TTK4130 Assignment 8 Eirik Falck

Problem 1

a) See code below

b) $$f(x) = \begin{bmatrix} x_1 \cdot x_2 - 2 \\ \frac{x_1^4}{4} + \frac{x_2^3}{3} - 1 \end{bmatrix} = 0$$

The infinity norm increases from the initial value and then steadily fall to almost 0.
This convergence is (close to) quadratic.

c) The initial value starts close to a stationary point and jumps around a lot before settling on a solution.

The middle subplot shows the "jumpy" behaviour of the x value.

To improve Newtons method to circumvent this, we can reduce its step length.

d) We see a very quick convergece in the first iteration, before it becomes slower.

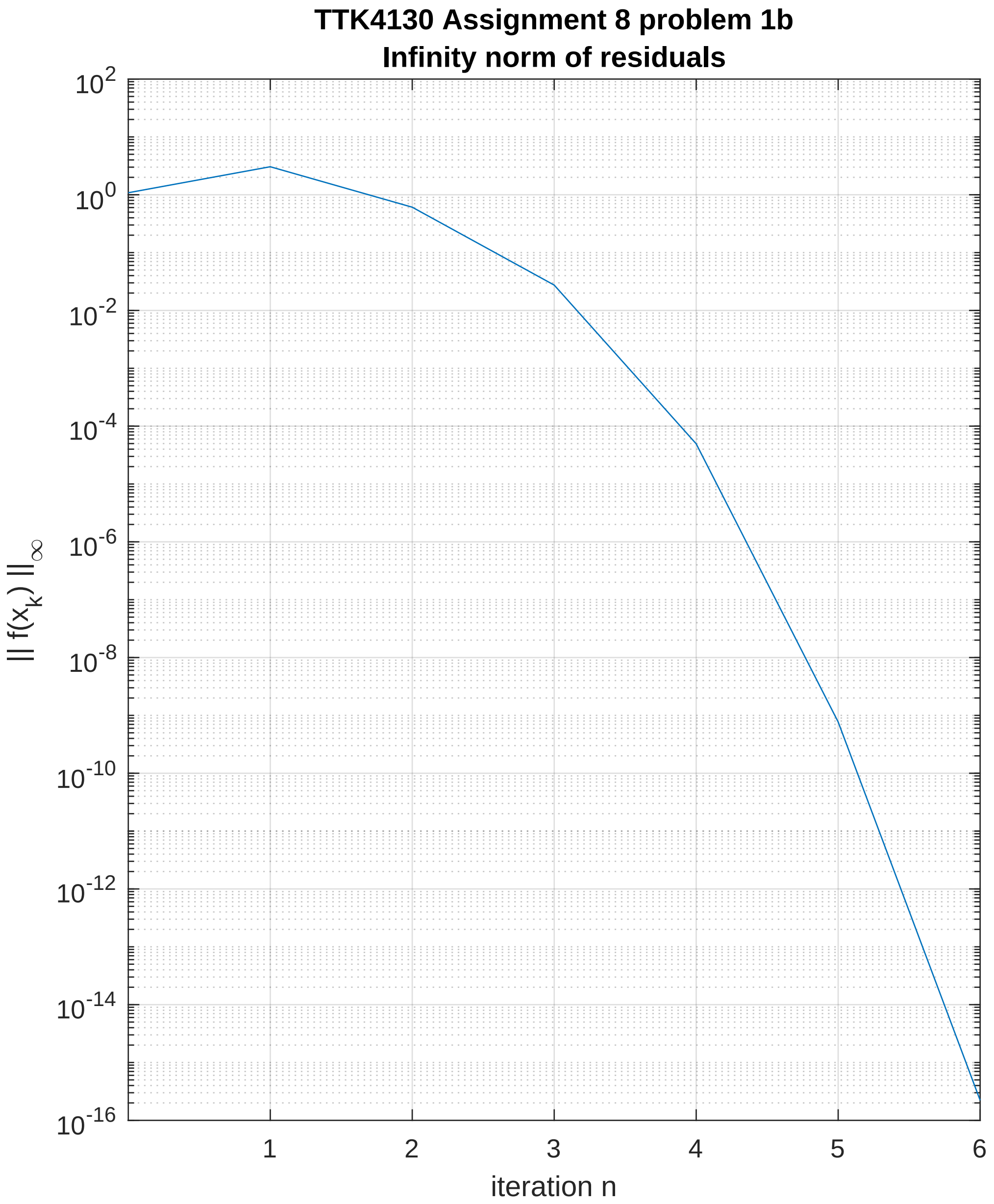
At $[1\ \pi]^T$ the Jacobian becomes rank defficient and the method becomes unreliable. The end point $[1\ 3.14]^T$ is close to this point.

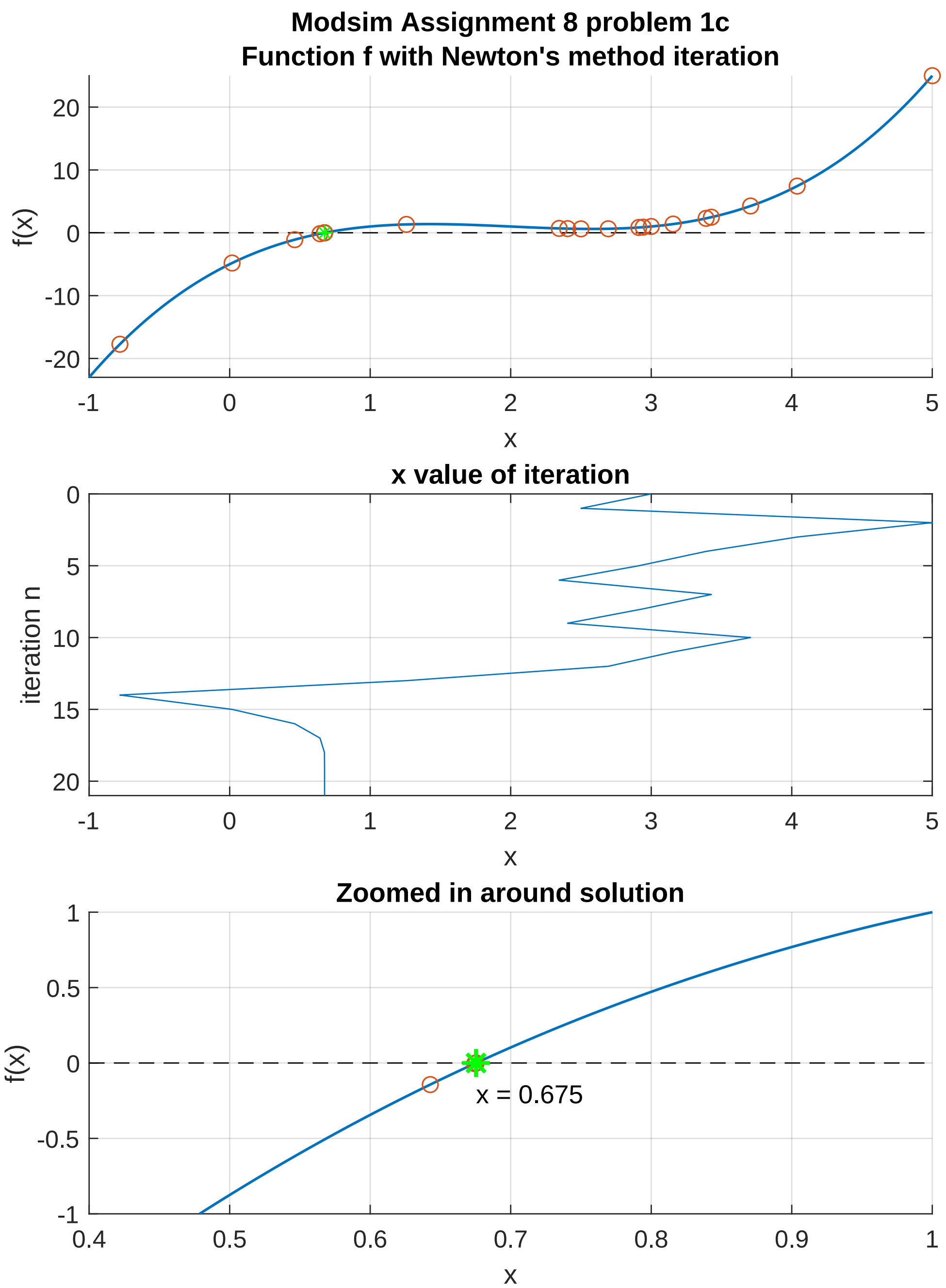
Newtons method can improved by approximating the gradient so it's always full rank.?

e) $x^* = [1\ \ 1]^T$

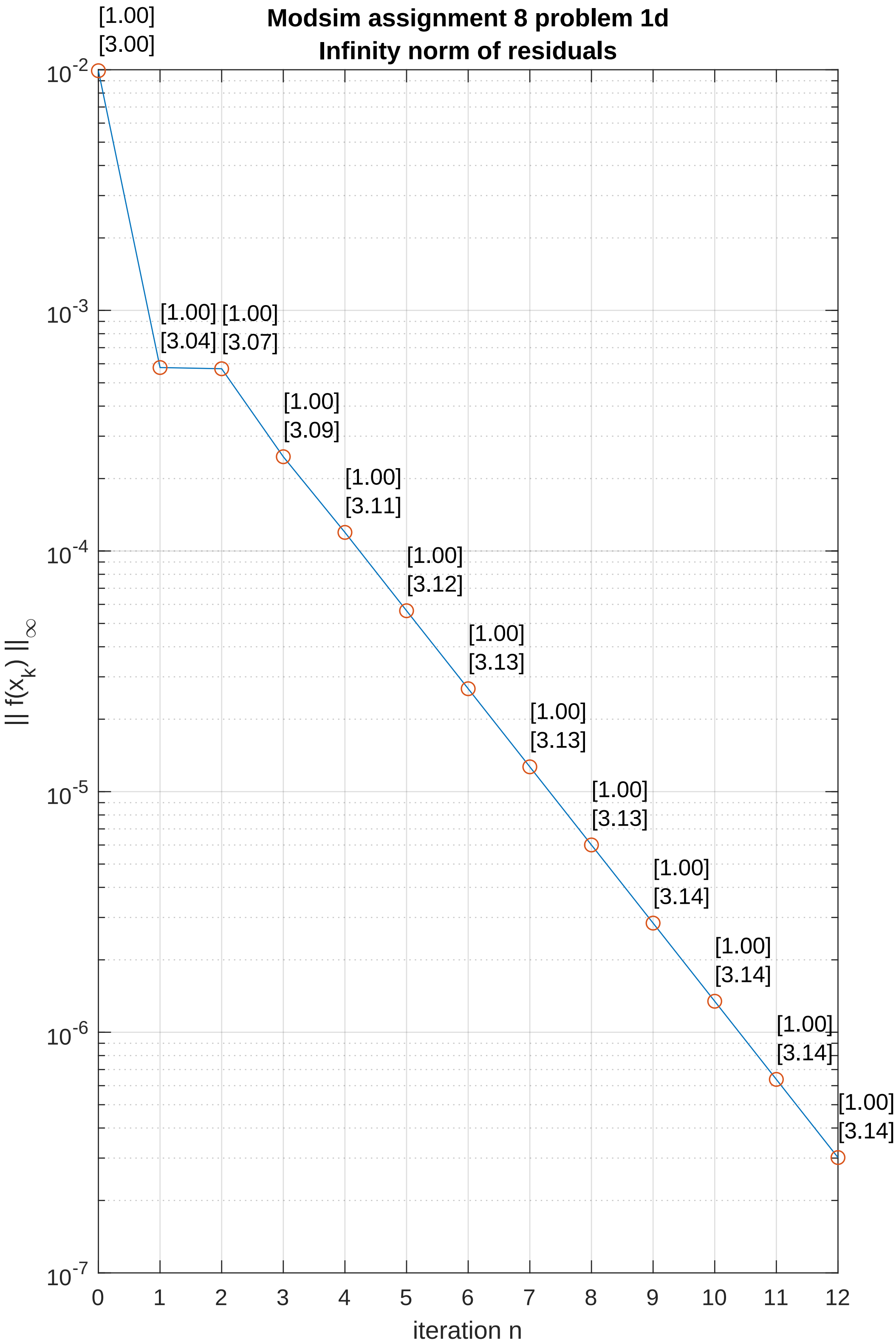
How do I calculate the convergence order? I've looked too long at $\|e_{n+1}\| = \mu \|e_n\|^p$ and taken the logarithm to try to figure something out. The only thing I find is the slope in the logarithmic plot which is -0.1761. Is this related to the order somehow?

```matlab
function X = NewtonsMethod(f, J, x0, tol, N)
    % Returns the iterations X of the Newton's method
    % f: Function handle
    %    Objective function, i.e. equation f(x)=0
    % J: Function handle
    %    Jacobian of f
    % x0: Initial root estimate, Nx x 1
    % tol: tolerance
    % N: Maximum number of iterations
    if nargin < 5
        N = 100;
    end
    if nargin < 4
        tol = 1e-6;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Define variables
    % Allocate space for iterations (X)
    nx = size(x0,1);
    X = NaN(nx,N+1);
    X(:,1) = x0;

    xn = x0; % initial estimate
    n = 1; % iteration number
    fn = f(xn); % save calculation
    % Iterate until f(x) is small enough or
    % the maximum number of iterations has been reached
    while norm(fn,Inf) > tol && n <= N
        % Calculate and save next iteration value x
        fn = f(xn);
        Jn = J(xn);
        dx = -Jn \ fn;
        xn = xn + dx;
        X(:,n+1) = xn;

        n = n + 1;
    end % while

    % remove NaN but keep shape of X
    X = X(:,1:n);
end % function
```

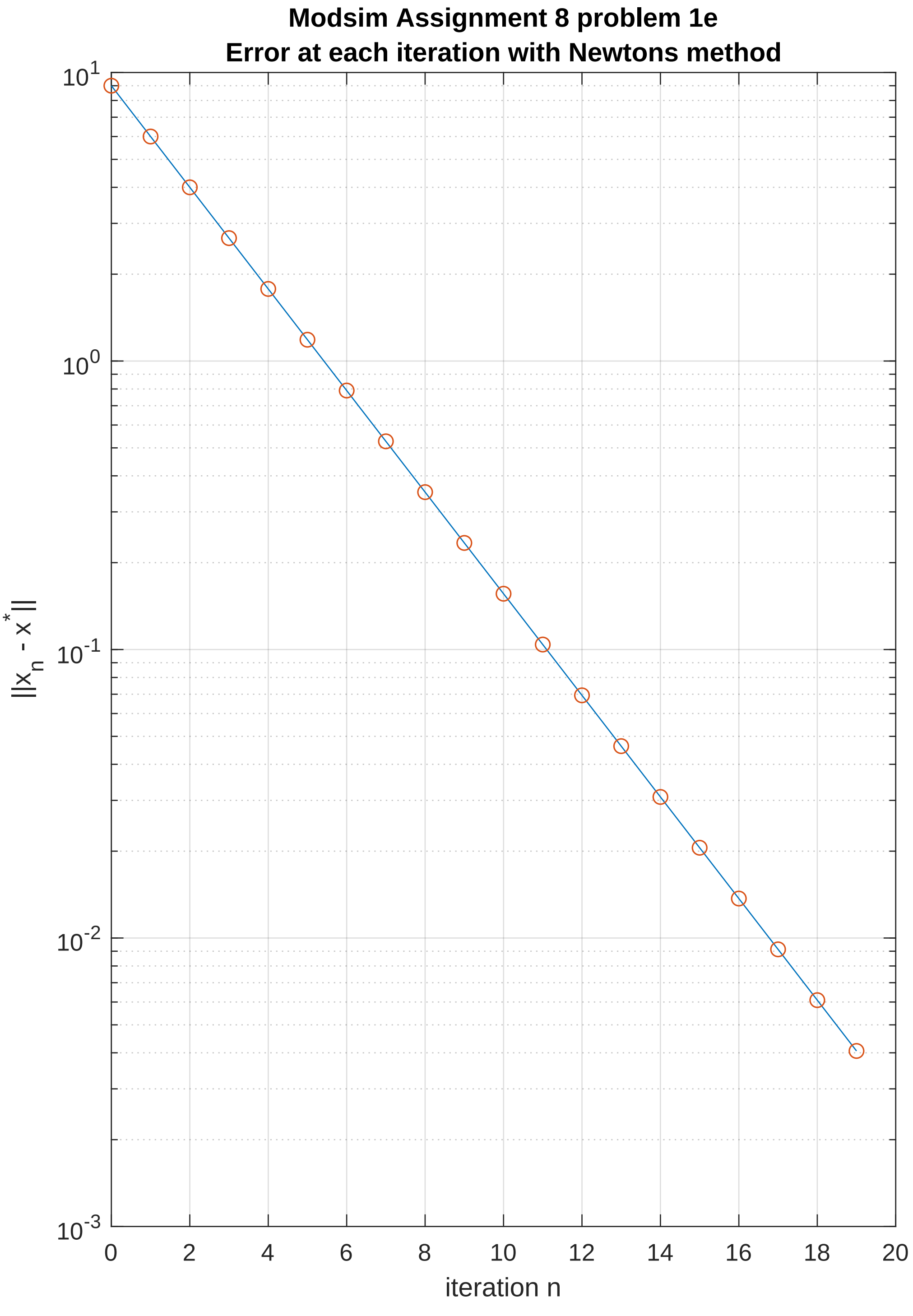

TTK4130 Assignment 8 problem 1b
Infinity norm of residuals
|| f(x_k) ||_∞
iteration n
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
10^-16
1
2
3
4
5
6



Modsim Assignment 8 problem 1c
Function f with Newton's method iteration
f(x)
x
x value of iteration
iteration n
x
Zoomed in around solution
f(x)
x = 0.675
x



Modsim assignment 8 problem 1d
Infinity norm of residuals
|| f(x_k) ||_∞
iteration n
10^-2
10^-3
10^-4
10^-5
10^-6
10^-7
0
1
2
3
4
5
6
7
8
9
10
11
12
[1.00] [3.00]
[1.00] [3.04]
[1.00] [3.07]
[1.00] [3.09]
[1.00] [3.11]
[1.00] [3.12]
[1.00] [3.13]
[1.00] [3.13]
[1.00] [3.13]
[1.00] [3.14]
[1.00] [3.14]
[1.00] [3.14]
[1.00] [3.14]


Modsim Assignment 8 problem 1e
Error at each iteration with Newtons method
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
$||x_n - x^*||$
0
2
4
6
8
10
12
14
16
18
20
iteration n

Problem 2

a) Implicit euler:

$$x_{k+1} = x_k + \Delta t \, f(t_k, x_{k+1})$$

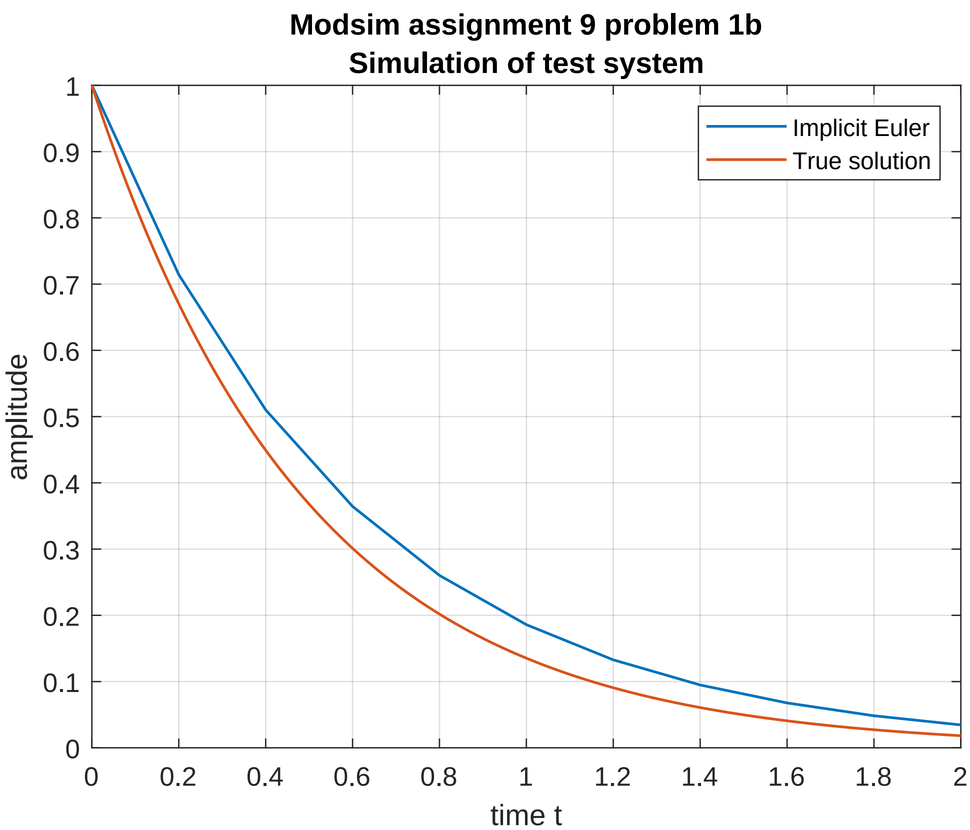
(1) $r(x_{k+1}) = x_k + \Delta t \, f(t_k, x_{k+1}) - x_{k+1}$

(2) $\dfrac{\partial r}{\partial x_{k+1}} = \Delta t \dfrac{\partial f}{\partial x_{k+1}} - I$

Solve (1) using Newtons method at each time step (this requires (2)).

b) As we can see, the implicit simulation is lagging somewhat behind the true solution, but seems to settle well towards the end since the error gets smaller.

```matlab
function x = ImplicitEuler(f, dfdx, T, x0)
    % Returns the iterations of the implicit Euler method
    % 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: Implicit Euler iterations, Nx x Nt
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Define variables
    % Allocate space for iterations (x)
    nx = size(x0,1);
    Nt = size(T,2);

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

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xt = x0;
    % Loop over time points
    for nt=2:Nt
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Update variables
        % Define the residual function for this time step
        % Define the Jacobian of this residual
        % Call your Newton's method function
        % Calculate and save next iteration value xt
        dt = T(nt) - T(nt-1);

        r = @(xt1) xt + dt * f(nt, xt1) - xt1;
        drdx = @(xt1) dt * dfdx(nt, xt1) - eye(nx);

        xt1 = NewtonsMethod(r, drdx, xt);
        xt1 = xt1(:,end); % last element is solution

        x(:,nt) = xt + dt * f(nt, xt1);
        xt = x(:,nt);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
end
```


Modsim assignment 9 problem 1b
Simulation of test system
Implicit Euler
True solution
amplitude
time t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2