Numerical Analysis (089180) A.Y. 2022-2023
prof. Simona Perotto Luca Liverotti
simona.perotto@polimi.it luca.liverotti@polimi.it

Lab 3 – Solutions

October 7, 2022

1 Modified Newton Method

Method 3.1 (Newton method). *Newton method consists in approximating the solution of $f(x) = 0$ with the sequence*

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \quad \textit{with} \quad k \geq 0 \quad \textit{and} \quad f'(x^{(k)}) \neq 0$$

If the root ξ is not simple the Newton method converges with order one. If m is the multiplicity of the root, then the modification

$$x^{(k+1)} = x^{(k)} - m\frac{f(x^{(k)})}{f'(x^{(k)})} \quad \text{with} \quad k \geq 0 \quad \text{and} \quad f'(x^{(k)}) \neq 0$$

allow us to recover the second order of convergence.

Remark 3.1 (Exam tips!). Complete exercise 2.2 and homework 2.5. ■

2 Fixed point methods

Method 3.2 (Fixed point method). *A* ***fixed point method*** *consists in the sequence*

$$x^{(k+1)} = \phi(x^{(k)})$$

with ***consistency*** *and* ***convergence*** *properties, that means ϕ is s.t.*

$$\xi = \phi(\xi) \qquad \textit{and} \qquad \lim_{k\to\infty} \left| x^{(k)} - \xi \right| = 0.$$

Theorem 3.1 (Local convergence, Ostrowski theorem). *Let ξ be a fixed point of a function ϕ which is continuous and differentiable in a neighborhood of ξ. If $|\phi'(\xi)| < 1$, then there exists $\delta > 0$ s.t., for any $x^{(0)} \in (\xi - \delta, \xi + \delta)$, the sequence $x^{(k+1)} = \phi(x^{(k)})$ converges to ξ. Moreover, the following limit holds*

$$\lim_{k\to\infty} \frac{x^{(k+1)} - \xi}{x^{(k)} - \xi} = \phi'(\xi)$$

Remark 3.2. Let ξ be a fixed point of the a function ϕ which is continuous and differentiable in a neighborhood of ξ.

- if $|\phi'(\xi)| > 1$, then the sequence $x^{(k+1)} = \phi(x^{(k)})$ will not converge to ξ;
- if $|\phi'(\xi)| = 1$, then no general conclusion can be drawn both convergence and divergence become possibile. ■

Remark 3.3 (Geometrical interpretation). Solving the fixed point problem $x = \phi(x)$ is equivalent to solve the system

$$\begin{cases} y = x \\ y = \phi(x) \end{cases}$$

i.e. to determine the intersections between $\phi(x)$ and the bisector line $y = x$ of the first and the third quadrants. ■

Theorem 3.2. *If ϕ is $\mathcal{C}^p$ in a suitable neighborhood of ξ, and if*

$$\phi^{(i)}(\xi) = 0 \qquad i = 1, \dots, p-1, \quad \phi^{(p)}(\xi) \neq 0,$$

then the fixed-point method $x^{(k+1)} = \phi(x^{(k)})$ has order p, i.e.

$$\lim_{k\to\infty} \frac{x^{(k+1)} - \xi}{(x^{(k)} - \xi)^p} = \frac{\phi^{(p)}(\xi)}{p!}$$

Remark 3.4 (How to estimate the rate of convergence p). Consider the limit of the above-mentioned quantity for two consecutive steps and, for $k \to \infty$, analyze the corresponding error defining $e^k = x^k - \xi$:

$$\frac{e^{(k+1)}}{(e^{(k)})^p} = \frac{e^{(k)}}{(e^{(k-1)})^p},$$

so that

$$\frac{e^{(k+1)}}{e^{(k)}} = \left(\frac{e^{(k)}}{e^{(k-1)}}\right)^p,$$

and applying the logarithm

$$p = \frac{\log \frac{e^{(k+1)}}{e^{(k)}}}{\log \frac{e^{(k)}}{e^{(k-1)}}} = \frac{\log e^{(k+1)} - \log e^{(k)}}{\log e^{(k)} - \log e^{(k-1)}}.$$ ■

Exercise 3.1. Solve the equation $x^2 - 5 = 0$ to find the root $\xi = \sqrt{5}$ using the following iterative methods:

1. $x^{(k+1)} = 5 + x^{(k)} - (x^{(k)})^2$;
2. $x^{(k+1)} = \frac{5}{x^{(k)}}$;
3. $x^{(k+1)} = 1 + x^{(k)} - \frac{1}{5}(x^{(k)})^2$;
4. $x^{(k+1)} = \frac{1}{2}\left(x^{(k)} + \frac{5}{x^{(k)}}\right)$.

a. Are these iterations consistent and convergent? Motivate your answer.

b. Implement a function for the generic fixed point iteration with function ϕ.

c. Assess numerically the convergence of the proposed iterative methods.

Solution Exercise 3.1.

```
function [xi, x_iter] = fixed_point(phi, x0, tol, maxit)
  %FIXED_POINT Find a root of the equation f(x) = 0 using the fixed point
  %  method x = phi(x), starting from the initial guess x0.
  %
  %  [xi, x_iter] = FIXED_POINT(phi, x0, tol, maxit)
  %
  %  Inputs : phi  = function handle to the iteration function
  %           x0   = initial guess
  %           tol  = requested tolerance
  %           maxit = maximum number of iterations
  %  Output :
  %       xi = approximation of the root
```

```
  %       x_iter  = vector of the approximations of the root at each step

  x_iter(1) = x0;

  for (iter = 1:maxit)

    x_iter(iter+1) = phi(x_iter(iter));

    if (abs (x_iter(iter+1) - x_iter(iter)) < tol)
      break;
    end

  end

  xi = x_iter(end);

end
function fixed_point_plot(phi, x0, tol, maxit)
  %FIXED_POINT_PLOT Plot the function f and the evolution of the fixed point method.
  %
  %  FIXED_POINT_PLOT(phi, x0, tol, maxit)
  %
  %  Inputs : phi  = function handle to the iteration function
  %           x0    = initial guess
  %           tol   = requested tolerance
  %           maxit = maximum number of iterations

  [xi, x] = fixed_point(phi, x0, tol, maxit);
  a = min(x);
  b = max(x);

  figure
  hold on, box on
  x_plot = linspace(a, b, 1000);
  plot(x_plot, phi(x_plot), 'LineWidth', 2)
  plot(x_plot, x_plot, 'k-', 'LineWidth', 1)
  xlabel('x','FontSize', 16)
  ylabel('phi(x)','FontSize', 16)
  set(gca,'FontSize', 16)
  set(gca,'LineWidth', 1.5)
  %pause

  old_color = 'kx-';
  new_color = 'gx-';
 for iter = 1:length(x)-1
    if (iter > 1)
        plot([x(iter-1) x(iter-1) x(iter)], [x(iter-1) x(iter) x(iter)], old_color, 'LineWidth',
         2, 'MarkerSize', 8)
    plot([x(iter) x(iter) x(iter+1)], [x(iter) x(iter+1) x(iter+1)], new_color, 'LineWidth', 2,
     'MarkerSize', 8)
    %pause
    end
 end
 pause
end

% EXERCISE 3.1
clear all
close all
clc

xi = sqrt(5);

% The substitution x{(k+1)} = x{(k)} = xi = sqrt{5} easily shows that all the methods are
consistent. The evaluation of the convergence properties of the iterative methods require the
computation of the first derivative of phi at x = xi.

% Method 1.

phi1 = @(x) 5 + x - x.^2;
```

```
dphi1 = @(x) 1 - 2*x;
abs(dphi1(xi))

% The absolute value of the derivative of phi at xi
% is greater than 1: the method will not converge.

% Method 2.

phi2 = @(x) 5./x;
dphi2 = @(x) -5./x.^2;
abs(dphi2(xi))

% The absolute value of the derivative of phi at xi
% is equal to 1: no theoretical conclusion can be stated in this case.

% Method 3.

phi3 = @(x) 1 + x - 1/5*x.^2;
dphi3 = @(x) 1 - 2/5*x;
abs(dphi3(xi))

% The absolute value of the derivative of phi at xi
% is less than 1: the method will converge provided that the initial guess x{(0)} is close
enough to xi (local convergence).

% Method 4.

phi4 = @(x) 1/2*(x + 5./x);
dphi4 = @(x) 1/2 - 5./(2*x.^2);
abs(dphi4(xi))

% The absolute value of the derivative of phi at xi
% is zero (i.e. less than 1): the method will converge provided that the initial guess x{(0)}
is close enough to xi (local convergence).

d2phi4 = @(x) 5./x.^3;
abs(d2phi4(xi))

% The second derivative is different from zero,
% so method 4 is expected to be of second order.

%%
pause
tol = 1e-6;
maxit = 1000;

% Method 1.
clc
x0 = xi + 0.001;
[xi1, x1] = fixed_point(phi1, x0, tol, maxit);
xi1
iter1 = numel(x1) - 1
[xi1, x1] = fixed_point_FV(phi1, x0, tol, maxit);
xi1
iter1 = numel(x1) - 1
% The approximation xi is incorrect and the number of performed iterations is the maximum: as
expected, the method did not converge.

%fixed_point_plot(phi1, x0, tol, 5);
pause

% Method 2.

x0 = 3;
[xi2, x2] = fixed_point(phi2, x0, tol, maxit);
xi2
iter2 = numel(x2) - 1
[xi2, x2] = fixed_point_FV(phi2, x0, tol, maxit);
xi2
iter2 = numel(x2) - 1
```

```
% The approximation xi is incorrect and the number of performed iterations is the maximum: the
 method did not converge.

%fixed_point_plot(phi2, x0, tol, 5);
pause

% Method 3.
x0 = 4;
[xi3, x3] = fixed_point(phi3, x0, tol, maxit);
xi3
iter3 = numel(x3) - 1
[xi3, x3] = fixed_point_FV(phi3, x0, tol, maxit);
xi3
iter3 = numel(x3) - 1

% The method converged to xi.

%fixed_point_plot(phi3, x0, tol, maxit);
pause

% But the convergence is only local...

x0 = 10;
[xi3, x3] = fixed_point(phi3, x0, tol, maxit);
xi3
iter3 = numel(x3) - 1
[xi3, x3] = fixed_point_FV(phi3, x0, tol, maxit);
xi3
iter3 = numel(x3) - 1
% With a different initial guess, the method may not converge to xi.

%fixed_point_plot(phi3, x0, tol, maxit);
pause

% Method 4.
x0 = 4;
[xi4, x4] = fixed_point(phi4, x0, tol, maxit);
xi4
iter4 = numel(x4) - 1
[xi4, x4] = fixed_point_FV(phi4, x0, tol, maxit);
xi4
iter4 = numel(x4) - 1

% The method converged to xi.

%fixed_point_plot(phi4, x0, tol, maxit);
```

3 Bisection - Newton methods

Theorem 3.3. *If $f \in \mathcal{C}^2([a,b])$ and $f'(x) \neq 0$ in an open interval containing ξ, then $\exists \delta > 0$* ***s.t.*** *$\forall x^{(0)} : \left|x^{(0)} - \xi\right| < \delta$ the Newton method converges quadratically to ξ.*

The convergence is guaranteed only if the initial guess $x^{(0)}$ is close enough to the root ξ, and for this reason the Newton method is a **locally** convergent method. A simple solution to overcome this issue consists in employing the bisection method to predict the inital guess $x^{(0)}$, as shown in the next exercise.

Exercise 3.2. Consider the following function in the interval $[-1, 6]$

$$f(x) = \arctan\left[7\left(x - \frac{\pi}{2}\right)\right] + \sin\left[\left(x - \frac{\pi}{2}\right)^3\right].$$

a. Plot f in order to find an interval containing a root. What is the multiplicity of the root?

b. Use the Newton method to find the root with a tolerance of 10^{-10} and initial guess $x^{(0)} = 1.5$. Compute the error.

c. Use the Newton method to find the root with a tolerance of 10^{-10} and initial guess $x^{(0)} = 4$. Compute the error.

d. If possible, apply the bisection method on the interval $[a, b] = [-1, 6]$ and tolerance $\frac{b-a}{2^{30}}$. Compute the error.

e. Write a function `bisection_newton.m` to find ξ using the Newton method starting from an initial guess obtained after few iterations of a bisection method. Test with $[a, b] = [-1, 6]$, 5 iterations of the bisection method and tolerance 10^{-10} for the Newton method.

Solution Exercise 3.2.

```
function [xi, x_iter_bisection, x_iter_newton] = bisection_newton(f, df, a, b, tol_bisection,
tol_newton, maxit_newton, multiplicity)
  %NEWTON Find a root of the equation f(x) = 0 using the Newton method, starting from an
  initial guess obtained by few iterations of a bisection method.
  %
  %  [xi, x_iter] = BISECTION_NEWTON(f, df, a, b, tol_bisection, tol_newton, maxit_newton,
  multiplicity)
  %
  %  Inputs : f  = function handle to the function f(x)
  %          df   = function handle to the derivative of the function f(x)
  %          a    = left bound
  %          b    = right bound
  %          tol_bisection = requested tolerance for the bisection method
  %          tol_newton = requested tolerance for the Newton method
  %          maxit_newton = maximum number of iterations for the Newton method
  %          multiplicity = multiplicity of the root
  %  Output :
  %      xi = approximation of the root
  %      x_iter  = vector of the approximations of the root at each step

  if (nargin < 8)
   multiplicity = 1;
  end

  [xi_bisection, x_iter_bisection] = bisection(f, a, b, tol_bisection);
  [xi_newton, x_iter_newton] = newton(f, df, xi_bisection, tol_newton, maxit_newton,
  multiplicity);

  xi = xi_newton;

end

% EXERCISE 3.2

a = -1;
b = 6;
f = @(x) atan(7*( x - pi/2)) + sin((x-pi/2).^3);
rootfinding_function_plot(f, a, b, true);

% The function is null at xi = {pi}/{2}.

xi_ex = pi/2;
df = @(x) 7 ./ ( 1 + 49 * ( x-pi/2 ).^2 ) + 3 * (x-pi/2).^2 .* cos( (x-pi/2).^3 );
df(xi_ex);
% The multiplicity of the root is one since the derivative of f at xi = {pi}/{2} is different
from zero.

x0 = 1.5;
tol = 1e-10;
maxit = 1000;
[xi1, x_iter1] = newton(f, df, x0, tol, maxit);
xi1
iter1 = numel(x_iter1) - 1
err1 = abs( xi1 - xi_ex)
% Newton method converges to xi within the prescribed tolerance in very few iterations.

x0 = 4;
```

```
tol = 1e-10;
maxit = 1000;
[xi2, x_iter2] = newton(f, df, x0, tol, maxit);
xi2
iter2 = numel(x_iter2) - 1
err2 = abs( xi2 - xi_ex)
% Newton method does not converge to xi, and stops because of the maximum number of iterations
 has been reached. In this case the initial guess is not close enough to the root xi and the
local convergence result does not apply.

tol = (b-a)/(2^30);
[xi3, x_iter3] = bisection(f, a, b, tol);
xi3
iter3=numel(x_iter3)

%%

tol_bisection = (b-a)/(2^5);
tol_newton = 1e-10;
maxit_newton = 1000;
[xi4, x_iter4_bisection, x_iter4_newton] = bisection_newton(f, df, a, b, tol_bisection,
tol_newton, maxit_newton);
xi4
iter4_bisection=numel(x_iter4_bisection)
iter4_newton=numel(x_iter4_newton)
err4 = abs( xi4 - xi_ex)
```