TTK4130 - Modeling and Simulatio

Assignement 8

Ingebrigt Stamnes Reinsborg

1a) I have appended all relevant code for this assignement.

b) Semilogarithmic plot of the infinity norm of the residuals:

see: "fig 1: inf norm of residuals"

Comment of results:

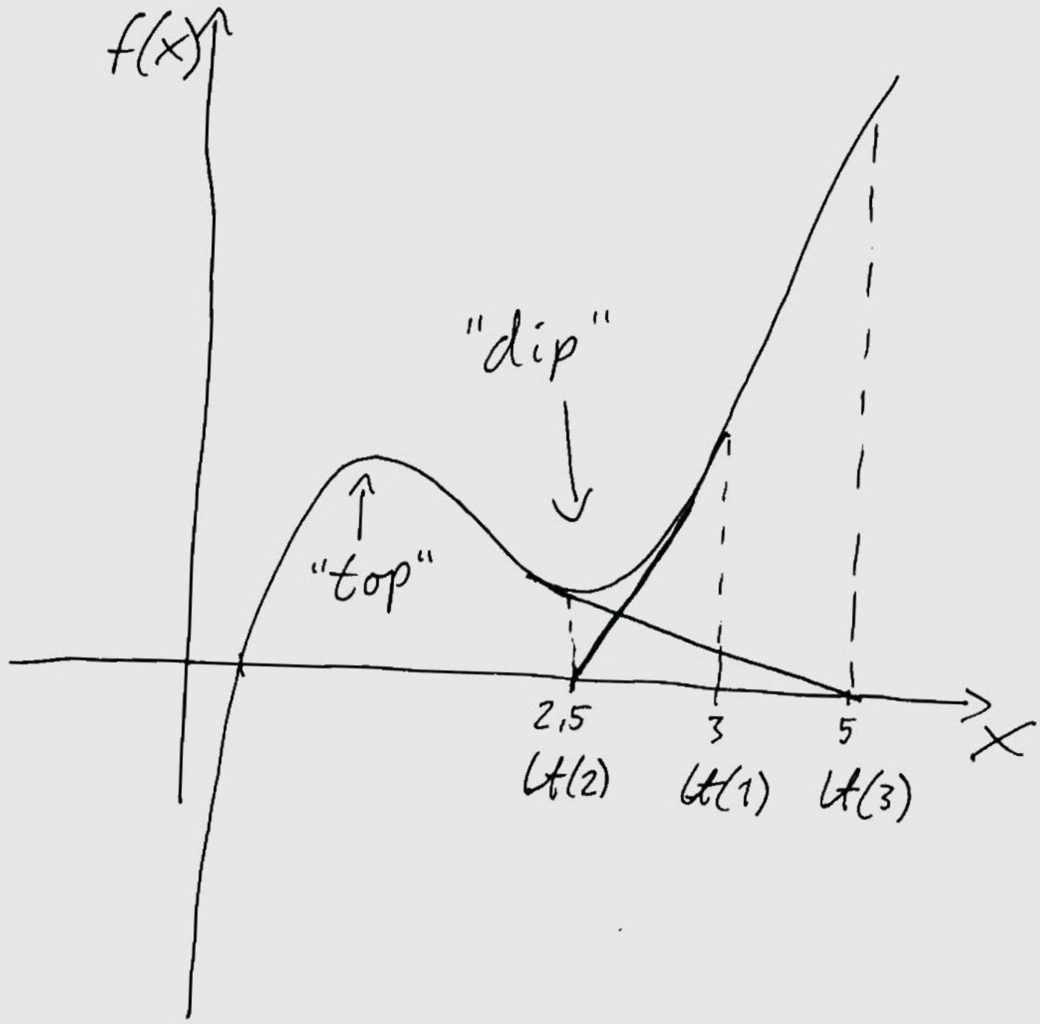
At first, there is a slight increase, however it then seems to decrease quadratically, which I believe fits well with ~~[illegible]~~ the theory

I've added the iteration values, see: "S_b' "

c) plot: "fig 2: Plots of obtained results in 1c)"

$$f(x) = (x-1)(x-2)(x-3) + 1$$

This looks something like this.



(Iteration values: "S_c")

One can see that the iteration values "bounce" around the so-called "dip"

The reason for this is that the gradients on both sides of the dip will send the next iteration to the other side until it manages "to escape the trap" and guess an x-value to the left of the "top" at which point it will converge very quickly.
(This wasn't very scientifically written... sorry...)
If an iteration or initial guess reaches a singular point, the jacobian becomes singular (non-invertible) and Newtons method would fail.
I could add a function that checks if an iteration is near a singular point, and if it is, starts again with a better initial guess.

d) Plot: "fig3: infNorm of residuals 1d)"

iteration values: "S_d"

The final iteration lands ~~approximately~~ near $\begin{bmatrix} 1 \\ \pi \end{bmatrix}$ which is the closest root according to the hint.

Since our initial guess is so close it converges quite fast.

If we were to guess something else, we would most likely get the same sort of problem we had in 1c, since these functions are nonlinear and contain trigonometric terms.

The Jacobian at the root is:

$$J_{dr} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

e) (Iteration values: "5_e")

According to Ernesto on the Forumpost "Question 1e Assignement 8", the rate of convergence (r_k) can be found by:

$$r_k = \log(e_{k+1}) - \frac{\log(e_k)}{(\log(e_k) - \log(e_{k-1}))}$$

so let's do that:

Residuals/errors pr. iteration:

k:	value($\cdot 10^3$)	r_k:
1	2.9160	undef
2	0.8640	7,96
3	0.2560	8,89
4	0.0759	6,67
5	0.0225	4,46
6	0.0067	

I'm not really sure how to interpret this, but from the iteration sequence, it seems to converge geometrically or asymptotically

2a) Implicit Euler:

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

$$\underbrace{x_k + \Delta t\, f(x_{k+1}, u_{k+1}) - x_{k+1} = 0}_{\text{Has to be solved for each iteration of IRK}}$$

The code can be found at the end of the document.

b) See plot: "fig 4: Implicit Euler compared to actual solution."

```
>> S_b'

ans =

   -1.0000   -1.0000
   -2.0417   -0.9583
   -1.7027   -1.1387
   -1.6058   -1.2394
   -1.6024   -1.2481
   -1.6024   -1.2481
```

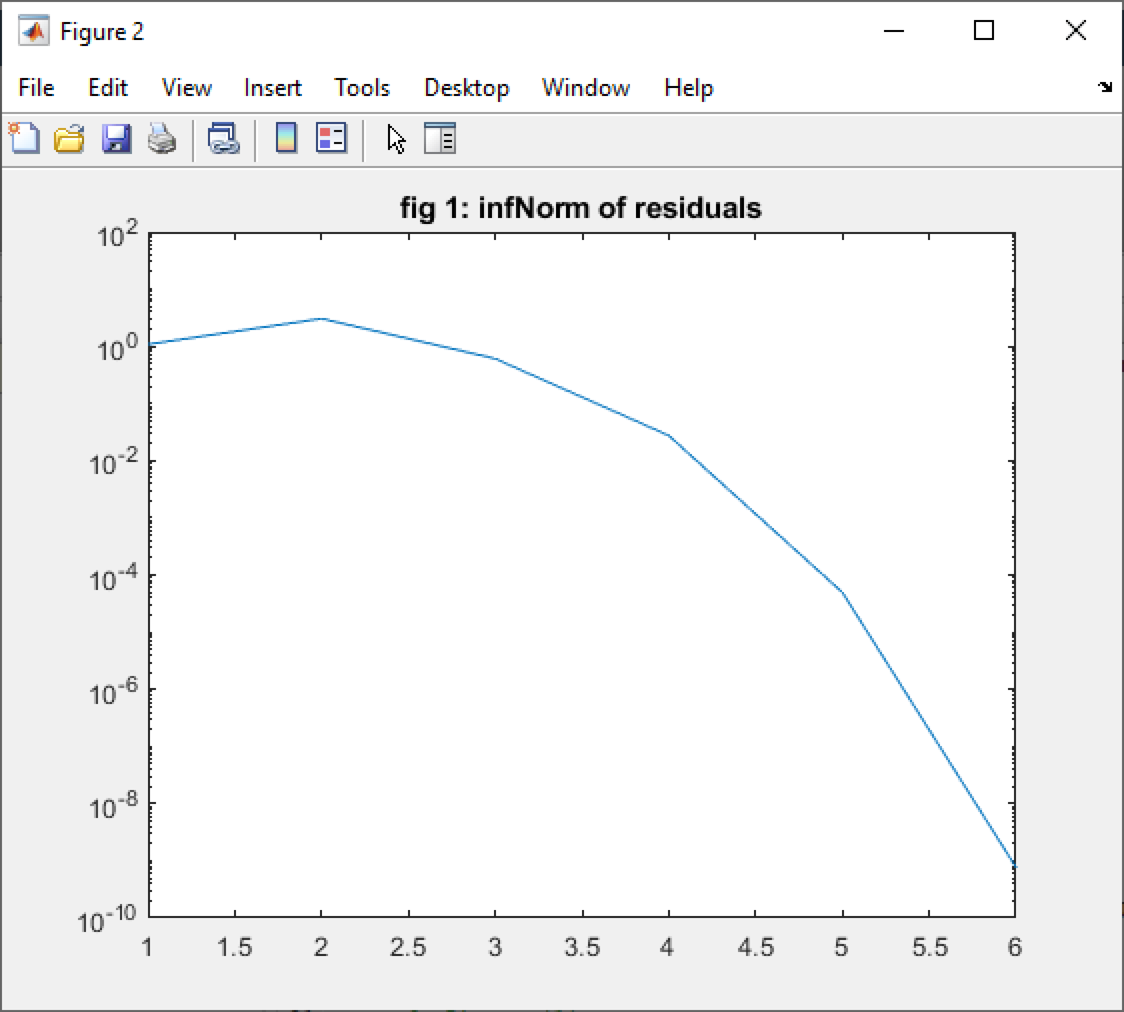


Figure 2
File Edit View Insert Tools Desktop Window Help
fig 1: infNorm of residuals
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6

```
>> S_c'

ans =

    3.0000
    2.5000
    5.0000
    4.0385
    3.3903
    2.9116
    2.3450
    3.4278
    2.9424
    2.4049
    3.7069
    3.1558
    2.6942
    1.2575
   -0.7813
    0.0173
    0.4631
    0.6427
    0.6743
    0.6753
    0.6753
```

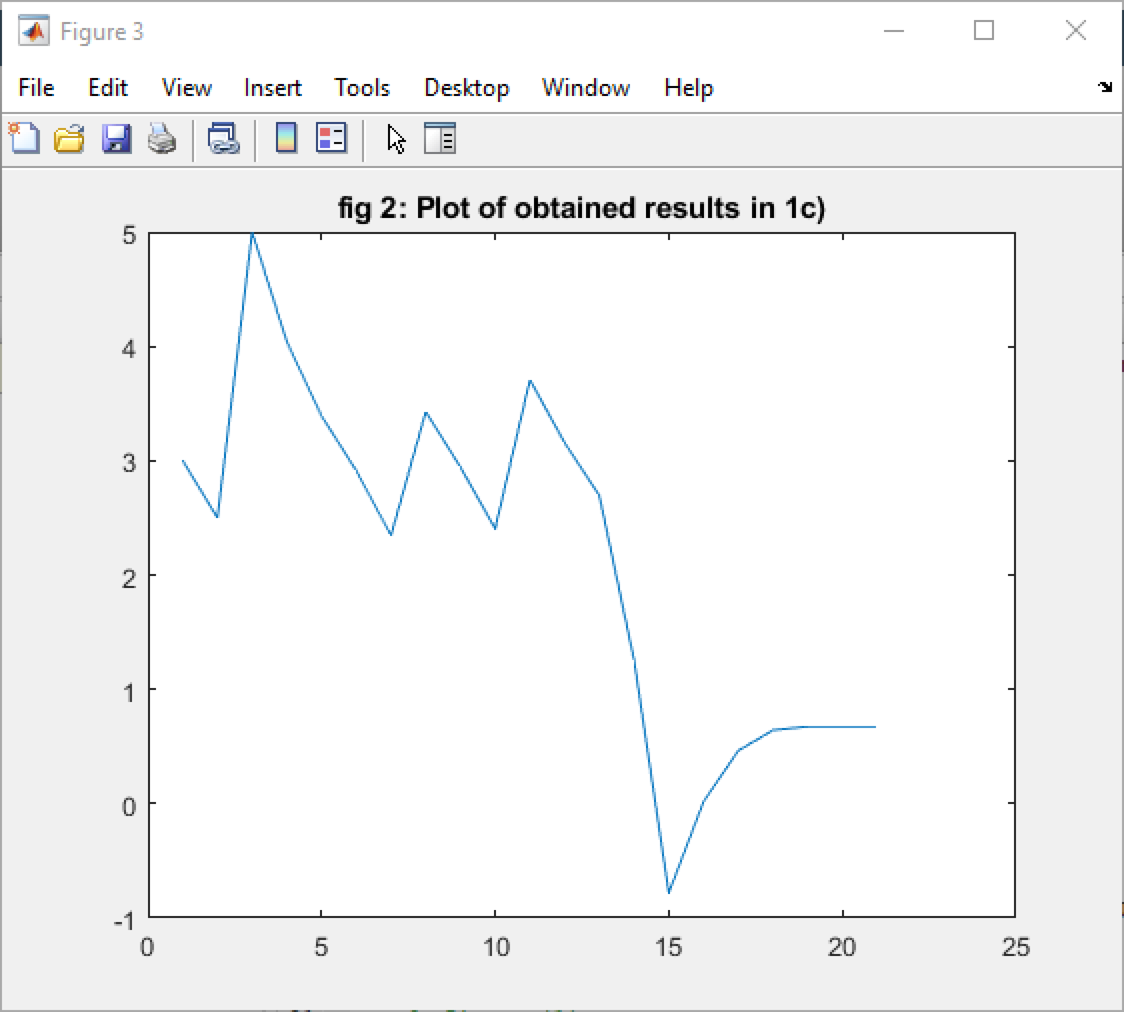


Figure 3
File Edit View Insert Tools Desktop Window Help
fig 2: Plot of obtained results in 1c)
5
4
3
2
1
0
-1
0
5
10
15
20
25

```
>> S_d'

ans =

    1.0000    3.0000
    1.0025    3.0357
    1.0010    3.0697
    1.0005    3.0920
    1.0002    3.1075
    1.0001    3.1181
    1.0001    3.1254
    1.0000    3.1305
    1.0000    3.1339
    1.0000    3.1363
    1.0000    3.1380
    1.0000    3.1391
```

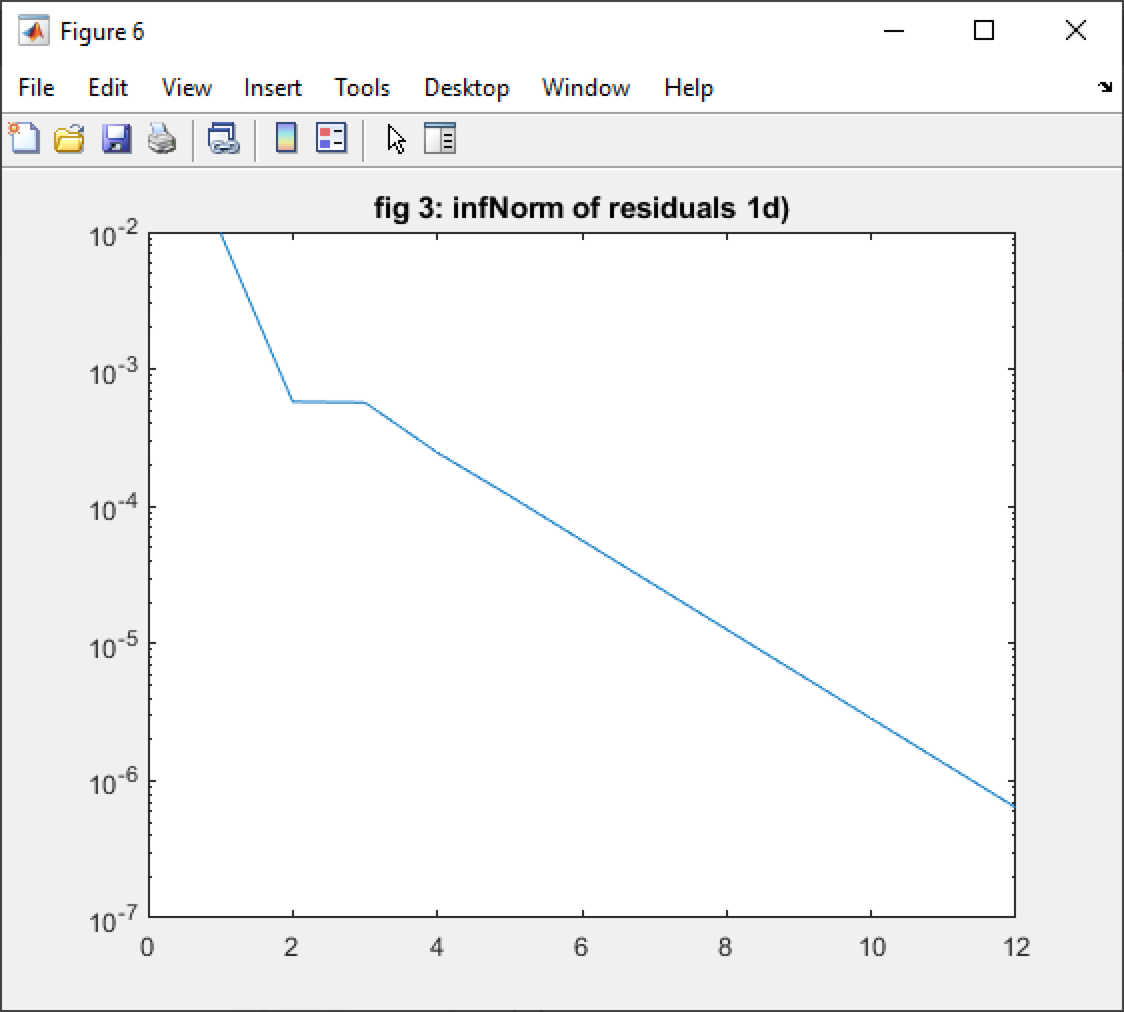


Figure 6
File Edit View Insert Tools Desktop Window Help
fig 3: infNorm of residuals 1d)
10^-2
10^-3
10^-4
10^-5
10^-6
10^-7
0 2 4 6 8 10 12

```
>> S_e'

ans =

   10.0000   10.0000
    7.0000    7.0000
    5.0000    5.0000
    3.6667    3.6667
    2.7778    2.7778
    2.1852    2.1852
    1.7901    1.7901
    1.5267    1.5267
    1.3512    1.3512
    1.2341    1.2341
    1.1561    1.1561
    1.1040    1.1040
    1.0694    1.0694
    1.0462    1.0462
    1.0308    1.0308
    1.0206    1.0206
    1.0137    1.0137
    1.0091    1.0091
    1.0061    1.0061
```

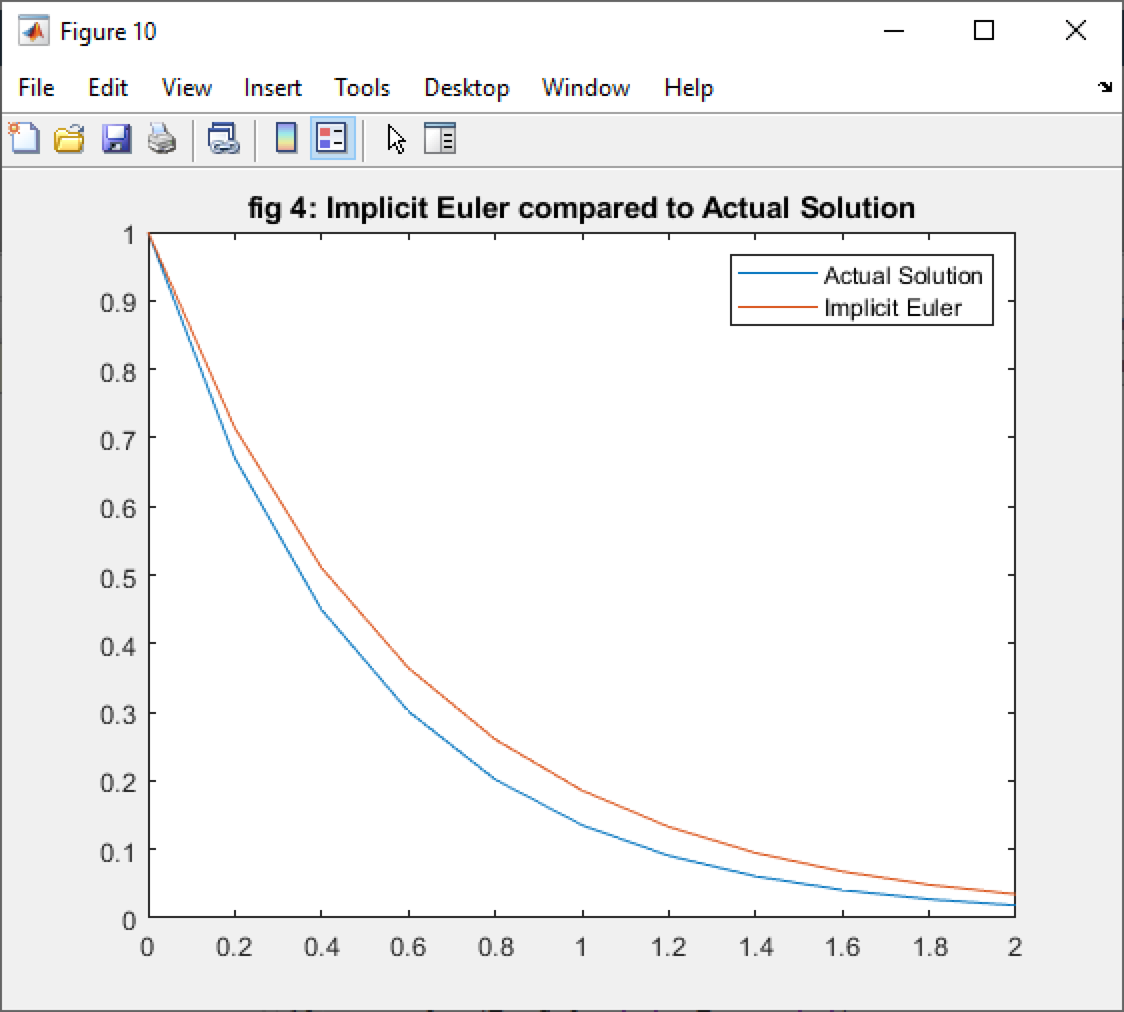


Figure 10
File
Edit
View
Insert
Tools
Desktop
Window
Help
fig 4: Implicit Euler compared to Actual Solution
Actual Solution
Implicit Euler
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

```matlab
%%Task 1a) and 1b)
syms x y;
f_sym1 = x*y - 2;
f_sym2 = ((x^4)/4) + ((y^3)/3) - 1;
f_sym = [f_sym1;
         f_sym2];

F = matlabFunction(f_sym, "vars", {[x;y]});

J_sym = jacobian(f_sym, [x;y]);
J = matlabFunction(J_sym, "vars", {[x;y]});

x0 = [-1;
      -1];
tol = 0.1;
N = 100;

[S_b, infNorm_b] = NewtonsMethodTemplate(F, J, x0);

%plots 1a) 1b)
% figure(1)
% loglog(infNorm_b);
% figure(2)
% semilogy(infNorm_b);
% title('fig 1: infNorm of residuals')


%%Task 1c)
f_c = (x - 1)*(x - 2)*(x - 3) + 1;
F_c = matlabFunction(f_c, "vars", x);
j_c = jacobian(f_c, x);
J_c = matlabFunction(j_c, "vars", x);
x_c_0 = 3;

[S_c, infNorm_c] = NewtonsMethodTemplate(F_c, J_c, x_c_0);

%plots 1c)
% figure(3)
% plot(S_c');
% title('fig 2: Plot of obtained results in 1c)')

%%Task 1d)
syms x_1 x_2;
f_d1 = x_1 - 1 + (cos(x_2)*x_1 + 1)*cos(x_2);
f_d2 = -x_1*sin(x_2)*(cos(x_2)* x_1 + 1);
f_d = [f_d1;
       f_d2];

F_d = matlabFunction(f_d, "vars", {[x_1;x_2]});
j_d = jacobian(f_d, [x_1;x_2]);
J_d = matlabFunction(j_d, "vars", {[x_1;x_2]});
```

```
x_d_0 = [1;
         3];

[S_d, infNorm_d] = NewtonsMethodTemplate(F_d, J_d, x_d_0);

%plots 1d)
% figure(6)
% semilogy(infNorm_d);
% title('fig 3: infNorm of residuals 1d)')

%%Task 1e)
f_e = 100*(x_2 - x_1)^2 + (x_1 - 1)^4;
df_e = [diff(f_e, x_1);
        diff(f_e, x_2)];

F_e = matlabFunction(df_e, "vars", {[x_1;x_2]});

j_e = jacobian(df_e, [x_1;x_2]);
J_e = matlabFunction(j_e, "vars", {[x_1;x_2]});

x_e_0 = [10;
         10];

[S_e, infNorm_e] = NewtonsMethodTemplate(F_e, J_e, x_e_0);

%plots 1e)
figure(7)
plot(infNorm_e,);
title('')
```

```
Error using dbstatus
Error: File: D:\OneDrive\Dokumenter\NTNU\ModSim\Ass8\modsim8\main.m
 Line: 80 Column: 16
Invalid expression. When calling a function or indexing a variable,
 use parentheses. Otherwise, check for mismatched delimiters.
```

Published with MATLAB® R2019a

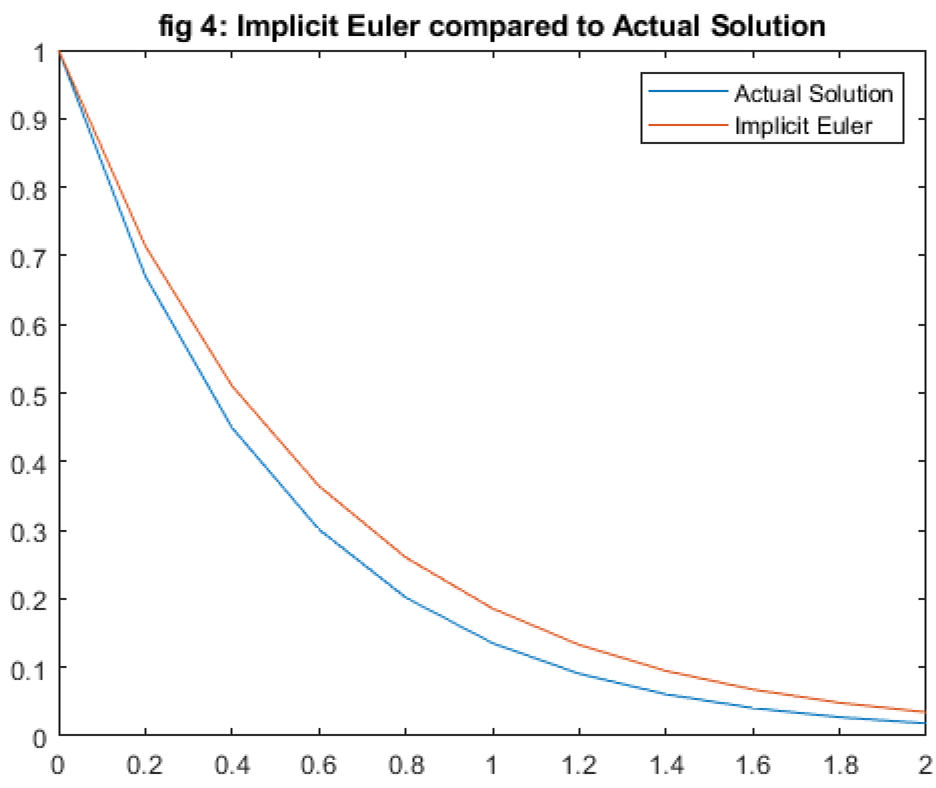
```
function [X,infNorm] = NewtonsMethodTemplate(f, J, x0, tol, N)
    % Returns the iterations of the Newton's method
    % f: Function handle
    %    Objective function, i.e. equation f(x)=0
    % J: Function handle
    %    Jacobian of f
    % x0: Initial root estimate, Nx x 1
    % tol: tolerance
    % 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 = zeros(Nx, N);
    X(:,1) = x0;
    infNorm = zeros(N,1);
    %r_k = zeros(N,1);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xn = x0; % initial estimate
    n = 1; % iteration number, change to 2 if you need x0 in
 iter.values
    fn = f(xn); % save calculation
    infNorm(1) = norm(fn,Inf);
    %r_k(1) = 0
    % Iterate until f(x) is small enough or
    % the maximum number of iterations has been reached
    iterate = norm(fn,Inf) > tol;
    while iterate
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Calculate and save next iteration value x
        dx = -J(xn)\f(xn);
        xn = xn + dx;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        fn = f(xn);
        X(:,n) = xn;
        n = n + 1;
        % save calculation for next iteration
        % Continue iterating?
        infNorm(n) = [norm(fn,Inf)];
        iterate = norm(fn,Inf) > tol && n <= N;
    end
    X = X(:,1:n-1);
    if n > N
```

```
        fprintf('No more iteration left because of Corona-hoarders,
 sorry')
    end;
end
```

```
Not enough input arguments.

Error in NewtonsMethodTemplate (line 20)
    Nx = size(x0,1);
```

Published with MATLAB® R2019a

```
%Task 2a) and 2b)
lambda = -2;
t_step = 0.2;
tf = 2;
T = 0:t_step:tf;

f_2b = @(t,x) lambda*x;
J = @(x) -2;

x_0 = 1;

x = ImplicitEulerTemplate(f_2b, J, T, x_0);

%Real Solution
Sol = x_0*exp(lambda*T);

figure(10)
plot(T, Sol, '-', T, x, '-');
legend('Actual Solution', 'Implicit Euler');
title('fig 4: Implicit Euler compared to Actual Solution');
```



Published with MATLAB® R2019a

```
function x = ImplicitEulerTemplate(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)
    N_x = size(x0,1);
    N_t = size(T,2);
    x = zeros(N_x,N_t);
    %
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    x_t = x0; % initial iteration
    x(:,1) = x_t;
    fn = f(T(1),x_t);
    % Loop over time points
    for n_t=2:N_t
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % 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(n_t) - T(n_t -1);
        x(:, n_t) = x(:, n_t - 1) + dt*fn;
        r = @(F) x(:,n_t-1) + dt * f(T(n_t), F)-F;
        J_ie = @(F) dt*dfdx(F) - eye(size(fn,1), N_x);
        [S_ie, infNorm_ie] = NewtonsMethodTemplate(r, J_ie, x(:,n_t));
        x(:,n_t) = S_ie(:,end);
        fn = f(T(n_t), x(:,n_t));
        %
        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
end
```

```
Not enough input arguments.

Error in ImplicitEulerTemplate (line 13)
    N_x = size(x0,1);
```

Published with MATLAB® R2019a