EPFL

Numerical Analysis and Computational Mathematics

Fall Semester 2019 - Section CSE

Dr. Rafael Vázquez Hernández

Assistant: Ondine Chanon

Session 3 - 2 October 2019

Solutions - Nonlinear equations: Newton method

Solution I (MATLAB)

We consider the following implementation of the MATLAB function `newton.m`:

```
function [xvect, resvect, nit] = newton( fun, dfun, x0, tol, nmax )
% NEWTON Find a zero of a nonlinear scalar function.
%   [XVECT] = NEWTON(FUN,DFUN,X0,TOL,NMAX) finds a zero of the differentiable
%   function FUN using the Newton method and returns a vector XVECT containing
%   the successive approximations of the zero (iterates). DFUN is the derivative of FUN.
%   FUN and DFUN accept real scalar input x and return a real scalar value;
%   FUN and DFUN can also be inline objects. X0 is the initial guess.
%   TOL is the tolerance on error allowed and NMAX the maximum number of iterations.
%   The stopping criterion based on the difference of successive iterates is used.
%   If the search fails a warning message is displayed.
%
%   [XVECT,RESVECT,NIT] = NEWTON(FUN,DFUN,X0,TOL,NMAX) also returns the vector
%   RESVECT of residual evaluations for each iterate, and NIT the number of iterations.
%   Note: the length of the vectors is equal to ( NIT + 1 ).

nit = 0;
xvect(nit+1) = x0;
resvect(nit+1) = fun( x0 );
err_estim = tol + 1;
while ( err_estim >= tol && nit < nmax )
    xvect(nit+2) = xvect(nit+1) - fun( xvect(nit+1) ) / dfun( xvect(nit+1) );
    resvect(nit+2) = fun( xvect(nit+2) );
    err_estim = abs( xvect(nit+2) - xvect(nit+1) ); % diff. successive iterates
    nit = nit + 1;
end

if err_estim >= tol
    warning(['Newton method stopped without converging to the desired tolerance, '...
             'the maximum number of iterations was reached.']);
end

return
```

Notice that in this implementation of the function `newton`, the output variable `xvect` stores all the iterates of the method starting from $x^{(0)}$ with the approximate zero in the last entry of `xvect`; similarly for the residuals `resvect`. The length of the vectors `xvect` and `resvect` is `nit+1`.

a) We consider the following MATLAB commands:

```
fun = @(x) sin(2*x) + x;
dfun = @(x) 2*cos(2*x) + 1;
x0 = 0.7; tol = 1e-5; nmax = 50;
[xvect, resvect, nit] = newton( fun, dfun, x0, tol, nmax );
alpha = 0;
err = abs( xvect(end) - alpha )
nit
% err =
%
%      3.956702654861650e-19
%
% nit =
%
%      5
```

We obtain $n_c = 5$ and the error $e^{(n_c)} = |x^{(n_c)} - \alpha| = 3.9567 \cdot 10^{-19}$ is negligible. Instead, by setting $tol = 10^{-2}$, we obtain $n_c = 4$ and $e^{(n_c)} = 7.6357 \cdot 10^{-7}$.

b) We verify that $f(x) \in C^2(I_\alpha)$ in a neighborhood I_α of α (in fact, $f(x) \in C^\infty(\mathbb{R})$) and $f'(\alpha) = f'(0) = 3 \neq 0$ for which the zero $\alpha = 0$ is simple. Therefore, the convergence order 2 is expected by the theory, if $x^{(0)}$ is chosen sufficiently close to α; specifically, we have:

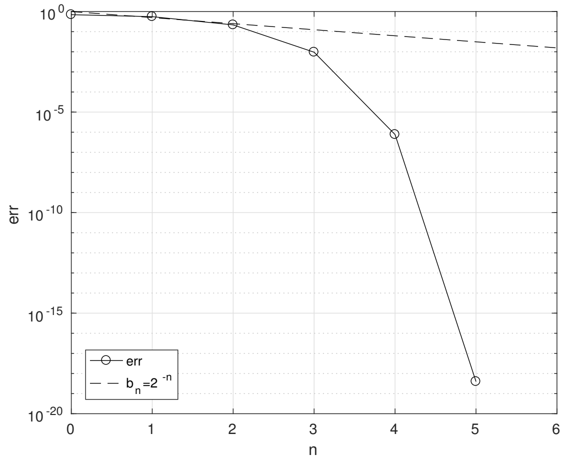
$$\lim_{n\to\infty} \frac{x^{(n+1)} - \alpha}{(x^{(n)} - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)} = 0,$$

with the asymptotic convergence factor $\frac{f''(\alpha)}{2f'(\alpha)} = 0$ since $f''(\alpha) = f''(0) = 0$.

c) The convergence order can be checked from the semi-logarithmic plot of the error $e^{(n)} = |x^{(n)} - \alpha|$ vs. n with the following commands:

```
x0 = 0.7; tol = 1e-12; nmax = 6;
[xvect, resvect, nit] = newton( fun, dfun, x0, tol, nmax );
errvect = abs( xvect - alpha );
nvect = 0 : nmax;
bnvect = 2.^( - nvect );
semilogy( nvect, errvect, '-ko', nvect, bnvect, '--k' ); grid on;
xlabel( 'n'); ylabel( 'err');
legend( 'err', 'b_n=2^{-n}', 'Location', 'southWest' );
```

If the convergence were of order 1, the error would decay as a straight line, similar to the one representing b_n. In this case much faster convergence is observed and so we verify that the Newton method indeed converges at least with order 2.



d) To implement the stopping criterion based on the absolute residual in the function `newton` it suffices to change the following line:

```
err_estim = tol + 1;
```

with:

```
err_estim = abs( resvect(nit+1) );
```

and then, the following line:

```
err_estim = abs( xvect(nit+2) - xvect(nit+1) ); % diff. successive iterates
```

with:

```
err_estim = abs( resvect(nit+2) ); % absolute residual
```

The Newton method with the stopping criterion based on the absolute residual is implemented in the MATLAB function `newton_residual` (see the file `newton_residual.m`).

We use the function `newton_residual` to obtain, e.g. for $\beta = 1$:

```
beta = 1e0;
fun = @(x) exp(beta * x) - 1;
dfun = @(x) beta * exp(beta * x);
x0 = 0.1; tol = 1e-7; nmax = 150;
[xvect, resvect, nit] = newton_residual( fun, dfun, x0, tol, nmax );
err = abs( xvect(end) - alpha )
res = abs( resvect(end) )
nit
% err =
%
%      6.822793885920731e-11
```

```
%
% res =
%
%      6.822786779991930e-11
%
% nit =
%
%      3
```

For $\beta = 1$, we obtain convergence to $\alpha = 0$ in $n_c = 3$ iterations, with the error $e^{(n_c)} = 6.8228 \cdot 10^{-11}$ and the residual $|r^{(n_c)}| = 6.8228 \cdot 10^{-11}$. For $\beta = 10^{-3}$, we obtain $n_c = 1$, $e^{(n_c)} = 4.9998 \cdot 10^{-6}$ and $|r^{(n_c)}| = 4.9998 \cdot 10^{-9}$. For $\beta = 10^{3}$, we obtain $n_c = 104$, $e^{(n_c)} = 1.2192 \cdot 10^{-13}$ and $|r^{(n_c)}| = 1.2192 \cdot 10^{-10}$.

Since $|f'(x)| \simeq \beta$ for x in a neighborhood of $\alpha = 0$, the stopping criterion based on the residual is satisfactory only for $\beta = 1$; when $\beta = 10^3$ and $\beta = 10^{-3}$, we have that $|f'(x)| \gg 1$ and $|f'(x)| \ll 1$, respectively, and the criterion is not satisfactory. Indeed, as we verified with MATLAB, the error $e^{(n_c)}$ is overestimated by the residual $r^{(n_c)}$ for $\beta = 10^3$ and more iterations than necessary are required by using this stopping criterion. On the contrary, for $\beta = 10^{-3}$, the error $e^{(n_c)}$ is underestimated by the residual $r^{(n_c)}$, for which the error is larger than predicted by the stopping criterion on the residual. We notice that $e^{(n_c)} \simeq \tilde{e}^{(n_c)} = |r^{(n_c)}|/|f'(\alpha)| = |r^{(n_c)}|/\beta$ for a zero of multiplicity one as in this case, where $\tilde{e}^{(n_c)}$ represents the error estimator at the converged iteration.

Solution II (MATLAB)

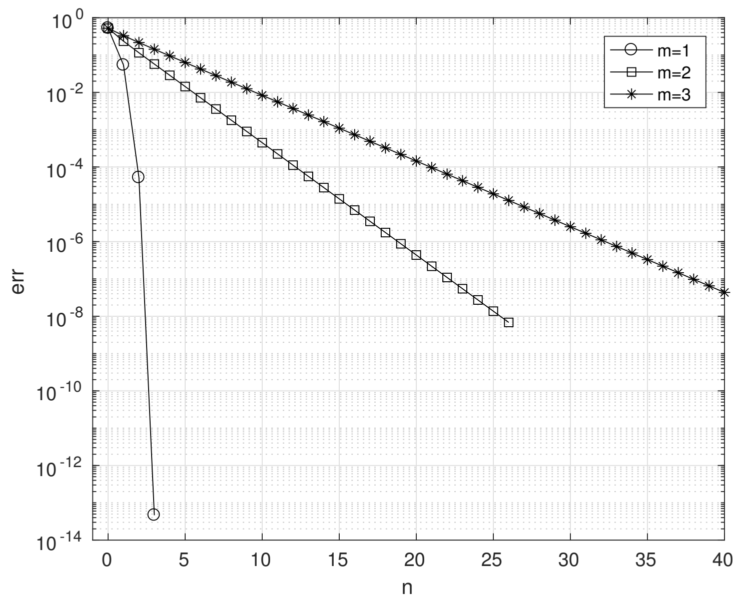
a) The only zero in the interval $(-\pi/2, \pi/2)$ is $\alpha = 0$. We can verify that $f(x) \in C^\infty(\mathbb{R})$ and the first $m-1$ derivatives of $f(x)$ in α are all zero, $f(0) = f'(0) = \ldots = f^{m-1}(0) = 0$, but $f^m(0) = m! \neq 0$. Therefore, $\alpha = 0$ is a zero of multiplicity m. This means that if $m \geq 2$, the Newton method converges only with order 1 (linear convergence), provided that the initial guess $x^{(0)}$ is sufficiently close to α. The order of convergence 2 (quadratic convergence) is expected only for the case $m = 1$ for which the zero α is simple ($f'(\alpha) = 1 \neq 0$ for $m = 1$).

b) By repeating the MATLAB commands reported in Exercise 1), point a) for the problem under consideration, we obtain that the number of iterations required for the convergence to α with the prescribed tolerance is $n_c = 4$ for the case $m = 1$, $n_c = 26$ for the case $m = 2$, and $n_c = 42$ for the case $m = 3$.

c) Similarly to Exercise 1, point c), we see that only in the case $m = 1$ (α is simple) we can observe convergence of order 2 as depicted in the following figure. The other two cases ($m = 2$ and $m = 3$) exhibit order of convergence 1 (linear convergence) as discussed in point a), due to the multiplicity $m \geq 2$ of the zero α. We use the following MATLAB commands:

```
x0 = pi/6; tol = 1e-8; nmax = 50;
alpha = 0;
errvect = {}; nvect={};
for m = 1:3
    fun = @(x) ( sin(x) ).^m;
    dfun = @(x) m * ( sin(x) ).^( m-1 ) .* cos(x);
    [xvect, resvect, nit] = newton( fun, dfun, x0, tol, nmax );
```

```
    errvect{ m } = abs( xvect - alpha );
    nvect{ m } = [0:nit];
end
semilogy( nvect{1}, errvect{1}, '-ko', nvect{2}, errvect{2}, '-ks', ...
    nvect{3}, errvect{3}, '-k*' ); grid on; axis( [-1 40 1e-14 1])
xlabel( 'n'); ylabel( 'err'); legend( 'm=1', 'm=2', 'm=3' );
```



d) In order to recover quadratic convergence (convergence order 2), the iterate of the Newton method for a zero of multiplicity $m \geq 1$ should be modified as:

$$x^{(n+1)} = x^{(n)} - m\frac{f(x^{(n)})}{f'(x^{(n)})} \quad \text{for all } n \geq 0.$$

The function `newton_modified` (see the file `newton_modified.m`) can be obtained from the function `newton` by changing the line:

```
function [xvect, resvect, nit] = newton( f, df, x0, tol, nmax )
```

with:

```
function [xvect, resvect, nit] = newton_modified( fun, dfun, x0, tol, nmax, m )
```

and then, the following line:

```
xvect(nit+2) = xvect(nit+1) - fun( xvect(nit+1) ) / dfun( xvect(nit+1) );
```

with:

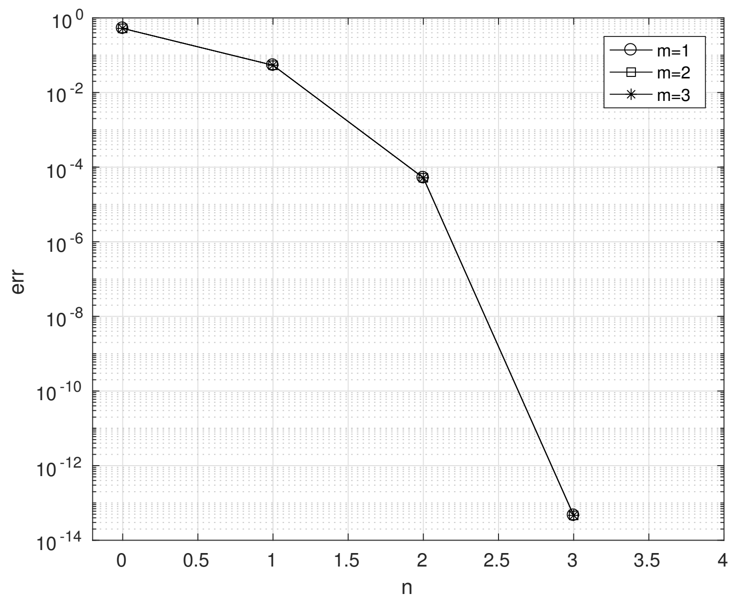
```
xvect(nit+2) = xvect(nit+1) - m * fun( xvect(nit+1) ) / dfun( xvect(nit+1) );
```

where m represents the multiplicity of the zero α.

e) The generic iterate of the modified Newton method for $f(x)$ reads:

$$x^{(n+1)} = x^{(n)} - m\frac{\sin(x^{(n)})^m}{m\sin(x^{(n)})^{m-1}\cos(x^{(n)})} = x^{(n)} - \frac{\sin(x^{(n)})}{\cos(x^{(n)})} \quad \text{for all } n \geq 0 \text{ and } m = 1, 2, \ldots$$

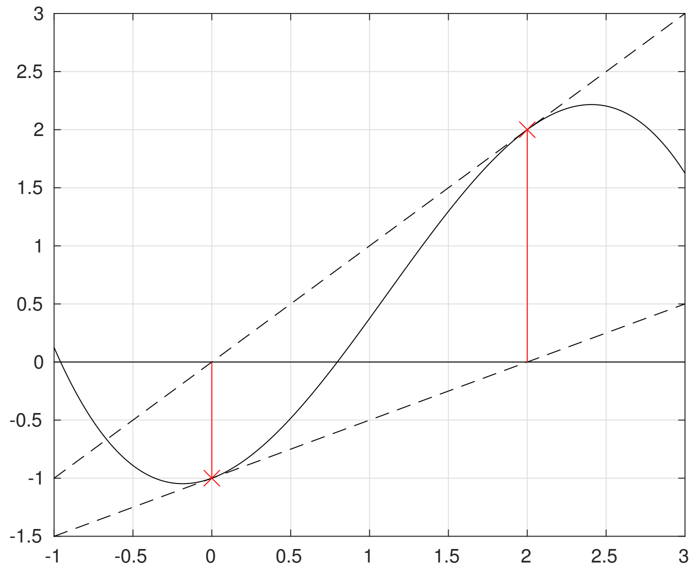
so that the iterations produce identical results in all three cases and the convergence is of order 2 as depicted in the following figure (use MATLAB commands similarly to point c)).



Solution III (MATLAB)

a) The plots of the polynomial and the tangent lines are obtained with the following commands:

```
p = @(x) -3/8 * x.^3 + 5/4 * x.^2 + x/2 - 1;
dp = @(x) -9/8 * x.^2 + 5/2 * x + 1/2;
t_0 = @(x) dp( 0 ) * x + p( 0 ); % tangent line in x=0
t_1 = @(x) dp( 2 ) * (x-2) + p( 2 ); % tangent in x=2
xv = linspace(-1,3,1001);
plot( xv, p( xv ), '-k', xv, t_0( xv ), '--k', xv, t_1( xv ), '--k');
```



b) The method converges to the zero $\alpha \simeq 0.7955$ in 8 iterations for the prescribed tolerance:

```
x0 = 1e-3; tol = 1e-8; nmax = 20;
[xvect, resvect, nit] = newton( p, dp, x0, tol, nmax );
xvect( nit+1 )
nit
% ans =
%
%    0.7955
%
% nit =
%
%      8
```

c) We use similar commands to point b). For $x^{(0)} = 0$ the Newton method reaches the maximum number of iterations without converging with the final iterate value equal to 0. Indeed, by construction and as highlighted graphically at point a), the Newton method produces the iterations:

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} = 0 - \frac{-1}{\frac{1}{2}} = 2,$$

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})} = 2 - 2 = 0,$$

for $x^{(0)} = 0$, so that $x^{(n)} = x^{(n+2)}$ for all $n = 0, 1, 2, \ldots$; i.e. the method starts to cycle and never converges as we verify by observing the iterates stored in the variable `xvect`:

```
% xvect =
%
%        0      2      0      2      0      2      0      2      0      2      0   ...
```

By setting $x^{(0)} = -10^{-3}$ we can recover convergence in 12 iterations, but the zero $\alpha_2 \simeq 3.4965$ does not belong to the interval $(0, 2)$, indeed the polynomial $p(x)$ possesses more than one zero.

The first few iterations are:

$$\left\{x^{(n)}\right\}_{n=1}^{\infty} = \{-0.001, 2.01001, -0.0414, 2.5399, 7.9100, 5.8262, \dots\}.$$

We conclude that, depending on the properties of the function $f(x)$, including the number of zeros, very small changes to the initial value $x^{(0)}$ can cause the Newton method to converge to completely different zeros, or, eventually to not converge to any of these. This fact stresses the importance of selecting an initial guess $x^{(0)}$ sufficiently close to the zero α.

Solution IV (MATLAB)

We consider the following implementation of the function `newtonsys`:

```
function [x, res, nit] = newtonsys( F, J, x0, tol, nmax )
% NEWTONSYS Find the zeros of a system of nonlinear equations.
%   [X] = NEWTONSYS(F,J,X0,TOL,NMAX) find the zero X of the
%   continuous and differentiable system of functions F nearest to X0 using the
%   Newton method. J is a function which takes X and returns the Jacobian matrix.
%   X0 is a column vector; F returns a column vector and J a square matrix.
%   The stopping criterion is based on the difference (norm) of successive
%   iterates.
%   If the search fails a warning message is displayed.
%
%   [X,RES,NITER] = NEWTONSYS(F,J,X0,TOL,NMAX) returns the value of the
%   residual RES in X and the number of iterations NITER required for computing X.
%   Note: only the final iterate is stored in X; similarly for RES.
%

nit = 0;
x = x0;
res = F( x0 );
err_estim = tol + 1;
while ( err_estim >= tol && nit < nmax )
    xold = x;
    x = xold - J( xold ) \ F( xold );
    res = F( x );
    err_estim = norm( x - xold ); % diff. successive iterates
    nit = nit + 1;
end

if err_estim >= tol
    warning(['Newton method stopped without converging to the desired tolerance, '...
             'the maximum number of iterations was reached.']);
end

return
```

We choose as initial datum $\mathbf{x}^{(0)} = (1.5, -2)^T$ for which the method converges in $n_c = 25$ iterations with an error $e^{(n_c)} = \left\|\mathbf{x}^{(n_c)} - \boldsymbol{\alpha}\right\|_2 = 6.5108 \cdot 10^{-6}$.

```
F = @(x) [ exp( x(1).^2 + x(2).^2 ) - 1; exp( x(1).^2 - x(2).^2 ) - 1 ];
J = @(x) [ 2 * x(1) * exp( x(1).^2 + x(2).^2 ), 2 * x(2) * exp( x(1).^2 + x(2).^2 );
2 * x(1) * exp( x(1).^2 - x(2).^2 ), -2 * x(2) * exp( x(1).^2 - x(2).^2 ) ];
alpha = [ 0; 0 ];
x0 = [ 1.5; -2 ]; tol = 1e-5;
nmax = 100;
[x, res, nit] = newtonsys( F, J, x0, tol, nmax );
err = norm( x - alpha )
nit
% err =
%
%      6.5108e-06
%
% nit =
%
%     25
```

By choosing $\mathbf{x}^{(0)} = (4, 4)^T$ we need $n_c = 50$ iterations to converge to the prescribed tolerance with the error $e^{(n_c)} = \left\|\mathbf{x}^{(n_c)} - \boldsymbol{\alpha}\right\|_2 = 7.4693 \cdot 10^{-6}$.
In the following figures we plot the logarithmic contours of $\|\boldsymbol{F}(\mathbf{x})\|_2$ in a neighborhood of the zero $\boldsymbol{\alpha} = (0, 0)^T$ and the sequence of solutions starting from $\mathbf{x}^{(0)} = (1.5, -2)^T$ and $\mathbf{x}^{(0)} = (4, 4)^T$. The function $\|\boldsymbol{F}(\mathbf{x})\|_2$ grows exponentially fast as we move away from $\boldsymbol{\alpha}$. As a consequence, the magnitude of the Newton correction term associated to the iterate $\mathbf{x}^{(n)}$ is $\left\|J_F(\mathbf{x}^{(n)})^{-1}\boldsymbol{F}(\mathbf{x})\right\|$ which is small for $\left\|\mathbf{x}^{(n)}\right\| \gg 1$. For the initial guess $\mathbf{x}^{(0)} = (4, 4)^T$ we already need $n_c = 50$ iterations to reach a solution that satisfies the stopping tolerance; we can see that this is due to the stagnation of the method during the initial iterations.