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

Lab 3 – Homework Solutions

October 7, 2022

A Homework

Homework 3.1 (Fixed point method). a. Let $\phi(x) = x - x^3$, which admits $\xi = 0$ as a fixed point. Compute $\phi'(\xi)$ and test the convergence of the sequence $x^{(k+1)} = \phi(x^{(k)})$ for $x^{(0)} \in [-1, 1]$.

b. Let $\phi(x) = x + x^3$, which admits $\xi = 0$ as a fixed point. Compute $\phi'(\xi)$ and test the convergence of the sequence $x^{(k+1)} = \phi(x^{(k)})$ for $x^{(0)} \in [-1, 1]$.

Solution Homework 3.1.

```
% HOMEWORK 3.1

xi = 0
% In both cases phi'(xi) = 1: this example confirms that in the case abs{phi'(xi)} = 1 no
general statement about convergence/non-convergence can be provided.

phi = @(x) x - x.^3;
tol = 1e-6;
maxit = 1000;
x0 = -1 + 2*rand(1); % random initial value in [-1,1]
[xi, x] = fixed_point(phi, x0, tol, maxit);
xi
iter = numel(x) - 1

% The fixed point method converges to xi, very slowly.

phi = @(x) x + x.^3;
tol = 1e-6;
maxit = 1000;
x0 = -1 + 2*rand(1); % random initial value in [-1,1]
[xi, x] = fixed_point(phi, x0, tol, maxit);
xi
iter = numel(x) - 1

% The fixed point method diverges.
```

Homework 3.2 (Fixed point method, Newton method). Given the equation

$$f(x) = e^{x^2} \log(x + 1) = 1$$

consider the following three iterative methods

1. $x^{(k+1)} = \sqrt{-\log\log(x^{(k)} + 1)}$;
2. $x^{(k+1)} = x^{(k)}\, e^{(x^{(k)})^2} \log(x^{(k)} + 1)$;
3. Newton method.

Verify the consistency of the schemes and apply them to compute the root ξ with a tolerance equal to 10^{-3}. Comment on the results.

Solution Homework 3.2.

```
% Rewrite f(x) = 1 as F(x) = f(x) - 1 = 0. Bolzano's theorem grants the existence of at least
a root xi because F is continuous in its domain (-1, +infinity), lim{x -> -1+} F(x) = -
infinity and lim{x -> +infinity} F(x) = +infinity.

F = @(x) exp(x.^2).*log(x + 1) - 1;
rootfinding_function_plot(F, -1, 2);

% The plot shows that there exists a root in the interval (0.5,1).

% Method 1.
% The method is defined only if x{(k)} in (0, e-1] for all k, as can be seen with an explicit
calculation of the domain of phi_1(x) = sqrt{-log log(x+1)}.
% The method is consitent. Indeed, ξ ∈ (0, e − 1] and
%
% xi = sqrt{-log log(xi+1)} -> - xi^2 = log log(xi+1) -> e^{- xi^2} = log(xi+1) -> e^{xi^2}
log(xi+1) = 1
%
% which is an identity.

% First method: check convergence condition.
phi1 = @(x) sqrt(-log(log(x + 1)));
dphi1 = @(x) -1./(2*log(x + 1).*(-log(log(x + 1))).^(1/2).*(x + 1));
rootfinding_function_plot(dphi1, 0.5, 1, true);
% The plot shows that the method converges when x{0} in [0.5, 1].

[xi1, x1] = fixed_point(phi1, 0.9, 1e-3, 1000);
xi1
iter1 = numel(x1)

% Method 2.
% The method is defined only if x{(k)} > - 1 for all k, as can be seen with an explicit
calculation of the domain of phi_2(x) = x e^{x^2}log(x+1)
% The method is consistent. Indeed, xi > -1 and
%
% xi = xi e^{xi^2}log(xi+1) -> e^{xi^2} log(xi+1) = 1
%
% which is an identity.
%
% Second method: check convergence condition.
phi2 = @(x) x.*exp(x.^2).*log(x + 1);
rootfinding_function_plot(phi2, -1, 1, true);
rootfinding_function_plot(@(x) x, -1, 1, true);

% The comparison with the slope of the bisector shows that, in a neighborhood of the root xi,
the absolute
% value of the derivative is greater than 1. For this reason the method will not converge.
% Notice that another fixed point appeared. This is due to the fact that we
% multiplied times x both sides of the equation, which of course vanishes
% for x=0.


[xi2a, x2a] = fixed_point(phi2, 0.75, 1e-3, 100);
xi2a
iter2a = numel(x2a)
[xi2b, x2b] = fixed_point(phi2, 0.76, 1e-3, 100);
xi2b
iter2b = numel(x2b)

% The fixed point method either converges to the new root x = 0 or diverges.

% Method 3.

% Third method: check convergence condition.
rootfinding_function_plot(F, -1, 2, true);
% The plot of F shows that F'(x) is always positive in [-1, 2], therefore the method is
locally convergent.

dF = @(x) 2*x*exp(x.^2).*log(x+1) + exp(x.^2)./(x+1);
```

```
[xi3, x3] = newton(F, dF, 1.4, 1e-3, 1000);

% Rate of convergence
[xiex, xex] = newton(F, dF, 1.4, 1e-12, 1000);
err1 = abs(x1 - xiex);
err3 = abs(x3 - xiex);

% Approximate the convergence order
p1 = diff( log(err1(2:end) ) ) ./ diff( log(err1(1:end-1) ) )
p3 = diff( log(err3(2:end) ) ) ./ diff( log(err3(1:end-1) ) )

% The order of the first method is 1, Newton method is as expected of second order.
figure
semilogy(err1, 'bs-','LineWidth',2)
hold on, box on
semilogy(err3, 'rs-','LineWidth',2)
set(gca,'LineWidth',1.5)
set(gca,'FontSize',16)
xlim([0 numel(err1)+1])
xlabel('iterations','FontSize',16)
ylabel('error','FontSize',16)
h = legend('Method 1','Newton');
set(h,'FontSize',16)
```