TTK4130 - Modeling and Simulation
Assignement 9
Ingebrigt Stamnes Reinsborg

1a) I have added my code at the end of my submission. I believe the implicit equations to be solved are:

$$k = f(y_n + h(a_i k), t_n + c_i h)$$

$$y_{n+1} = y_n + h(bk)$$

(14.120 - 123 from the book)

which is:

$$x(:, nt) = x(:, nt-1) + delta\_t(k \cdot b');$$

(and the eq's for rf, J_r and k) in my code.

b) I simulated for both IRK4 and ERK4 in figure (1), but the result was more or less the same for both. Now, I'm inclined to not trust this result as I have a gut feeling that they should be slightly different.

I therefore decided to investigate changing the timeframe and -step.

tf	ts	figure
4s	1s	(2)
10s	1,395s	(3)

Increasing ts makes ERK less reliable, but IRK always hits the right values.

I can therefore deduce that this applies when $t_f = 2$ and $t_s = 0.4$. The IRK has correct values for its iterations.

Very cool.

c) The IRK is A-stable, so it won't matter what λ is. stable either way you tweak it.

2a)

$$\ddot{x} + g\left(1 - \left(\frac{x_d}{x}\right)^k\right) = 0 \quad (2)$$

$$x, x_d, g > 1 \qquad k \geq 1$$

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

$$\dot{E} = \frac{mg}{k-1}x_d^k(1-k)\cdot\frac{\dot{x}}{x^k} + mg\dot{x}$$

$$+ m\dot{x}\ddot{x}$$

$$= -mg\left(\frac{x_d^k}{x^k}\right)\cdot\dot{x} + mg\dot{x} + mg\dot{x}$$

$$\cdot\left(\left(\frac{x_d^k}{x^k}\right) - 1\right)$$

$$= 0 \qquad \square$$

b) All code can be found at the end.
see figure(4) and figure(5)
It seems that the only scheme that can show the energy being conserved is Implicit Midpoint or "G2" as I've called it.

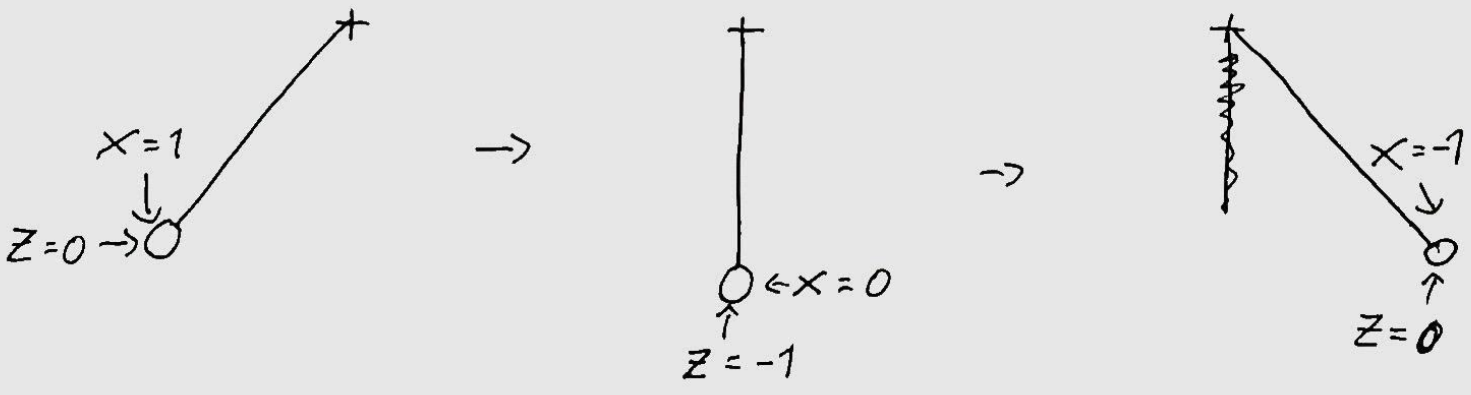
To show the conservation of energy, the scheme has to be stiffly accurate and L-stable.

The scheme has to be able to dampen $Re(j\omega h) \to \infty$

$\Rightarrow b = A^T e_0 \wedge A$ non-singular

Implicit midpoint (G2) is the only scheme that satisfies both of these requirements.

3a) See code for task3 in bottom of "main".
I plotted the simulated states in figure (6) and figure (7)
I couldn't quite understand how the constraint should have been implemented, however, the model/simulation made sense to me anyway.
It behaves like a normal 3D-pendulum, it swings back and forth along the "Y-axis" as I've called it



By decreasing the timestep, the model becomes marginally more accurate, but the computational time increases drastically. Increasing the timestep gives less computational time, but beyond $ts > 0.49$ (or something like that) the model sort of "collapses". I think using a higher order RK-method and a timestep in the goldilock-zone would probably give better results.

b) MatLab gave me a warning about a singular matrix. You can't have singular matrices in Newtons method as you'd end up "dividing by zero"