Chapter 2 The Solution of Nonlinear Equations $f(x) = 0$

In this chapter we will study methods for find the solutions of functions of single variables, i.e. values of x such that $f(x) = 0$.

For example, $f(x) = x - cos(x)$ find x such that $f(x) = 0$. We could look directly for values of x where $f(x)$ crosses x-axis.

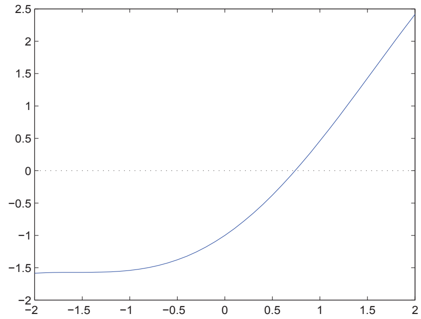

Figure 1: Graph of $y = x - cos(x)$

Section 2.1 Fixed Point Iteration for Solving $x = g(x)$

To solve $x - cos(x) = 0$ we can rearrange the equation in the form $x = cos(x)$. The values of x we are seeking then occur where curve $y = cos(x)$ and $y = x$ intersect. From the figure below, there is only one point of intersection in this example, near 0.7.

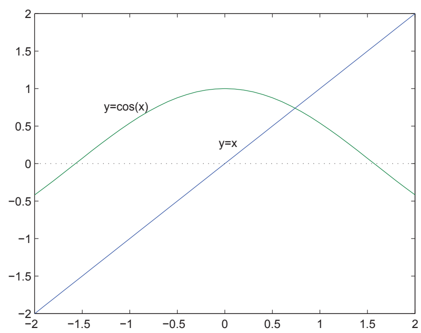

Figure 2: Intersection of $y = cos(x)$ and $y = x$

In general, to obtain a fixed point iteration, we rearrange $f(x) = 0$ into the form $x = g(x)$ so that if x satisfies $f(x) = 0$ then it also satisfies $x = g(x)$. Then the fixed point iteration is the computation $p_{k+1} = g(p_k)$ together with a starting point p_0. The sequence has the pattern

$$\begin{aligned} &p_0 \\ &p_1 = g(p_0) \\ &p_2 = g(p_1) \\ &\vdots \\ &p_k = g(p_{k-1}) \\ &p_{k+1} = g(p_k) \\ &\vdots \end{aligned}$$

Do the numbers Converge or Diverge??

Consider $g(x) = cos(x)$. To perform the iteration $p_{k+1} = g(p_k) = cos(x_k)$ we require a start guess p_0. The computation proceeds as follows:

$$\begin{aligned} p_0 &= 0.75 \text{ starting guess} \\ p_1 &= cos(p_0) = 0.73169 \\ p_2 &= cos(p_1) = 0.74405 \\ p_3 &= cos(p_2) = 0.73573 \\ &\text{etc.} \end{aligned}$$

A **fixed point** of a function $g(x)$ is a real number P such that $P = g(P)$.

The iteration $p_{n+1} = g(p_n)$ for $n = 0, 1...$ is called **fixed point iteration**

We say that an iteration $p_{k+1} = g(p_k)$ **converges** to P if

$$p_i \to P \text{ as } i \to \infty$$

Example:

$$x^2 - x - 1 = 0$$

$x^2 = x + 1$
$x = 1 + \frac{1}{x}$
$p_{k+1} = 1 + \frac{1}{p_k}$

pick $p_0 = 2$
$p_1 = 1 + \frac{1}{2} = 1.5$
$p_2 = 1 + \frac{1}{1.5} = 1.6666$
$p_3 = 1 + \frac{1}{1.6666} = 1.6$
$p_4 = 1 + \frac{1}{1.6} = 1.625$
$p_5 = 1 + \frac{1}{1.625} = 1.612538$

converging to 1.618...

$x^2 - x = 1$
$x(x - 1) = 1$
$x = \frac{1}{x-1}$
$p_{k+1} = \frac{1}{p_k - 1}$

pick $p_0 = 1.6$
$p_1 = \frac{1}{1.6-1} = 1.6666$
$p_2 = \frac{1}{1.6666-1} = 1.5$
$p_3 = \frac{1}{1.5-1} = 2$
$p_4 = \frac{1}{2-1} = 1$

not converging

```
function Example (n)

p = zeros (1, n);

%p (1) = 2;
p (1) = 1.6;
for k = 2 : n
    %p (k) = 1 + 1 / p (k - 1);
    p (k) = 1 / (p (k - 1) - 1);
end

p (n)
```

Theorem 2.3 (Fixed-point Theorem). Assume $g, \acute{g} \in C[a,b]$, K is a positive constant, $p_0 \in (a,b)$, and $g(x) \in [a,b]$ for all $x \in [a,b]$.

if $|\acute{g}(x)| \leq K < 1$ for all $x \in [a,b]$, then iteration $p_n = g(p_{n-1})$ will converge to the unique fixed point $P \in [a,b]$.

if $|\acute{g}(x)| > 1$ for all $x \in [a,b]$, then iteration $p_n = g(p_{n-1})$ will not converge to P.

Example: $g(x) = 1 + x - \frac{x^2}{4}$. The two solutions (fixed points of g) are $x = -2$ and $x = 2$. The derivatives of the function $\acute{g}(x) = 1 - \frac{x}{2}$.

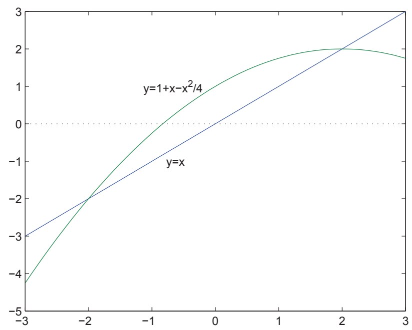

Figure 3: Intersection of $y = 1 + x - \frac{x^2}{4}$ and $y = x$

case (i): $P = -2$

start with $p_0 = -2.05$
$p_1 = -2.100625$
$p_2 = -2.20378$
$p_3 = -2.417944$
$\vdots$
$\lim_{n\to\infty} p_n = -\infty$

$|\acute{g}(x)| > \frac{3}{2}$ on $[-3, -1]$
Not Converging

case (ii): $P = 2$

start with $p_0 = 1.6$
$p_1 = 1.96$
$p_2 = 1.99996$
$p_3 = 1.999999996$
$\vdots$
$\lim_{n\to\infty} p_n = 2$

$|\acute{g}(x)| < \frac{1}{2}$ on $[1, 3]$
Converging

What will happen when $\acute{g}(P) = 1$??

Example: $g(x) = 2(x-1)^{1/2}$ for $x \geq 1$. One fixed point $P = 2$ exists. The derivatives of the function $\acute{g}(x) = \frac{2}{(x-1)^{1/2}}$ and $\acute{g}(2) = 1$.

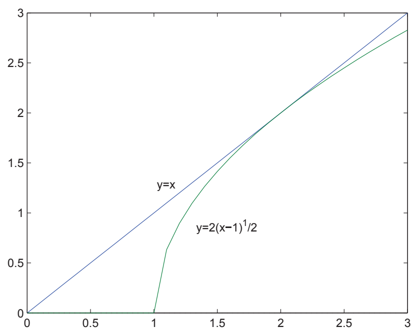

Figure 4: Intersection of $y = 2(x-1)^{1/2}$ and $y = x$

Consider 2 values that lie to left and right of $P = 2$

case (i) start with $p_0 = 1.5$	case (i) start with $p_0 = 2.5$
$p_1 = 1.4142$	$p_1 = 2.4492$
$p_2 = 1.2871$	$p_2 = 2.4078$
$p_3 = 1.0717$	$p_3 = 2.3730$
$p_4 = 0.5359$	$p_4 = 2.3435$
$\vdots$	$\vdots$
$p_5 = 2(-0.4640)^{1/2}$	$\lim_{n\to\infty} p_n = 2$
p_4 lies outside the domain	converging too slowly to $P = 2$
p_5 **can not be computed**	**Converging**

Absolute and Relative Error Consideration

What about a criterion for stopping iteration?

The difference between consecutive terms that shows the closeness of consecutive terms does not guarantee that accuracy has been achieved. But it is usually the only criterion available and is often used to terminate an iterative procedure.

```
function [k, p, err] = FixedPoint (p0, tol, max)
% tol is tolerance
% max is maximum number of iterations

% k is number of iterations
% p is approximation to fixed point
% err is error of approximation

P (1) = p0;
for k = 2 : max
    P (k) = 1 + P (k - 1) - P (k - 1) ^ 2 / 4;
    err = abs (P(k) - P (k - 1));
    relerr = err / (abs (P(k) + eps));
    p = P (k);

    if (err < tol) | (relerr < tol)
        break;
    end
end

if k == max
    disp ('Maximum number of iterations exceeded')
end
```

Section 2.2 Bracketing Methods for Locating a Root

Definition (Root of an Equation, Zero of a Function).
Assume that $f(x)$ is a continuous function. Any number r for which $f(x) = 0$ is called a **root of the equation** $f(x) = 0$. Also, we say r is a **zero of the function** $f(x)$.

For example
The equation $2x^2 + 5x - 3$ has two roots $r_1 = 0.5$ and $r_2 = -3$.
The corresponding function $f(x) = 2x^2 + 5x - 3$ has two real zeros, $r_1 = 0.5$ and $r_2 = -3$.

The Bisection Method of Bolzano

We start with interval $[a, b]$, where $f(a)$ and $f(b)$ have opposite signs. Since $y = f(x)$ is a continuous function, it will cross x-axis at zero $x = r$. The bisection method moves the end points closer together until we obtain an interval of small width that brackets the zero.

Find midpoint $c = \frac{a+b}{2}$

if $f(a)$ and $f(c)$ have opposite signs, zero lies in $[a, c]$
if $f(c)$ and $f(b)$ have opposite signs, zero lies in $[c, b]$
if $f(c) = 0$, then the zero is c

Repeat the process

$[a_0, b_0]$ is the starting interval and $c_0 = \frac{a_0+b_0}{2}$ is the midpoint
$[a_1, b_1]$ is the second interval and $c_1 = \frac{a_1+b_1}{2}$ is the midpoint
$\vdots$
$[a_n, b_n]$ is the nth interval and $c_n = \frac{a_n+b_n}{2}$ is the midpoint

where $c_n = \frac{a_n+b_n}{2}$, and if $f(a_{n+1})f(b_{n+1}) < 0$, then

$[a_{n+1}, b_{n+1}] = [a_n, c_n]$ or $[a_{n+1}, b_{n+1}] = [c_n, b_n]$ for all n.

Theorem (Bisection Method). Assume that $f \in C[a, b]$ and there exists a number $r \in [a, b]$ such that $f(r) = 0$. If $f(a)$ and $f(b)$ have opposite signs and $\{c_n\}_{n=0}^{\infty}$ represents the sequence of midpoints, then

$$|r - c_n| \le \frac{b-a}{2^{n+1}} \text{ for } n = 0, 1, \cdots$$

and the sequence $\{c_n\}_{n=0}^{\infty}$ converges to the zero $x = r$; that is

$$lim_{n \to \infty} c_n = r$$

.

Proof. The distance between c_n and r cannot be greater than half the width of the interval. Thus

$$|r - c_n| \le \frac{b_n - a_n}{2} \qquad (*)$$

Observe that

$$b_1 - a_1 = \frac{b_0 - a_0}{2^1}$$
$$b_2 - a_2 = \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}$$

By induction

$$b_n - a_n = \frac{b_0 - a_0}{2^n} \qquad (**)$$

Combining (*) and (**) results in

$$|r - c_n| \le \frac{b-a}{2^{n+1}} \text{ for all n}$$

Example: $f(x) = xsin(x) - 1$ for $x \in [0, 2]$. Use bisection method to find a zero of the function.

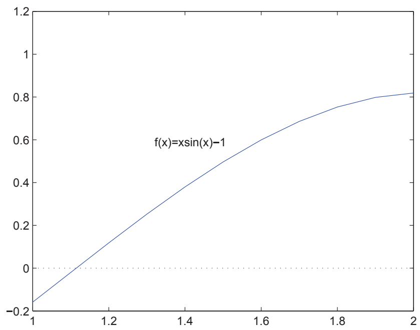

Figure 5: Graph of $f(x) = xsin(x) - 1$

Starting with $a_0 = 0$ and $b_0 = 2$ we compute $f(0) = -1.0$ and $f(2) = 0.8185$. so a root lies in the interval $[0, 2]$.

At the midpoint $c_0 = 1$ we find $f(1) = -0.1589$. Hence the function changes sign on $[c_0, b_0] = [1, 2]$.

Set $a_1 = c_0$ and $b_1 = b_0$. The midpoint is $c_1 = 1.5$ and $f(c_1) = 0.4962$.

$f(1) = -0.1589$ and $f(1.5) = 0.4962$, then root lies in interval $[a_1, c_1] = [1.0, 1.5]$.

Set $a_2 = a_1$ and $b_2 = c_1$.

$\vdots$

We obtain a sequence $\{c_k\}$ that converges to $r \approx 1.114157$

```
function [c, err, yc] = Bisecttion (a, b, delta)
%Input   - a and b are the left and right endpoints
%        - delta is the tolerance
%Output  - c is the zero
%        - yc= f(c)
%        - err is the error estimate for c

ya = a * sin (a) - 1;
yb = b * sin (b) - 1;
if ya * yb > 0,return,end

max = 1 + round ((log(b-a)-log(delta))/log(2));
for k=1:max
    c = (a+b)/2;
    yc = c * sin (c) - 1;
    if yc == 0
        a = c;
        b = c;
    elseif yb * yc > 0
        b = c;
        yb = yc;
    else
        a = c;
        ya = yc;
    end
    if b - a < delta, break,end
end

c = (a+b)/2;
err = abs (b-a);
yc = c * sin (c) - 1;
```

Method of False Position

The bisection method converges at a fairly slow speed.

A better approximation for the midpoint c is obtained if we find a point $(c, 0)$ where the secant line L joining the points $(a, f(a))$ and $(b, f(b))$ crosses the x-axis.

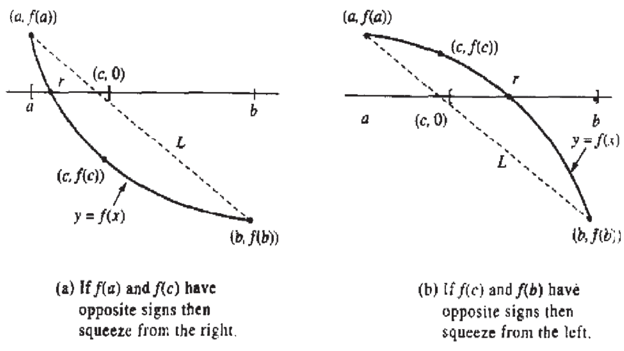

Figure 6: False position method

To find c, we write two versions of the slope m of line L:

$$m = \frac{f(b)-f(a)}{b-a} \qquad (*)$$

$$m = \frac{0-f(b)}{c-b} \qquad (**)$$

Slope in (*) and (**)

$$\frac{f(b)-f(a)}{b-a} = \frac{0-f(b)}{c-b}$$

$$c = b - \frac{f(b)(b-a)}{f(b)-f(a)}$$

At each step the approximation of zero r is

$$c_n = b_n - \frac{f(b_n)(b_n-a_n)}{f(b_n)-f(a_n)}$$

Example: $f(x) = xsin(x) - 1$ for $x \in [0, 2]$. Use false position method to find a zero of the function.

Starting with $a_0 = 0$ and $b_0 = 2$ we compute $f(0) = -1.0$ and $f(2) = 0.8185$. so a root lies in the interval $[0, 2]$.

The midpoint $c_0 = 2 - \frac{0.8185(2-0)}{0.08185-(-1)} = 1.0997$ and $f(c_0) = -0.2001$. Hence the function changes sign on $[c_0, b_0] = [1.0997, 2]$. Set $a_1 = c_0$ and $b_1 = b_0$.

The midpoint is $c_1 = 2 - \frac{0.08185(2-1.0997)}{0.08185-(-0.2001)} = 1.1212$ and $f(c_1) = 0.00983$.

$\vdots$

We obtain a sequence $\{c_k\}$ that converges to $r \approx 1.114157$

```
%%function [c,err,yc] = FalsePosition (a,b,delta,epsilon,max)
%     - delta is the tolerance for the zero
%     - epsilon is the tolerance for the value of f at the zero

 ya = a * sin (a) - 1;
 yb = b * sin (b) - 1;

 if ya * yb >0
disp ('Note: f(a)*f(b) >0'),
  return,
 end

 for k=1:max
dx = yb * (b-a) / (yb-ya);
c = b - dx;
ac = c - a;
yc = c * sin (c) - 1;
if yc == 0,break;
elseif yb * yc > 0
    b = c;
    yb = yc;
else
    a = c;
    ya = yc;
end
dx = min(abs(dx),ac);
if abs(dx) <delta,break,end
if abs(yc) <epsilon, break,end
 end

 c;
 err=abs(b-a)/2;
 yc = c * sin (c) - 1;
```

2.3 Checking for Convergence

Termination criterion or strategy must be designed ahead of time so that the computer will stop when an accurate approximation is reached.
One termination is $|f(p_n)| < \epsilon$ for some small ϵ
Another termination $|p_n - P| < \delta$ for some small δ

If we require that $|p_n - P| < \delta$ and $|f(p_n)| < \epsilon$, then point P_n will be located in the rectangular region about the solution $(p, 0)$, as shown in figure 2.12(a).

If we require that $|p_n - P| < \delta$ or $|f(p_n)| < \epsilon$, then point P_n will be located anywhere in the region formed by the union of the horizonal and vertical stripes, as shown in figure 2.12(b).

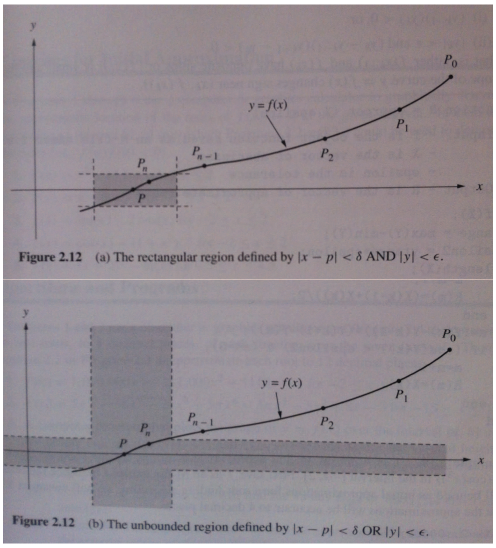

Figure 2.12 (a) The rectangular region defined by $|x - p| < \delta$ AND $|y| < \epsilon$.

Figure 2.12 (b) The unbounded region defined by $|x - p| < \delta$ OR $|y| < \epsilon$.

Practice

Use Fixed Point Iteration program to approximate the fixed points (if any) of each function. Answers should be accurate to 12 decimals places. Produce a graph of each function and the line $y = x$ that clearly shows any fixed points.

(a) $g(x) = x^5 - 3x^3 - 2x^2 + 2$

(b) $g(x) = cos(sin(x))$

(c) $g(x) = x^2 - sin(x + 0.15)$

(d) $g(x) = x^{x - cos(x)}$

What will happen if the bisection method is used with function $f(x) = \frac{1}{(x-2)}$ and **(a) the interval** $[3, 7]$ **(b) the interval** $[1, 7]$

2.4 Newton-Raphson Method

Assume that the initial approximation p_0 is near the root p, The graph $y = f(x)$ intersects the x-axis at the points $(p, 0)$ and the point $(p_0, f(p_0))$ lies on the curve near $(p, 0)$.

Define p_1 to be the point of intersection of the x-axis and the tangent line to the curve at the point $(p_0, f(p_0))$. Then p_1 will be closer to p than p_0.

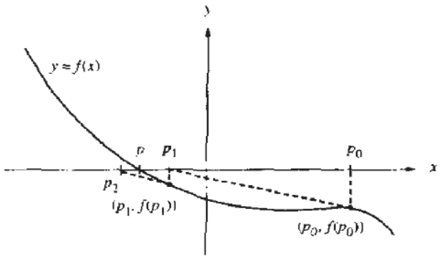

Figure 2.13 The geometric construction of p_1 and p_2 for the Newton-Raphson method.

If we write to versions of the slope of the tangent line L:

$$m = \frac{0 - f(p_0)}{p_1 - p_0} \qquad (*)$$

$$m = f'(p_0) \qquad (**)$$

Combining (*) and (**), we get

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}$$

The process above can be repeated to obtain the sequence $\{p_k\}$ that converges to p

Theorem (Newton Raphson Theorem) Assume that $f \in C^2[a,b]$ and there exists and number $p \in [a,b]$ where $f(p) = 0$. if $\acute{f}(p) \neq 0$ then there exists $\delta > 0$ such that the sequence $\{p_k\}_{k=0}^{\infty}$ define by the iteration

$$p_k = g(p_{k-1}) = p_{k-1} - \frac{f(p_{k-1})}{\acute{f}(p_{k-1})} \text{ for } k = 1, 2, \ldots$$

will converge to p for initial approximation $p \in [p - \delta, p + \delta]$

Remark: the function $g(x)$ is define as

$$g(x) = x - \frac{f(x)}{\acute{f}(x)}$$

Corollary (Newton's Iteration for finding Square Root)
Assume that $A > 0$ is a real number and let $p_0 > 0$ be initial approximation to $\sqrt{A}$. Define the sequence $\{p_k\}_{k=0}^{\infty}$ using the recursive rule

$$p_k = \frac{p_{k-1} + \frac{A}{p_{k-1}}}{2} \text{ for } k = 1, 2, \ldots$$

Then the sequence $\{p_k\}_{k=0}^{\infty}$ will converge to $\sqrt{A}$

Proof: Start with the function $f(x) = x^2 - A$, and notice the roots of the equation $x^2 - A = 0$ are $\pm\sqrt{A}$.

Write down the Newton Raphson iteration

$$g(x) = x - \frac{f(x)}{\acute{f}(x)} = x - \frac{x^2 - A}{2x}$$

The formula can be simplified to obtain

$$g(x) = \frac{x + \frac{A}{x}}{2}$$

Secant Method

The Newton-Raphson method requires the evaluation of two functions $f(p_{k-1})$ and $\acute{f}(p_{k-1})$.
Two initial points $(p_0, f(p_0)$ and $(p_1, f(p_1))$ near the point $(p, 0)$ are needed. Define p_2 to be the intersection point through these two points and the x-axis; then p_2 will be closer to p than either p_0 or p_1.

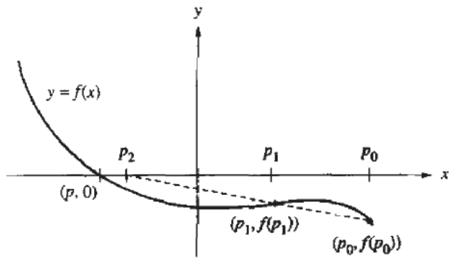

Figure 2.16 The geometric construction of p_2 for the secant method.

Figure 8: Secant method

The slop

$$m = \frac{f(p_1)-f(p_0)}{p_1-p_0} \text{ and } m = \frac{0-f(p_1)}{p_2-p_1}$$

$$p_2 = g(p_0, p_1) = p_1 - \frac{f(p_1(p_1-p_0)}{f(p_1)-f(p_0)}$$

The general term is given by the two point iteration formula

$$p_{k+1} = g(p_{k-1}, p_k) = p_k - \frac{f(p_k(p_k-p_{k-1})}{f(p_k)-f(p_{k-1})}$$

```
function [p0,err,k,y]=NewtonRaphson (f,df,p0,delta,epsilon,max)

%Input - f is the object function input as a string 'f'
%       - df is the derivative of f input as a string 'df'
%       - p0 is the initial approximation to a zero of f
%      - delta is the tolerance for p0
%      - epsilon is the tolerance for the function values y
%      - max is the maximum number of iterations
%Output - p0 is the Newton-Raphson approximation to the zero
%       - err is the error estimate for p0
%       - k is the number of iterations
%       - y is the function value f(p0)

for k = 1: max
    p1 = p0 - feval (f,p0) / feval(df,p0);
    err = abs(p1-p0);
    relerr = 2*err / (abs(p1)+delta);
    p0 = p1;
    y = feval (f,p0);
    if (err<delta) | (relerr<delta) | (abs(y)<epsilon),break,end
end
```

```
function [p1,err,k,y]=Secant (f,p0,p1,delta,epsilon,max)

%Input - f is the object function input as a string 'f'
%       - p0 and p1 are the initial approximations to a zero of f
%      - delta is the tolerance for p1
%      - epsilon is the tolerance for the function values y
%      - max is the maximum number of iterations
%Output - p1 is the secant method approximation to the zero
%       - err is the error estimate for p1
%       - k is the number of iterations
%       - y is the function value f(p1)

for k=1: max
    p2= p1-feval(f,p1)*(p1-p0)/(feval(f,p1)-feval(f,p0));
 err=abs(p2-p1);
 relerr=2*err/(abs(p2)+delta);
    p0=p1;
    p1=p2;
    y=feval(f,p1);
    if (err<delta)|(relerr<delta)|(abs(y)<epsilon),break,end
end
```

Practice

$f(x) = x^2 - 2x - 1$, start with $p_0 = 2.6$ and $p_1 = 2.5$.

Plot the function:

```
x=-3:0.1:3;
y=x.^2 - 2 * x - 1;
plot (x, y)
hold on
plot([-3 3], [0 0], 'k:')
hold off
```

Use **feval** function

```
function [y] = f (x)
    y = x ^ 2 - 2 * x - 1;
end

function [y] = df (x)
    y = 2 * x - 2;
end
```

Example call:
[p0,err,k,y] = NewtonRaphson ('f','df',2.6,0.000001,0.0000001,100)