Modsim Assignment 9 Eirik Falck

Problem 1

a) See code below.

b) We can see the simulated solution following the true curve very well and has (almost) the exact same function value at the sample points.

c) $b = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T$

$$K_1 = f(x_k + \Delta t[a_{11}K_1 + a_{12}K_2])$$
$$= \lambda(x_k + \Delta t a_{11} K_1 + \Delta t a_{12} K_2)$$
$$= \lambda x_k + \lambda \Delta t a_{11} K_1 + \lambda \Delta t a_{12} K_2$$

$$(1-\lambda\Delta t a_{11})K_1 = \lambda x_k + \lambda \Delta t a_{12} K_2$$

$$K_1 = \frac{\lambda}{1-\lambda\Delta t a_{11}}(x_k + \Delta t a_{12} K_2)$$

$$K_2 = \frac{\lambda}{1-\lambda\Delta t a_{22}}(x_k + \Delta t a_{21} K_1)$$

$$K_1 = \frac{\lambda}{1-\lambda\Delta t a_{11}}\left(x_k + \Delta t a_{12}\frac{\lambda}{1-\lambda\Delta t a_{22}}(x_k + \Delta t a_{21}K_1)\right)$$
$$= \frac{\lambda}{1-\lambda\Delta t a_{11}}\left(x_k + \frac{\Delta t a_{12}\lambda}{1-\lambda\Delta t a_{22}}x_k + \frac{\Delta t^2 a_{12}a_{21}K_1\lambda}{1-\lambda\Delta t a_{22}}\right)$$
$$= \frac{\lambda}{1-\lambda\Delta t a_{11}}\left(\frac{(1-\lambda\Delta t a_{22})x_k + \lambda\Delta t a_{12}x_k}{1-\lambda\Delta t a_{22}} + \frac{\lambda\Delta t^2 a_{12}a_{21}}{1-\lambda\Delta t a_{22}}K_1\right)$$

$$\frac{(1-\lambda\Delta t a_{11})(1-\lambda\Delta t a_{22}) - \lambda^2\Delta t^2 a_{12}a_{21}}{\cancel{(1-\lambda\Delta t a_{22})}} K_1$$

$$= \frac{\lambda(1-\lambda\Delta t a_{22}) + \lambda^2\Delta t a_{12}}{\cancel{(1-\lambda\Delta t a_{22})}} x_k$$

$$K_1 = \underbrace{\frac{\lambda(1-\lambda\Delta t a_{22}) + \lambda^2\Delta t a_{12}}{(1-\lambda\Delta t a_{11})(1-\lambda\Delta t a_{22}) - \lambda^2\Delta t^2 a_{12}a_{21}}}_{\Delta_1} x_k$$

$$K_2 = \frac{\lambda}{1-\lambda\Delta t a_{22}}\left(x_k + \Delta t a_{21} K_1\right)$$

$$= \left[\frac{\lambda}{1-\lambda\Delta t a_{22}} + \frac{\lambda\Delta t a_{21}}{1-\lambda\Delta t a_{22}} \cdot \frac{\lambda(1-\lambda\Delta t a_{22}) + \lambda^2\Delta t a_{12}}{(1-\lambda\Delta t a_{11})(1-\lambda\Delta t a_{22}) - \lambda^2\Delta t^2 a_{12}a_{21}}\right] x_k$$

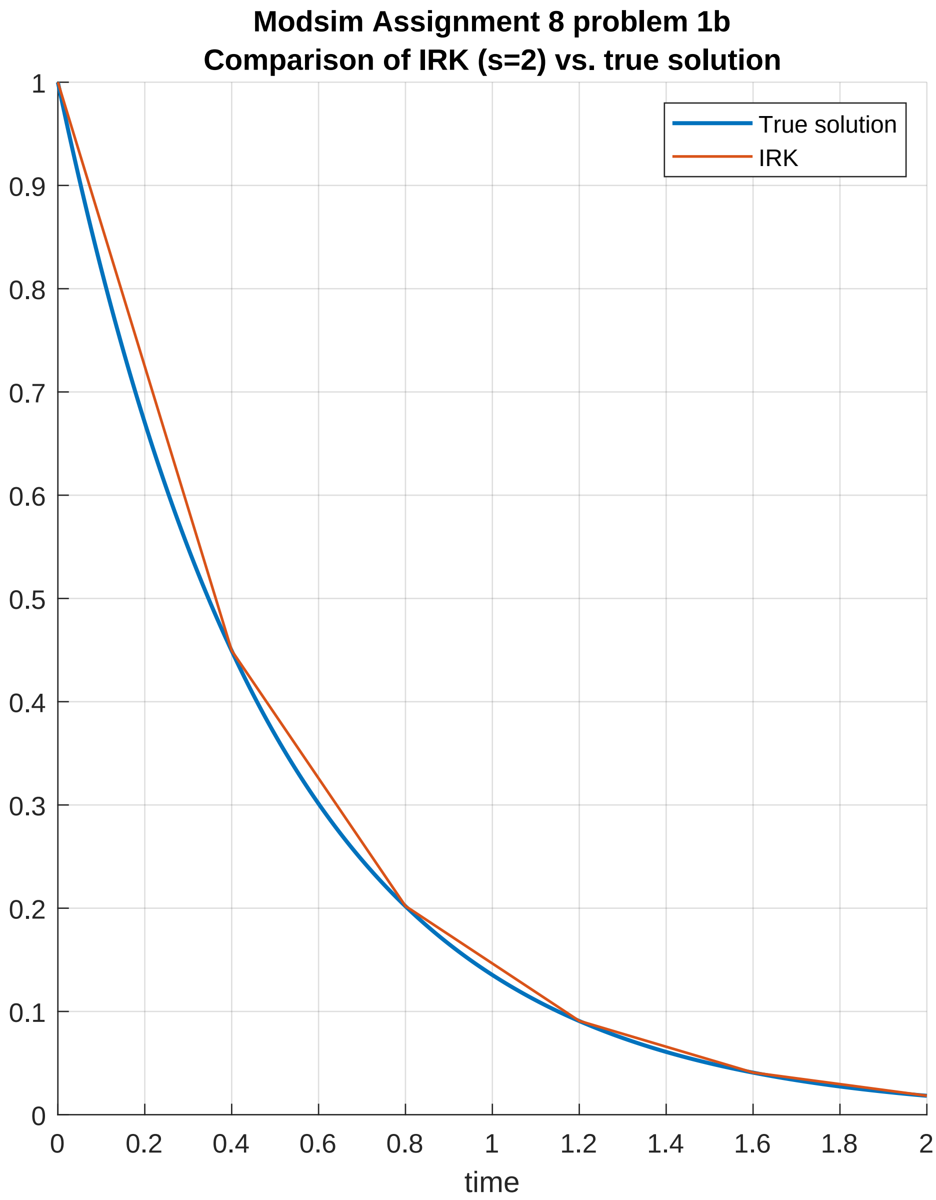
$$= \underbrace{\frac{\lambda(1-\lambda\Delta t a_{11})(1-\lambda\Delta t a_{22}) - \lambda^3\Delta t^2 a_{12}a_{21} + \lambda^2\Delta t a_{21}\left([1-\lambda\Delta t a_{22}] + \lambda\Delta t a_{12}\right)}{(1-\lambda\Delta t a_{22})\left([1-\lambda\Delta t a_{11}][1-\lambda\Delta t a_{22}] - \lambda^2\Delta t^2 a_{12}a_{21}\right)}}_{\Delta_2} x_k$$

$$x_{k+1} = x_k + \Delta t\, b_1 K_1 + \Delta t\, b_2 K_2$$

$$= \underbrace{(1 + \Delta t\, b_1 \Delta_1 + \Delta t\, b_2 \Delta_2)}_{\text{stable if } |\cdot| \leq 1} x_k$$

```matlab
function x = IRK(ButcherArray, f, dfdx, T, x0)
    % Returns the iterations of an IRK method using Newton's method
    % ButcherArray: Struct with the IRK's Butcher array
    %   A: Nstage x Nstage
    %   b: Nstage x 1
    %   c: Nstage x 1
    % f: Function handle
    %    Vector field of ODE, i.e., x_dot = f(t,x)
    % dfdx: Function handle
    %       Jacobian of f w.r.t. x
    % T: Vector of time points, 1 x Nt
    % x0: Initial state, Nx x 1
    %
    % x: IRK iterations, Nx x Nt
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Define variables
    % Allocate space for iterations (x) and k1,k2,...,ks
    A = ButcherArray.A;
    b = ButcherArray.b;
    c = ButcherArray.c;

    Nx = size(x0,1);
    Nt = size(T,2);
    Nstage = size(A,1);

    x = zeros(Nx, Nt);
    x(:,1) = x0;

    % K: Nx x Nstage matrix of Ki's
    % k: Nx*Nstage x 1 column vector of Ki
    K = zeros(Nx, Nstage);
    K = repmat(x0, 1, Nstage);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xt = x0; % initial iteration
    k = reshape(K,Nx*Nstage,1); % initial guess
    % Loop over time points
    for nt=2:Nt
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Update variables
        % Get the residual function for this time step
        % and its Jacobian by defining adequate functions
        % handles based on the functions below.
        % Solve for k1,k2,...,ks using Newton's method
        % Calculate and save next iteration value x_t
        dt = T(nt) - T(nt-1);
        t = T(nt);

        r  = @(k) IRKODEResidual(k,xt,t,dt,A,c,f);
        dr = @(k) IRKODEJacobianResidual(k,xt,t,dt,A,c,dfdx);

        k = NewtonsMethod(r,dr,k);
        K = reshape(k,Nx,Nstage);

        xt = xt + dt*K*b;
        x(:,nt) = xt;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
```

```matlab
end

function g = IRKODEResidual(k,xt,t,dt,A,c,f)
    % Returns the residual function for the IRK scheme iteration
    % k: Column vector with k1,...,ks, Nstage*Nx x 1
    % xt: Current iteration, Nx x 1
    % t: Current time
    % dt: Time step to next iteration
    % A: A matrix of Butcher table, Nstage x Nstage
    % c: c matrix of Butcher table, Nstage x 1
    % f: Function handle for ODE vector field
    Nx = length(xt);
    Nstage = size(A,1);
    K = reshape(k,Nx,Nstage);
    Tg = t+dt*c';
    Xg = xt+dt*K*A';
    g = reshape(K-f(Tg,Xg),[],1);
end

function G = IRKODEJacobianResidual(k,xt,t,dt,A,c,dfdx)
    % Returns the Jacobian of the residual function
    % for the IRK scheme iteration
    % k: Column vector with k1,...,ks, Nstage*Nx x 1
    % xt: Current iteration, Nx x 1
    % t: Current time
    % dt: Time step to next iteration
    % A: A matrix of Butcher table, Nstage x Nstage
    % c: c matrix of Butcher table, Nstage x 1
    % dfdx: Function handle for Jacobian of ODE vector field
    Nx = length(xt);
    Nstage = size(A,1);
    K = reshape(k,Nx,Nstage);
    TG = t+dt*c';
    XG = xt+dt*K*A';
    dfdxG = cell2mat(arrayfun(@(i) dfdx(TG(:,i),XG(:,i))',1:Nstage, ...
        'UniformOutput',false))';
    G = eye(Nx*Nstage)-repmat(dfdxG,1,Nstage).*kron(dt*A,ones(Nx));
end
```


Modsim Assignment 8 problem 1b
Comparison of IRK (s=2) vs. true solution
True solution
IRK
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time

Problem 2

(2) $\ddot{x} + g\left(1 - \left(\frac{x_d}{x}\right)^{\kappa}\right) = 0 \qquad x, x_d, g > 0 \qquad \kappa \geq 1$

(3) $E = \frac{mg}{\kappa - 1} \frac{x_d^{\kappa}}{x^{\kappa - 1}} + mgx + \frac{1}{2} m \dot{x}^2$

a) $\dot{E}(t) = \frac{mg\,x_d^{\kappa}}{\kappa-1} \cdot \frac{d}{dt} x^{1-\kappa} + mg\dot{x} + m\dot{x}\ddot{x}$

$= \frac{-mg\,x_d^{\kappa}}{\cancel{\kappa-1}} \cancel{(1-\kappa)} x^{-\kappa} \cdot \dot{x} + mg\dot{x} + m\dot{x}\ddot{x}$

$= -mg\,x_d^{\kappa} x^{-\kappa} \cdot \dot{x} + \cancel{mg\dot{x}} + mg\dot{x}\left[\left(\frac{x_d}{x}\right)^{\kappa} \cancel{-1}\right]$

$= \cancel{-mg\left(\frac{x_d}{x}\right)^{\kappa} \dot{x}} + 0 + \cancel{mg\left(\frac{x_d}{x}\right)^{\kappa} \dot{x}}$

$= 0$

b) $\dot{x}_1 = x_2$

$\dot{x}_2 = -g\left(1 - \left(\frac{x_d}{x_1}\right)^{\kappa}\right)$

$\Rightarrow \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -g\left(1 - \left(\frac{x_d}{x_1}\right)^{\kappa}\right) \end{bmatrix} = f(x)$

$x = [x_1 \;\; x_2]^T = [x \;\; \dot{x}]^T$

With Explicit Euler, the simulation becomes unstable and actually gains energy.

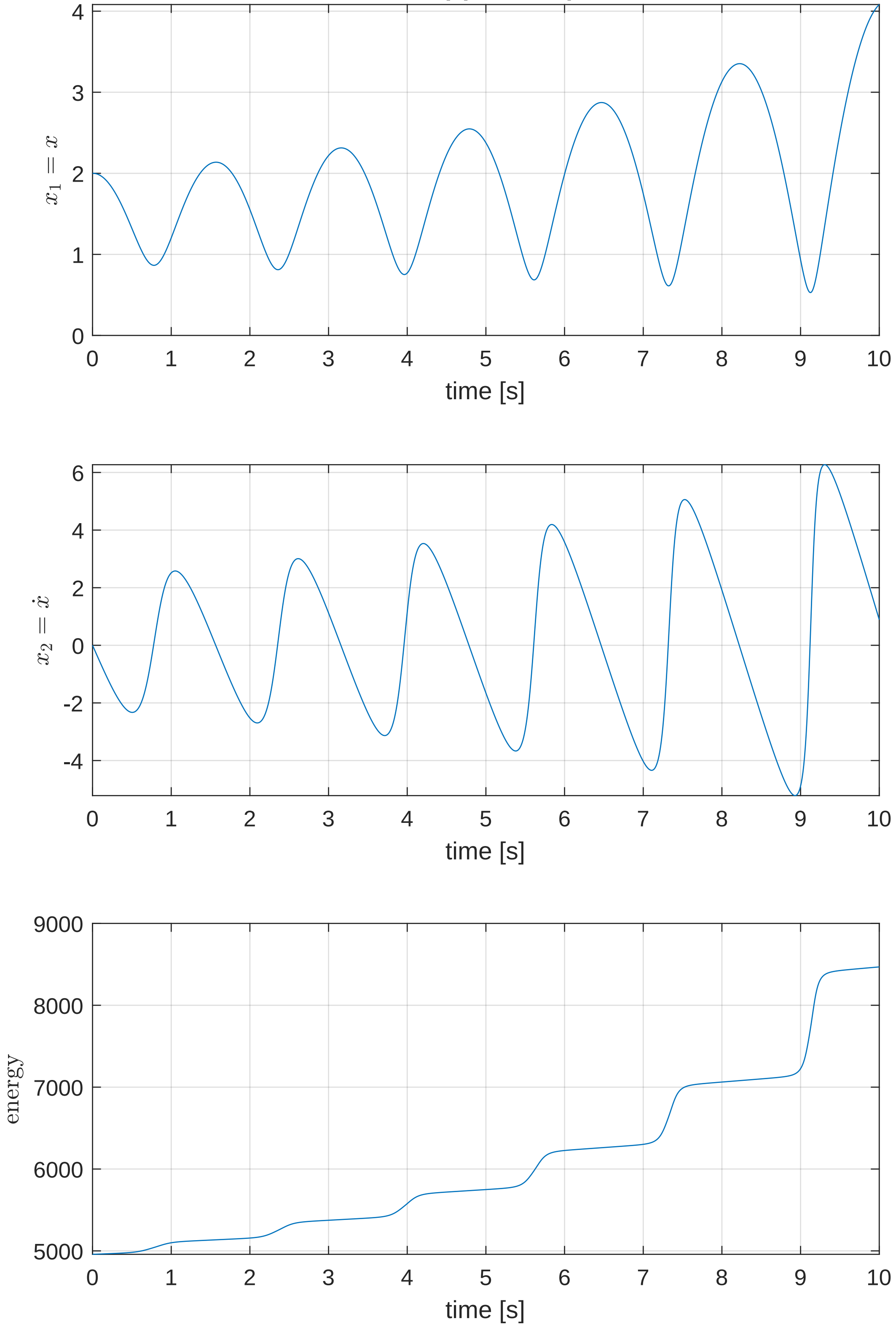
With Implicit Euler, the simulation is stable, but looses energy although from a) we know the energy is constant.

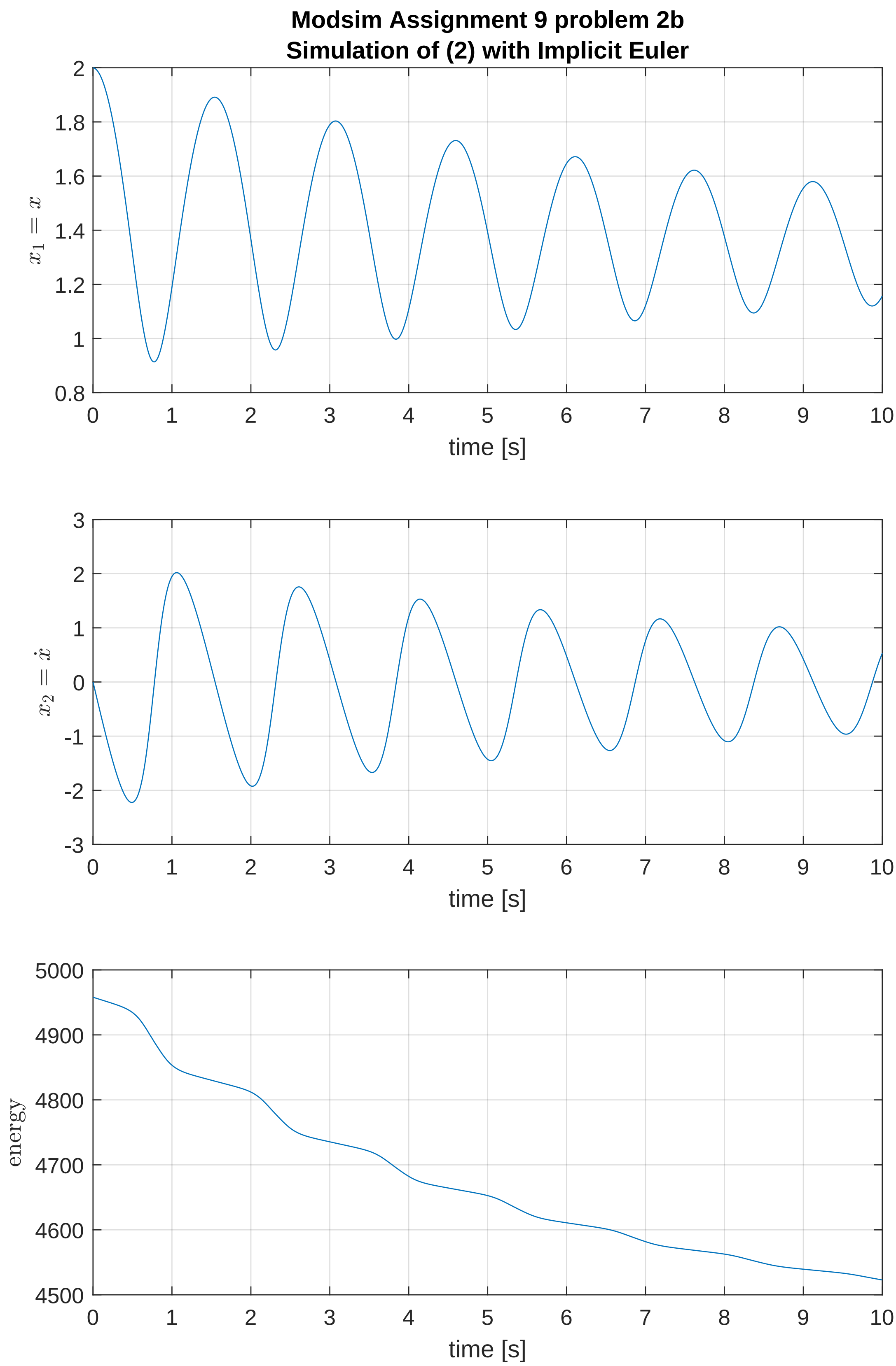
With the midpoint rule, the solution is marginally stable with standing oscillations. The energy fluctuate slightly, but at such a small amplitude we can approximate it as constant (fluctuations of ±0,00005).

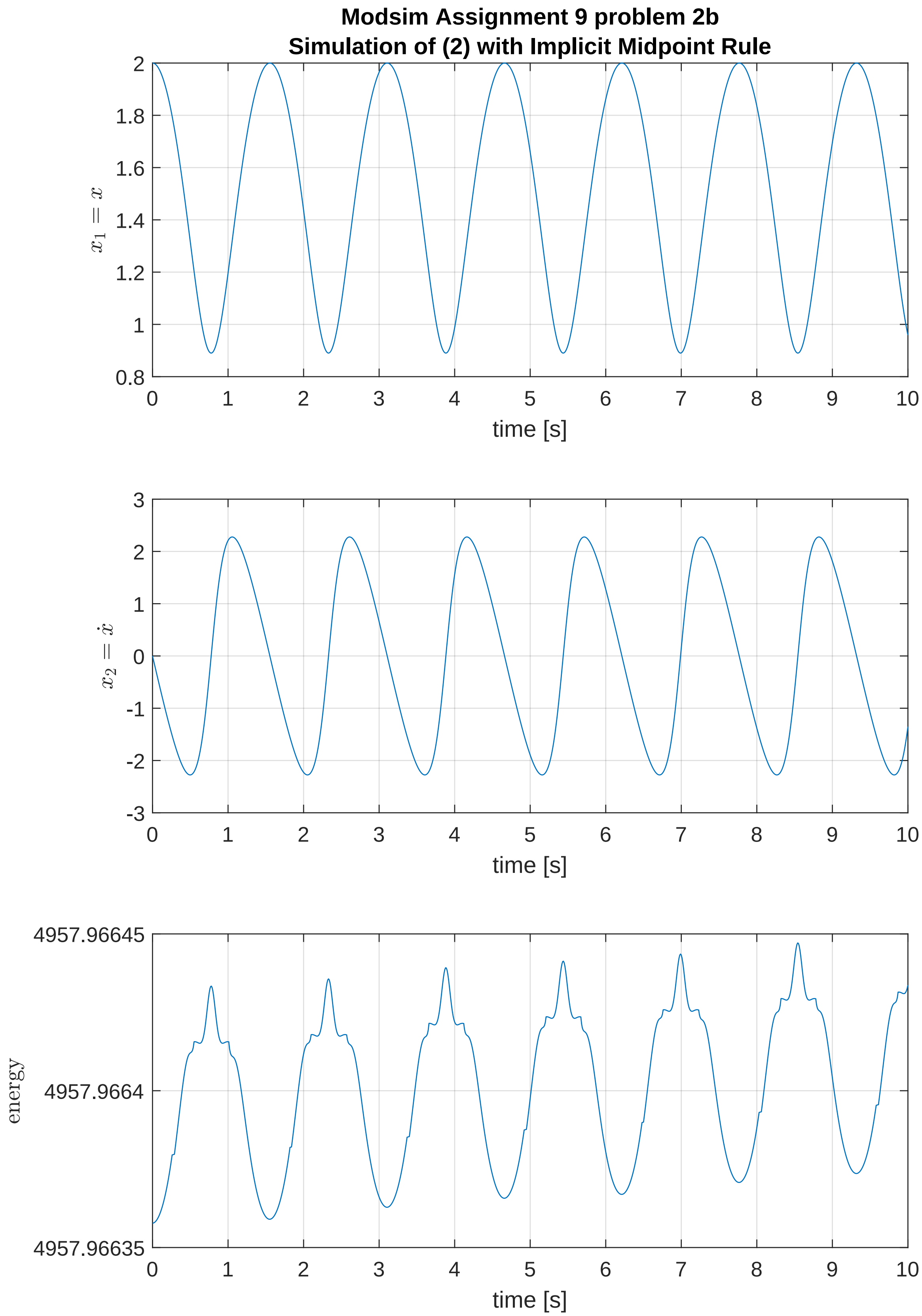
We know all IRK methods are A-stable. This means they can handle fast dynamics without becoming unstable. This is observed in the two implicit method plots.

Explicit methods are not A-stable and only stable within a specific region. For Explicit Euler, this is the unit circle around -1. Since we observe the method becoming unstable, we must be outside this region.

Modsim Assignment 9 problem 2b
Simulation of (2) with Explicit Euler




Modsim Assignment 9 problem 2b
Simulation of (2) with Implicit Euler
$x_1 = x$
time [s]
$x_2 = \dot{x}$
time [s]
energy
time [s]



Modsim Assignment 9 problem 2b
Simulation of (2) with Implicit Midpoint Rule
$x_1 = x$
$x_2 = \dot{x}$
energy
time [s]
time [s]
time [s]
2
1.8
1.6
1.4
1.2
1
0.8
3
2
1
0
-1
-2
-3
4957.96645
4957.9664
4957.96635
0
1
2
3
4
5
6
7
8
9
10

Problem 3

a) $c(q) = \frac{1}{2}(p^T p - L^2) = 0$, $L = 1$

$$\dot{p} = v$$

$$\dot{v} = -g\begin{bmatrix}0\\0\\1\end{bmatrix} - \overbrace{\frac{1}{m}\tilde{z}}^{z} p$$

$$0 = p^T \dot{v} + v^T v$$

$$x = \begin{bmatrix} p \\ v \end{bmatrix} \qquad x(0) = [1 \ 0 \ 0 \ \vdots \ 0 \ 1 \ 0]^T \in \mathbb{R}^6$$

$$\dot{x} = \begin{bmatrix} 0 & I \\ -zI & -g\begin{bmatrix}0 \\ 0 \\ 1\end{bmatrix} \end{bmatrix} x$$

When $\Delta t = 0{,}5$, the method becomes unstable, and no meaningful path is found.

When $\Delta t = 0{,}1$, the result is a nice graph, although a little choppy. This is due to the "long" time step.

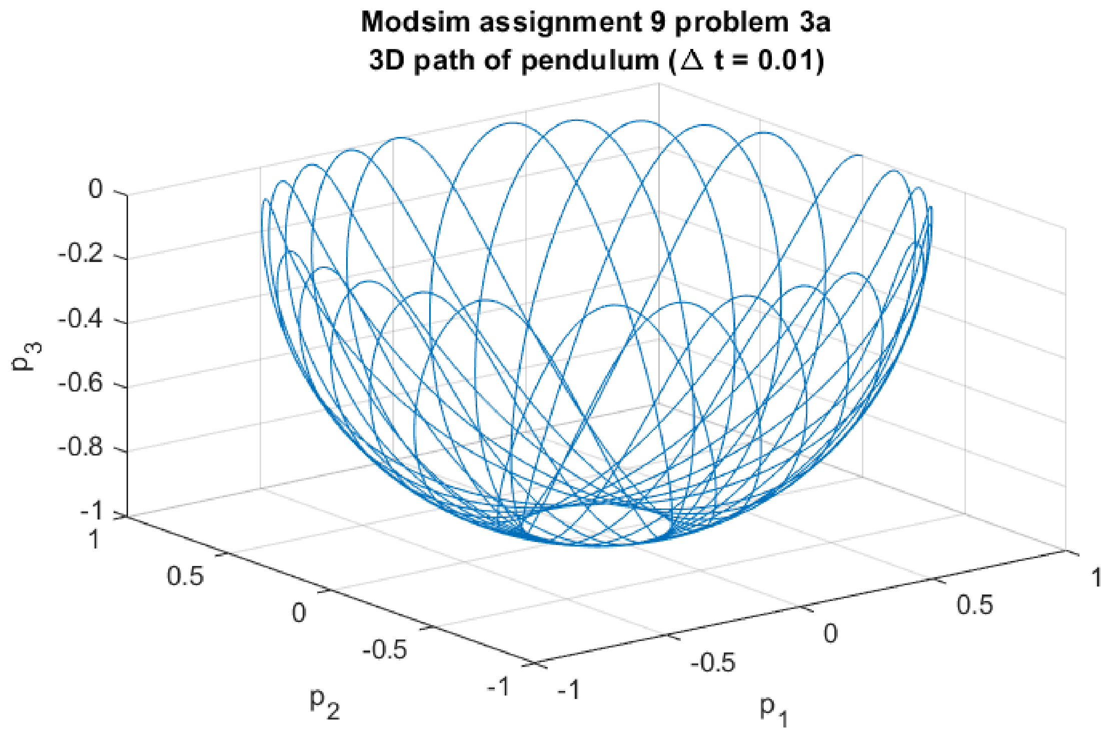
When $\Delta t = 0{,}01$, the result is much smoother.

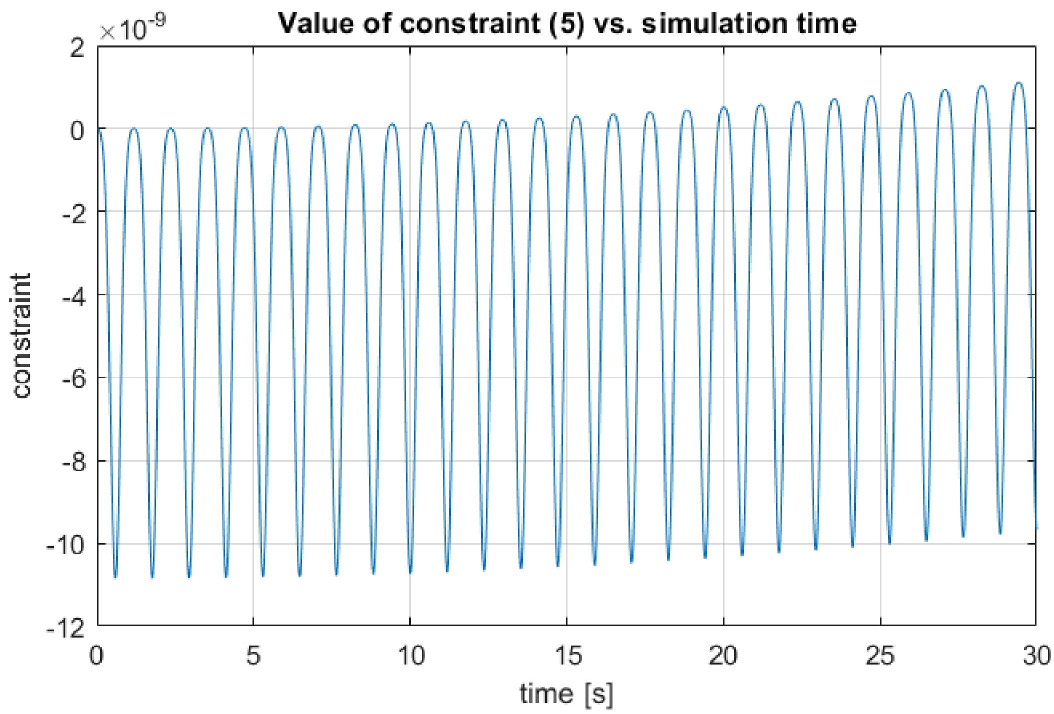
As we see, when $\Delta t = 0{,}1$ or $\Delta t = 0{,}01$, the constraint value oscillates a bit, but has generally low amplitude (10^{5} and 10^{-9} respectively).

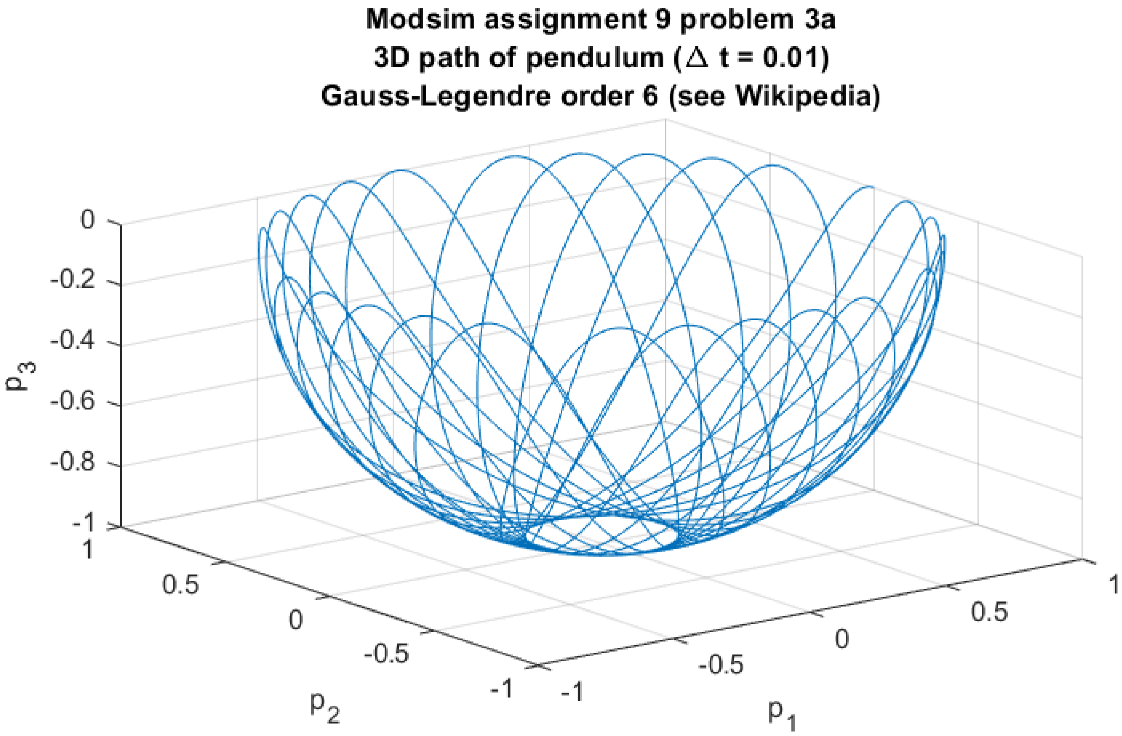
The reason for the oscillations may be the fixed time step introduces aliasing effects.

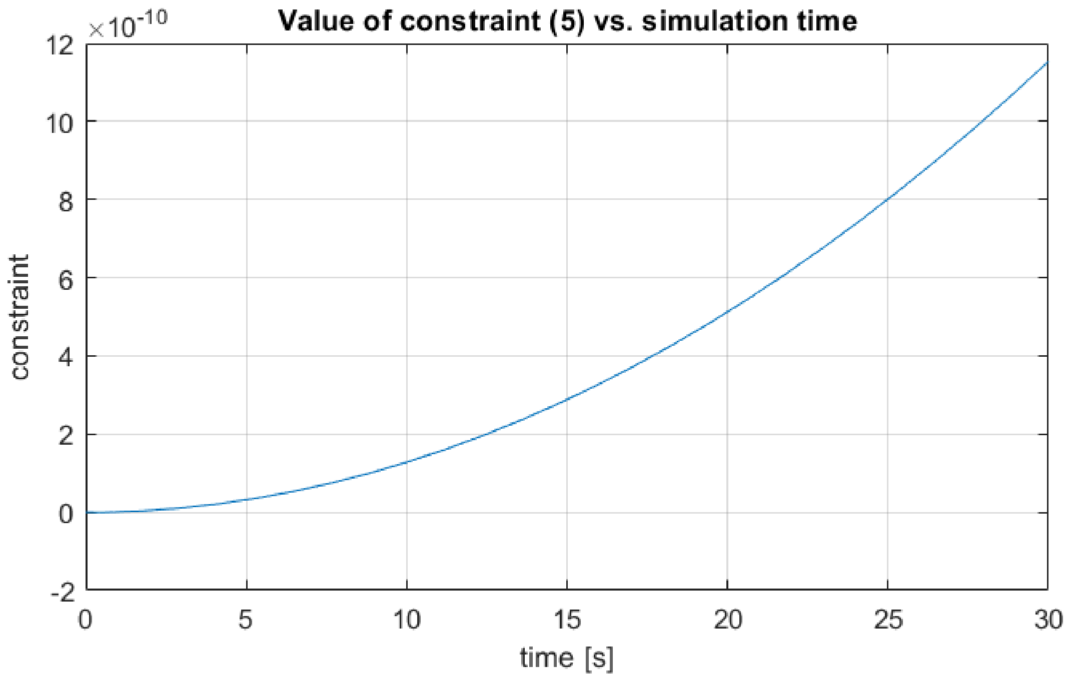
To improve the results, variable step length can be used. Alternatively, a different IRK method can be used. IRK6 (Butcher Tableau can be found on Wikipedia) was tried, and the constraint didn't oscillate and had slightly smaller amplitude.

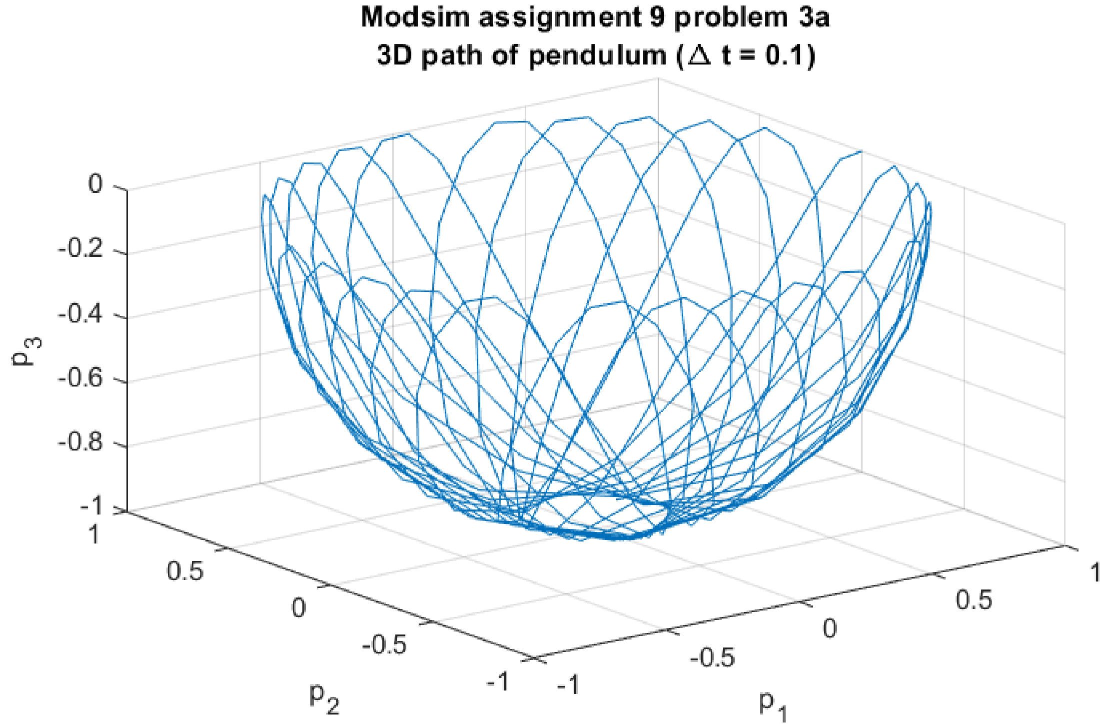
b) When using the model obtained directly from Lagrange, the Jacobian passed to the Newton's method is "singular to working precision". This is because z doesn't enter as a variable in the constraint and thus a value cannot be found for z.

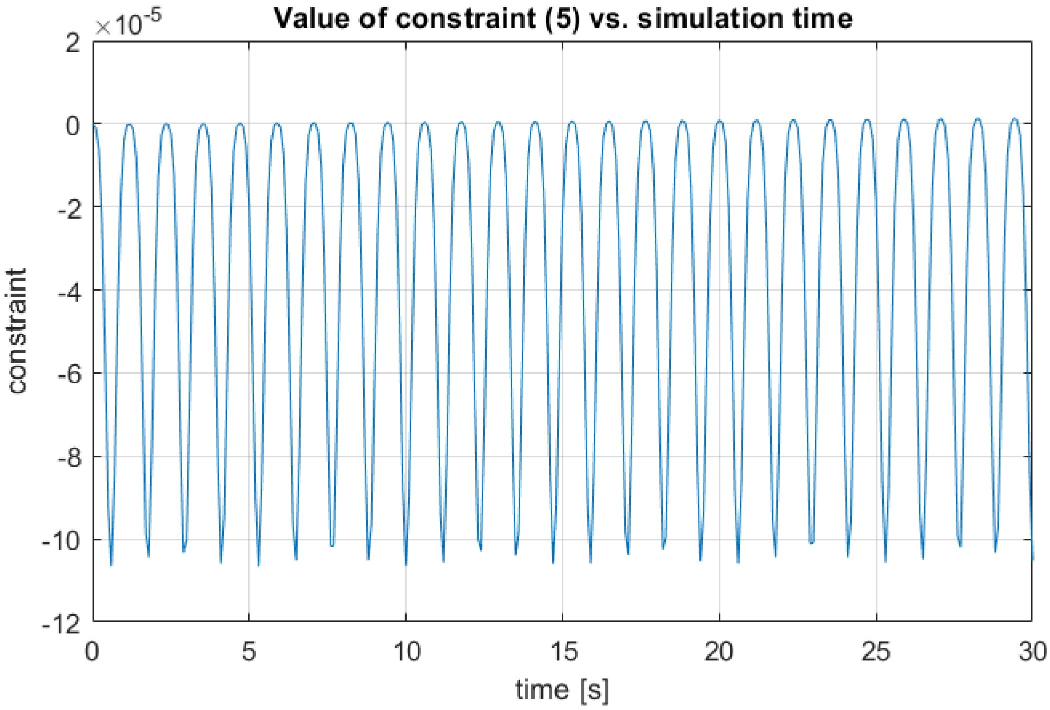

Modsim assignment 9 problem 3a
3D path of pendulum (Δ t = 0.01)
p3
p2
p1

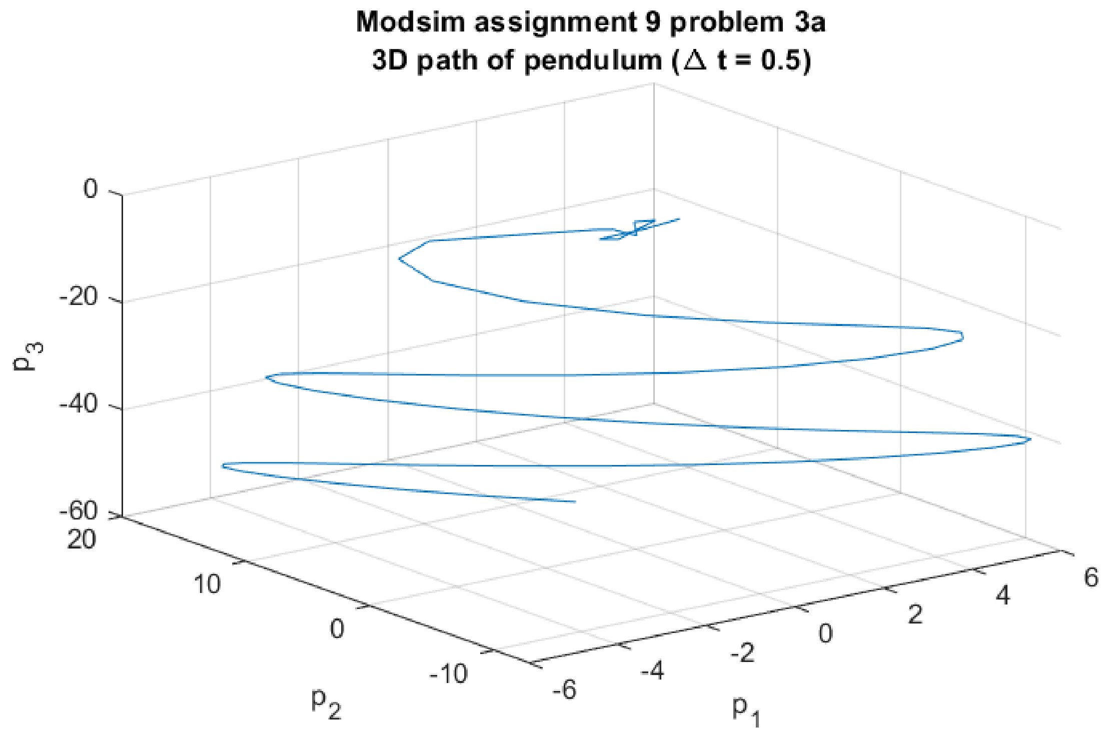

Value of constraint (5) vs. simulation time
×10^-9
constraint
time [s]

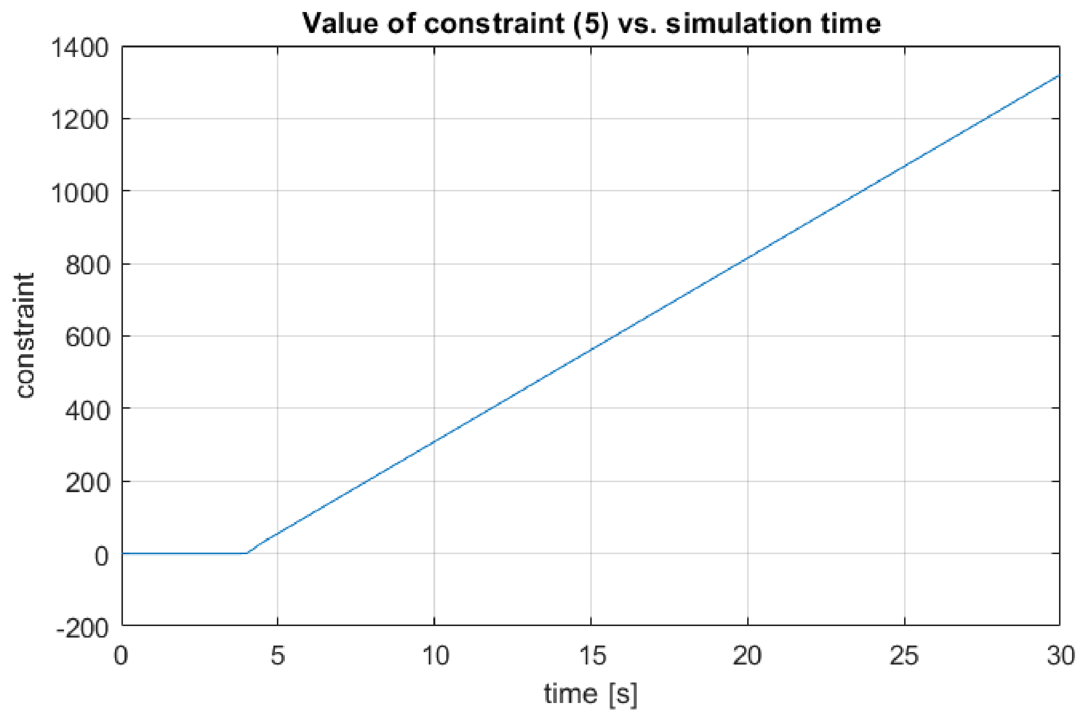

Modsim assignment 9 problem 3a
3D path of pendulum (Δ t = 0.01)
Gauss-Legendre order 6 (see Wikipedia)
0
-0.2
-0.4
-0.6
-0.8
-1
p3
1
0.5
0
-0.5
-1
p2
-1
-0.5
0
0.5
1
p1


Value of constraint (5) vs. simulation time
×10^-10
12
10
8
6
4
2
0
-2
constraint
0
5
10
15
20
25
30
time [s]


Modsim assignment 9 problem 3a
3D path of pendulum (Δ t = 0.1)
0
-0.2
-0.4
-0.6
-0.8
-1
p3
1
0.5
0
-0.5
-1
p2
-1
-0.5
0
0.5
1
p1


Value of constraint (5) vs. simulation time
×10^-5
2
0
-2
-4
-6
-8
-10
-12
constraint
0
5
10
15
20
25
30
time [s]


Modsim assignment 9 problem 3a
3D path of pendulum (△ t = 0.5)
0
-20
-40
-60
p3
20
10
0
-10
p2
-6
-4
-2
0
2
4
6
p1


Value of constraint (5) vs. simulation time
1400
1200
1000
800
600
400
200
0
-200
constraint
0
5
10
15
20
25
30
time [s]