MAT 423: Solution homework 2 (09/05)

Ex 1. [2pts] See scripts at the end.

Ex 2. [4pts] + [extra 2pts] Let $f(x) = xe^x - 1$.

a) We use bisection method with $a = 0$ and $b = 1$. In order to have precision $\varepsilon = 10^{-16}$, a sufficient condition on the number of iteration n is given by: 1pt

$$\frac{1}{2^{n+1}} \cdot |b - a| < \varepsilon \quad \Rightarrow \quad -(n+1)\ln 2 < -16 \ln 10 \quad \Rightarrow \quad n > 53.$$

After $n = 53$ iterations, we obtain: 1pt

$$x_{ref} = 0.567143290409784.$$

b) Using the reference solution x_{ref}, we estimate the error $e_n = |x_n - x_{ref}|$ for the three methods at after each iterations. We use $x_0 = 1$ as initial point for the Newton method, $(x_0, x_1) = (0, .1)$ for the secant method. The errors are plotted in figure 1 with log scale in the y-axsis. 1pt 1pt

c^*) [extra +2pts] To further estimate the decay of each method, we need to estimate the paramaters C and α such that: $\frac{e_{n+1}}{e_n^\alpha} \approx C$ in other words $e_{n+1} \approx C e_n^\alpha$. To make this relationship linear, we take the logarithm:

$$\log e_{n+1} \approx \log C + \alpha \log e_n.$$

Therefore, denoting $y = \log e_{n+1}$ and $x = \log e_n$, we have:

$$y \approx \beta + \alpha\, x,$$

with $\beta = \log C$. We plot the data points $(x_i, y_i)_i$ for the three methods and estimate the linear-relationship in figure +1pt

We obtain the following constant: +1pt

- bisection method: $\alpha = .991 \approx 1$ and $C = .438 \approx \frac{1}{2}$.
- secant method: $\alpha = 1.62 \approx \frac{1+\sqrt{5}}{2}$ and $C = .926$.
- Newton method: $\alpha = 1.94 \approx 2$ and $C = .713$.

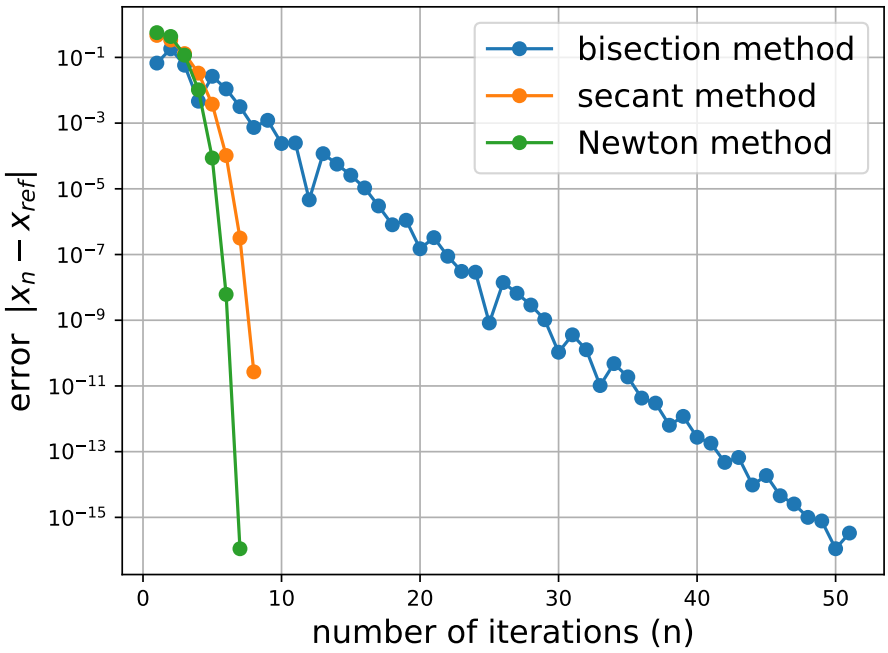

Figure 1: Evolution of the error $e_n = |x_n - x_{ref}|$ for the three methods. In log-scale, the convergence of the bisection is linear whereas secant and Newton methods are faster.

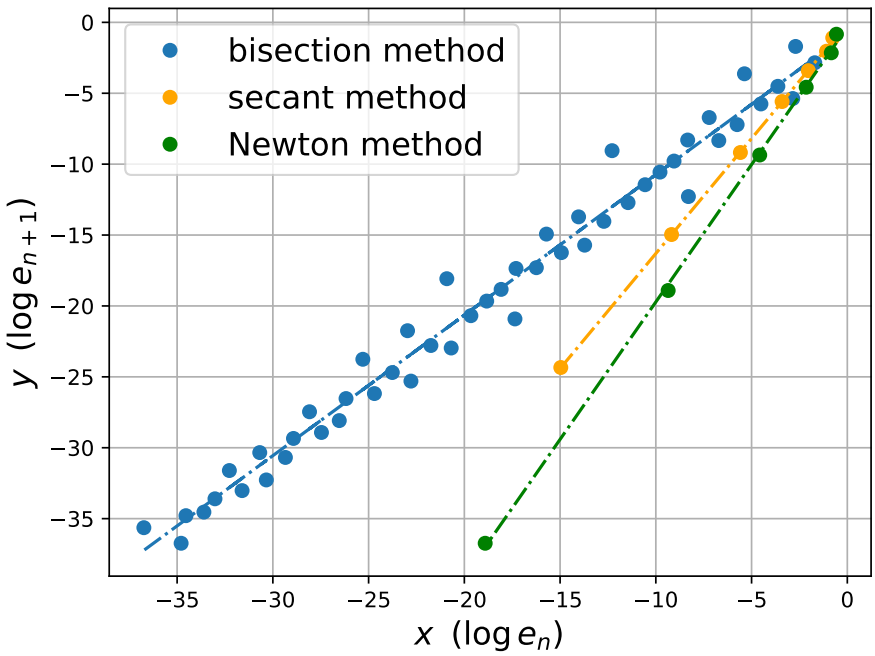

Figure 2: Estimation of the relationship $e_{n+1} \approx C \cdot e_n^{\alpha}$ using log-log plot.

Ex 3. 4pts

Let $\varphi(x) = \sqrt{x+1}$.

a) We show that φ is a contraction on $I = [0, \infty)$.

- For any $x \geq 0$, $\varphi(x) \geq 0$. Thus, $\varphi(I) \subset I$. .5pt
- $|\varphi'(x)| = \left|\frac{1}{2(x+1)}\right| \leq \frac{1}{2}$ on I.
  Thus, for any $x, y \in I$, we have: $|\varphi(x) - \varphi(y)| \leq \frac{1}{2}|x - y|$. .5+.5pt

Therefore, φ is a contraction on I. We deduce that there exists a unique fixed point x_* of φ on I. .5pt

b) Let $J = [1, 2]$. We have $\varphi(J) \subset J$, thus we can narrow our interval ($x_* \in J$).
Consider the sequence $\{x_n\}_n$ defined recursively by: $x_{n+1} = \varphi(x_n)$ with $x_0 = 1$. We have:

$$|x_n - x_*| \leq \frac{k^n}{1-k} \cdot |b - a| = 1 \cdot \left(\frac{1}{2}\right)^{n-1}.$$ 1pt

Thus, in order to have the error $e_n = |x_n - x_*|$ less than 10^{-4}, a sufficient condition is:

$$\frac{1}{2^{n-1}} \leq 10^{-4} \Rightarrow n \geq \frac{4 \ln 10}{\ln 2} + 1 \approx 14.3.$$ 1pt

After 15 iterations, the error is less than 10^{-4}.

Ex 4. Let $\varphi(x) = \sqrt{x^2+1}$ and $I = [0, \infty)$.

a) By the mean value theorem:

$$|\varphi(x) - \varphi(y)| = |\varphi'(c)(x - y)|,$$

with $c \in [x, y]$. Since $|\varphi'(s)| = \left|\frac{s}{\sqrt{s^2+1}}\right| < 1$, we deduce that: $|\varphi(x) - \varphi(y)| < |x - y|$.

b) We notice that $x_n = \sqrt{n}$. Thus, the sequence does not converge.

c) $\varphi(x) = x$ implies: $\sqrt{x^2+1} = x$ which is not possible. Thus, φ does not have a fixed point.
This does not contradict the *fixed-point theorem* since φ is **not** a contraction: it does not exist $k < 1$ such that $|\varphi(x) - \varphi(y)| \leq k|x - y|$.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      Newton/secant methods (MATLAB/OCTAVE)      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function saveX=Newton_method(f,df,x0,N)
%%  Newton Method in 1D:
%%    x_n+1 = x_n - f(x_n)/f'(x_n)
%%
    %% A) initialisation
    saveX = zeros(1,N+1);
    x = x0;
    saveX(1) = x0;
    %%--------------------------%%
    %% B)        Loop           %%
    %%--------------------------%%
    for i=1:N
        x = x - f(x)/df(x);       % update
        saveX(i+1) = x;           % save
    end
end

function saveX=secant_method(f,x0,x1,N)
%%  secant Method in 1D:
%%    x_n+1 = x_n - f(x_n)*(x_n-x_n-1)/(f(x_n)-f(x_n-1))
%%
    %% A) initialisation
    saveX = zeros(1,N+1);
    x_nm = x0;
    x_n = x1;
    saveX(1) = x1;                % does not keep x0
    %%--------------------------%%
    %% B)        Loop           %%
    %%--------------------------%%
    for i=1:N
        x_np = x_n - f(x_n)*(x_n-x_nm)/(f(x_n)-f(x_nm));
        saveX(i+1) = x_np;        % save
        x_nm = x_n;
        x_n = x_np;
    end
end
```

```
#####################################################
##         Newton/secant methods (Python)          ##
#####################################################
import numpy as np

def Newton_method(f,df,x0,N):
    '''  Newton Method in 1D:
    x_n+1 = x_n - f(x_n)/f'(x_n)
    '''
    ## A) initialisation
    saveX = np.zeros(N+1)
    x = x0
    saveX[0] = x0
    ##--------------------------##
    ## B)        Loop           ##
    ##--------------------------##
    for i in range(N):
        x = x - f(x)/df(x)      # update
        saveX[i+1] = x          # save
    ## C) output
    return saveX

def secant_method(f,x0,x1,N):
    '''  secant Method in 1D:
    x_n+1 = x_n - f(x_n)*(x_n-x_n-1)/(f(x_n)-f(x_n-1))
    '''
    ## A) initialisation
    saveX = np.zeros(N+1)
    x_nm,x_n = x0,x1
    saveX[0] = x1               # does not keep x0
    ##--------------------------##
    ## B)        Loop           ##
    ##--------------------------##
    for i in range(N):
        x_np = x_n - f(x_n)*(x_n-x_nm)/(f(x_n)-f(x_nm))
        saveX[i+1] = x_np       # save
        x_nm = x_n
        x_n = x_np
    ## C) output
    return saveX
```

```julia
#####################################################
##         Newton/secant methods (JULIA)           ##
#####################################################
function Newton_method(f,df,x0,N)
    #=  Newton Method in 1D:
    x_n+1 = x_n - f(x_n)/f'(x_n)
    =#
    ## A) initialisation
    saveX = zeros(N+1)
    x = x0
    saveX[1] = x0
    ##--------------------------##
    ## B)        Loop           ##
    ##--------------------------##
    for i=1:N
        x = x - f(x)/df(x)      # update
        saveX[i+1] = x          # save
    end
    ## C) output
    return saveX
end

function secant_method(f,x0,x1,N)
    #=  secant Method in 1D:
    x_n+1 = x_n - f(x_n)*(x_n-x_n-1)/(f(x_n)-f(x_n-1))
    =#
    ## A) initialisation
    saveX = zeros(N+1)
    x_nm,x_n = x0,x1
    saveX[1] = x1               # does not keep x0
    ##--------------------------##
    ## B)        Loop           ##
    ##--------------------------##
    for i=1:N
        x_np = x_n - f(x_n)*(x_n-x_nm)/(f(x_n)-f(x_nm)) # update (could be speed up...)
        saveX[i+1] = x_np       # save
        x_nm = x_n
        x_n = x_np
    end
    ## C) output
    return saveX
end
```