.

The rest of the solution is not necessary, but included for completeness.

A full proof of stability for the closed-loop system with PID-controllers using energy functions is a bit involved, and not in the scope of this course (but note that (2.173) implies stability when the desired values are zero). One may of course also resort to Bode- or Nyquist plots to show stability, since the system is linear.

We provide here a simple proof using slightly modified P-controllers. Assume the desired setpoints are i_a^* and ω_m^*, and let

$$\begin{aligned}u_a &= -R_a i_a^* - K_E \omega_m^* - K_{P1}\left(i_a - i_a^*\right), \\ T_L &= K_T i_a^* - K_{P2}\left(\omega_m - \omega_m^*\right),\end{aligned}$$

where K_{p1}, K_{p2} are constants. Use the energy function

$$E = \frac{1}{2} L_a \left(i_a - i_a^*\right)^2 + \frac{1}{2} J_m \left(\omega_m - \omega_m^*\right)^2,$$

to find that

$$\dot{E} = -(R_a + K_{P1})\left(i_a - i_a^*\right)^2 - K_{P2}\left(\omega_m - \omega_m^*\right)^2,$$

that is, the system is stable if $K_{P1} \geq -R_a$ and $K_{P2} \geq 0$ (and exponentially stable if the inequalities are strict).