TTK 4130 MODELING & SIMULATION | ASSIGNMENT 9

ODIN A. SEVERINSEN | MTTK

1

a) See attached code

b) The IRK-method is very close to the exact solution

c) The IRK-method ~~does not~~ become ~~sg~~ unstable as long as the actual system is stable. This was tested up to $\lambda = -1400$

2 We have $\ddot{x} = -g\left(1 - \left(\frac{x_d}{x}\right)^{\kappa}\right)$, $E = \frac{mg}{\kappa-1}\frac{x_d^{\kappa}}{\underbrace{x^{\kappa-1}}_{x^{-(\kappa-1)} = x^{-\kappa+1}}} + mgx + \frac{1}{2}m\dot{x}^2$

and $\dot{E} = 0$

$$\Rightarrow \frac{dE}{dt} = \frac{\partial E}{\partial x}\cdot\dot{x} + \frac{\partial E}{\partial \dot{x}}\cdot\ddot{x}, \quad \frac{\partial E}{\partial x} = -mg\left(\frac{x_d}{x}\right)^{\kappa} + mg$$

$$\frac{\partial E}{\partial \dot{x}} = m\dot{x}$$

$$\Rightarrow \dot{E} = \overbrace{\left(-mg\left(\frac{x_d}{x}\right)^k + mg\right)}^{\partial E/\partial x}\overbrace{\dot{x}}^{\dot{x}} + \overbrace{m\dot{x}}^{\partial E/\partial \dot{x}}\overbrace{\left(-g\left(1-\left(\frac{x_d}{x}\right)^k\right)\right)}^{\ddot{x}}$$

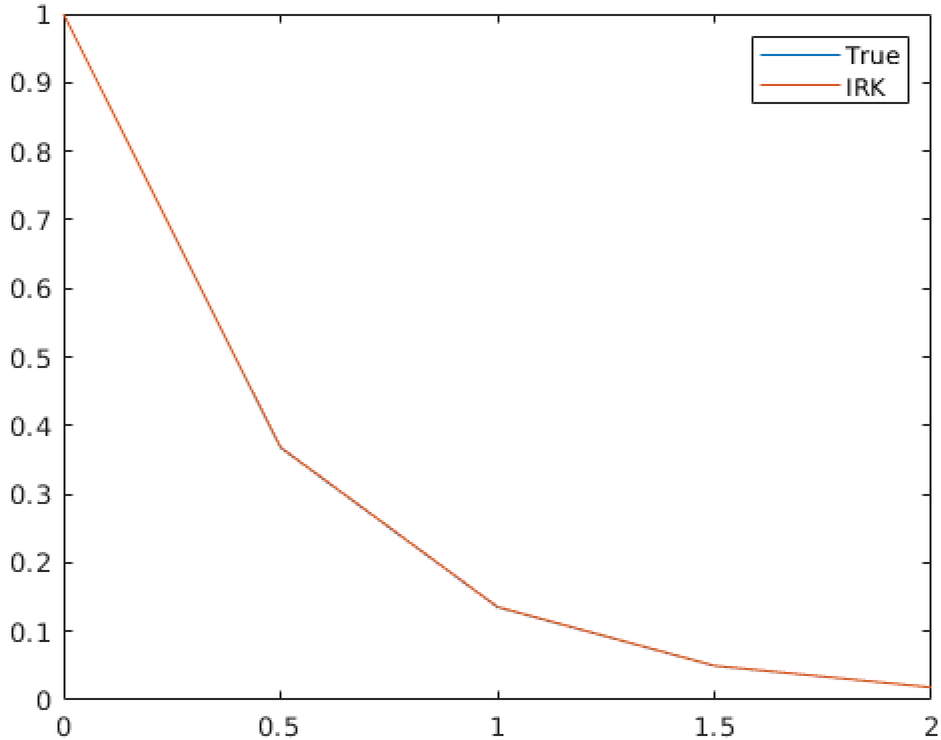
$$= \cancel{-mg\dot{x}\left(\frac{x_d}{x}\right)^k} + \cancel{m\dot{x}g} - \cancel{m\dot{x}g} + \cancel{m\dot{x}g\left(\frac{x_d}{x}\right)^k}$$

$$= \underline{\underline{0}}$$

q.e.d.

b) See attached code. We ~~[illegible]~~ probably have energy loss because machine precision, so the derivative of E is not perfect 0.

```
clear; close all;
% x = sym('x', [2 1]);
syms f x t real
lambda = 2;

f = -lambda*x;
J = jacobian(f, x);

f = matlabFunction(f, 'Vars', [t, {x}]);
dfdx = matlabFunction(J, 'Vars', [t, {x}]);

t0 = 0;
tf = 2;
dt = 0.4;
Nt = (tf - t0) / dt;
T = linspace(t0, tf, Nt);
x0 = 1;

A = [          1/4, 1/4-sqrt(3)/6;
     1/4+sqrt(3)/6,           1/4];
c = [1/2 - sqrt(3)/6; 1/2 + sqrt(3)/6];
b = [1/2; 1/2];

ButcherArray.A = A;
ButcherArray.b = b';
ButcherArray.c = c;


X = IRKTemplate(ButcherArray, f, dfdx, T, x0);

xtrue = x0*exp(-lambda*T);
plot(T, xtrue, T, X);
legend('True', 'IRK');
```



Published with MATLAB® R2019a

```
function x = IRKTemplate(ButcherArray, f, dfdx, T, x0)
    % Returns the iterations of an IRK method using Newton's method
    % ButcherArray: Struct with the IRK's Butcher array
    % 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
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tol=10^(-3);
    A=ButcherArray.A;
    b=ButcherArray.b;
    c=ButcherArray.c;
    x=zeros(length(x0),length(T));
    x(:,1) = x0; % initial iteration
    k = zeros(length(x0),length(A)); % initial guess
    % Loop over time points
    %r=IRKODEResidual(k(1,:),xt(:,1),1,del_t,A,c,f);
    r=1000;
    N=1000;
    Nt=length(T);
    alpha=1;
    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
        %
        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        del_t = T(nt) - T(nt - 1);
        K = x(:,nt)*ones(length(x), length(A));
        k_reshape=reshape(k,[length(x0)*length(A),1]);
        r=IRKODEResidual(k_reshape,x(:,nt-1),nt,del_t,A,c,f);
        while norm(r)>tol
            r=IRKODEResidual(k,x(:,nt-1),nt,del_t,A,c,f);
            J_r=IRKODEJacobianResidual(k,x(:,nt-1),nt,del_t,A,c,dfdx);
            d=@(del_k) J_r*del_k+r;
            J=@(del_k) J_r;
            del_k=NewtonsMethod(d,J,k',tol,N);
            k=k+alpha*del_k';
            k_reshape=reshape(k,[length(x0)*length(A),1]);
```

```
        end
        x(:,nt)=x(:,nt-1)+(del_t*b*k');
    end
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
```

```
Not enough input arguments.

Error in IRKTemplate (line 18)
    A=ButcherArray.A;
```

Published with MATLAB® R2019a

Table of Contents


```
clear; close all;
```

Set up parameters

```
xd = 1.32; % m
k = 2.40; % 1
g = 9.81; % ms^-2
m = 200; % kg
x0 = [2, 0]';
Nx = length(x0);
dt = 0.01; % s
t0 = 0;
tf = 10;
Nt = (tf - t0) / dt;
T = linspace(t0, tf, Nt);
```

Setup process model and its Jacobian

```
syms f t real
x = sym('x', [2 1]);

f = [x(2); -g.*(1-(xd./x(1)).^k)];
dfdx = jacobian(f, x);
E = m*g/(k - 1)*xd^k./(x(1).^(k - 1)) + m*g*x(1) + 1/2*m*x(2).^2;

f = matlabFunction(f, 'Vars', {t, x});
E = matlabFunction(E, 'Vars', {x});
dfdx = matlabFunction(dfdx, 'Vars', {t, x});
```

Solve with explicit Euler

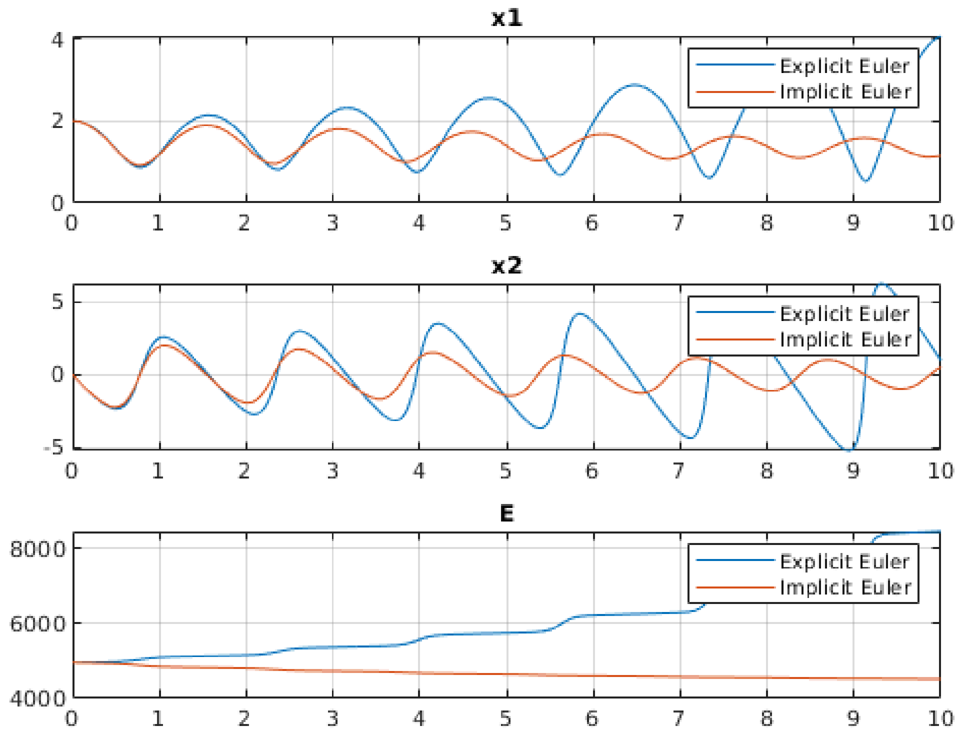
```
A = 0;
c = 0;
b = 1;

ButcherArray.A = A;
ButcherArray.b = b;
ButcherArray.c = c;

x_expeul = ERKTemplate(ButcherArray, f, T, dt, x0);
E_expeul = E(x_expeul);
```

Solve with Implicit Euler

```
x_impleul = ImplicitEulerTemplate(f, dfdx, T, x0);
E_impleul = E(x_impleul);
```

Plot results

```
figure(1);
subplot(3, 1, 1);
    plot(T, x_expeul(1,:), T, x_impleul(1,:));
    legend('Explicit Euler', 'Implicit Euler');
    grid on;
    title('x1');

subplot(3, 1, 2);
    plot(T, x_expeul(2,:), T, x_impleul(2,:));
    legend('Explicit Euler', 'Implicit Euler');
    grid on;
    title('x2');

subplot(3, 1, 3);
    plot(T, E_expeul, T, E_impleul);
    legend('Explicit Euler', 'Implicit Euler');
    grid on;
    title('E');
```



Published with MATLAB® R2019a