Nonlinear Equations

- ♣ Bisection Method
- ♣ Newton Method
- ♣ Secant Method
- ♣ Fixed Point and Functional Iteration
- ♠ A System of Nonlinear Equations
- ♢ Other Topics with Applications

1. $x^3 + 3x - 1 = 0$

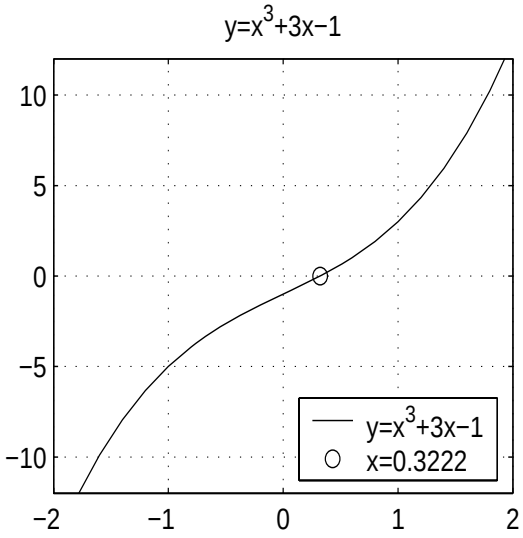
2. $e^x - sin(x) = 0, \quad e^x - 1.5 - \tan^{-1}(x) = 0$

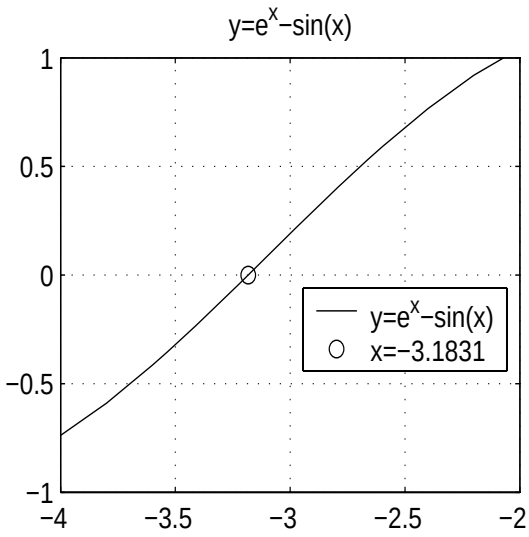
3. $3 + \frac{1}{2}\sin(x) - x = 0, \quad 4 + \frac{1}{3}\sin(2x) - x = 0$

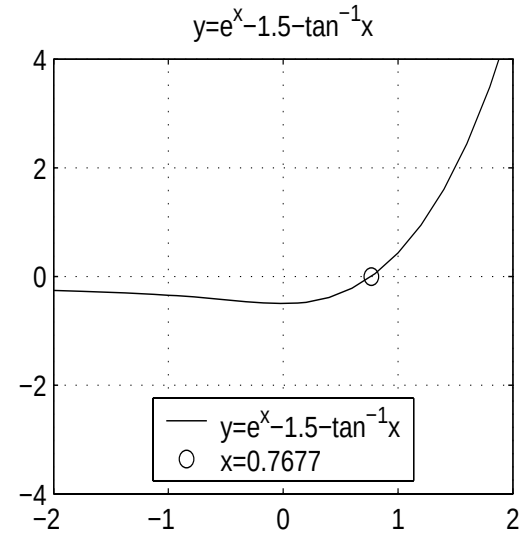
4. $x^3 - sinh(x) + 4x^2 + 6x + 9 = 0$

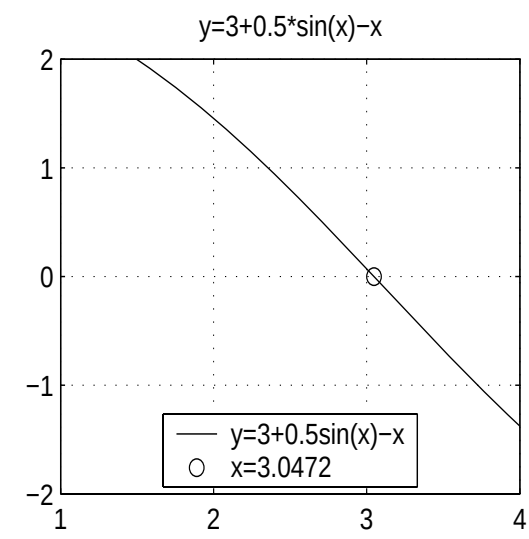
MATLAB Codes for Plot Nonlinear Functions

```
%
% Plot of four nonlinear funciotns
% Function funcA
subplot(2,2,1)
X=-2:0.2:2;
Y=X.^3+3*X-1;
V=[-2 2,-12 12];
plot(X,Y,'b-',[0.3222],[0],'ro'); axis(V); grid
% set(gca,'Xtick',[]), set(gca,'YTick',[0]); grid on
legend('y=x^3+3x-1','x=0.3222',0);
title('y=x^3+3x-1')

% Function funcB
subplot(2,2,2)
X=-4:0.2:-2;
Y=exp(X)-sin(X);
V=[-4 -2, -1 1];
plot(X,Y,'b-',[-3.1831],[0],'ro'); axis(V); grid
% set(gca,'Xtick',[]), set(gca,'YTick',[0]); grid on
legend('y=e^x-sin(x)','x=-3.1831',0);
title('y=e^x-sin(x)')

% Function funcC
subplot(2,2,3)
X=-2:0.2:2;
Y=exp(X)-1.5-atan(X);
V=[-2 2,-4 4];
plot(X,Y,'b-',[0.7677],[0],'ro'); axis(V); grid
% set(gca,'Xtick',[]), set(gca,'YTick',[0]); grid on
legend('y=e^x-1.5-tan^{-1}x','x=0.7677',0);
title('y=e^x-1.5-tan^{-1}x')

% Function funcD
subplot(2,2,4)
X=1:0.2:4;
Y=3+0.5*sin(X)-X;
V=[1 4,-2 2];
plot(X,Y,'b-',[3.0472],[0],'ro'); axis(V); grid
% set(gca,'Xtick',[]), set(gca,'YTick',[0]); grid on
legend('y=3+0.5sin(x)-x','x=3.0472',0);
title('y=3+0.5*sin(x)-x')
```


y=x^3+3x−1
y=x^3+3x−1
x=0.3222


y=e^x−sin(x)
y=e^x−sin(x)
x=−3.1831


y=e^x−1.5−tan^−1x
y=e^x−1.5−tan^−1x
x=0.7677


y=3+0.5*sin(x)−x
y=3+0.5sin(x)−x
x=3.0472

Bisection Algorithm

Let f be a continuous function on $[a, b]$ and $f(a)f(b) < 0$. This algorithm is to find a $c \in [r, s] \subseteq [a, b]$ such that$|f(c)| < \varepsilon$ and $|s - r| < \delta$, where δ are user-specified small numbers.

input $a,\ b,\ M,\ \delta,\ \varepsilon$
$u \leftarrow f(a)$
$v \leftarrow f(b)$
$e \leftarrow b - a$
$k \leftarrow 0$
print $k,\ a,\ b,\ u\ ,\ v$
if sgn(u **)=sgn(**v**) return**
for k=1, 2 ...,M
    $e \leftarrow e/2$
    $e \leftarrow a + e$
    $w \leftarrow f(c)$
print $k,\ a,\ b,\ c,\ f(c)$
    if $|e| < \delta$ **or** $|w| < \varepsilon$ **return**
    if sgn(w **)≠sgn(**u**) then**
        $b \leftarrow c$
        $v \leftarrow w$
    else
        $a \leftarrow c$
        $u \leftarrow w$
    endif
endfor

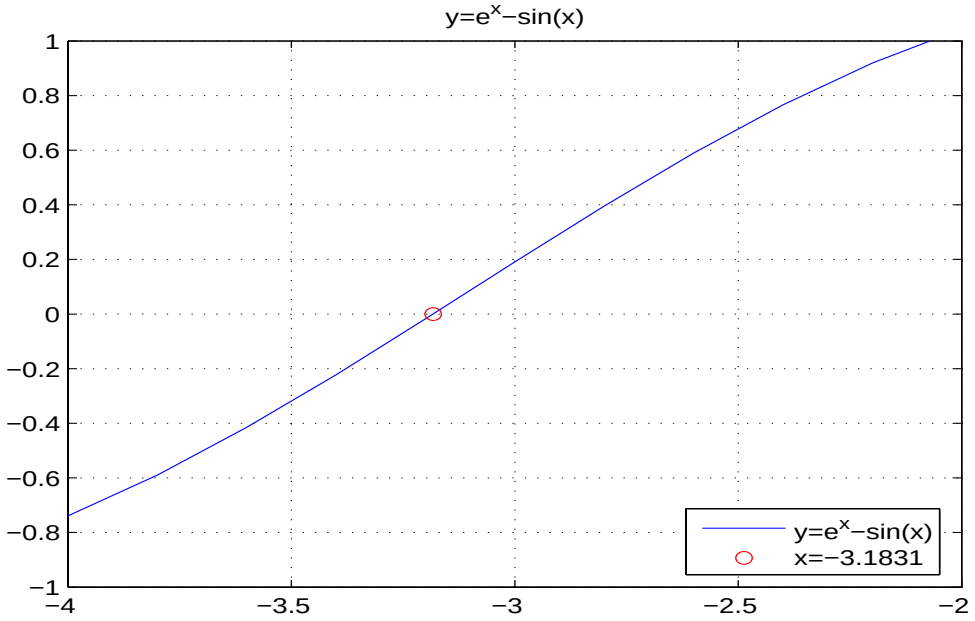


Figure 1: The function of $e^x - sin(x)$

```
/* Brute-force method for finding Zero of f(x) */

#include <stdio.h>
#include <math.h>
main()
{
  int i, max_num=51;
  double x, y,dx=0.02;

  x=-4.0; y=exp(x)-sin(x);
  for(i=0; i<max_num; i++)
  {
   printf("%11.6f, %11.6f ",x,y);
   x+=dx; y=exp(x)-sin(x);
  }
}
```

Newton's Method

$$0 = f(x+h) = f(x) + hf'(x) + O(h^2)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \textbf{for} \quad n \geq 0$$

Algorithm

input x, M
$k \leftarrow 0$
$y \leftarrow f(x)$
print k, x, y
for k =1, 2,. . .,M
    $x \leftarrow x - \frac{y}{f'(x)}$
    $y \leftarrow f(x)$
    print k, x, y
endfor

Example

$$f(x) = e^x - 1.5 - \tan^{-1} x \quad \Longrightarrow \quad f'(x) = e^x - \frac{1}{1+x^2}$$

0	-7.0000000, -0.070189
1	-10.677096, -0.022567
2	-13.279167, -0.004366
3	-14.053656, -0.000239
4	-14.101110, -0.000001
5	-14.101270, -0.000000
6	-14.101270, -0.000000

```
/* Newton's method for finding the zero of f(x)=exp(x)-1.5-atan(x) */

#include <stdio.h>
#include <sys/time.h>
#include <math.h>

main( )
{
   int k, max_num=10;
   double x, y, z;

   x=-7.0; y=exp(x)-1.5-atan(x);

   for(k=0; k<max_num; k++)
   {
     printf("t%2d %11.6f %11.6f\n',k,x,y);
     z=exp(x)-1.0/(1.0+x*x);
     x-=(y/z);
     y=exp(x)-1.5-atan(x);
    }
}
```

Secant Method

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$x_{n+1} = x_n - f(x_n) \left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right] \quad for \quad n \geq 1$$

Algorithm

input x, y, M,

$k \leftarrow 0$

$zx \leftarrow f(x)$

$zy \leftarrow f(y)$

print k, x, zx

for k=1, 2 ...,M

    print k, y, zy,

    t≤y-zy*(y-x) / (zy-zx)

    $x \leftarrow y$

    $zx \leftarrow zy$

    $y \leftarrow t$

    $zy \leftarrow f(y)$

endfor

Example

$$f(x) = x^3 - sinh(x) + 4x^2 + 6x + 9$$

where

$$sinh(x) = \frac{e^x - e^{-x}}{2}$$

0	7.000000, 41.683877
1	8.000000, -665.478826
2	7.058945, 20.798251
3	7.087465, 10.037673
4	7.114068, -0.401497
5	7.113045, 0.007307
6	7.113063, -0.000005
7	7.113063, 0.000000
8	7.113063, -0.000000

```
/* Secant method for finding the zero of */
/* f(x)=x sup 3 -sinh(x)+ 4x sup 2 + 6x + 9 */

#include <stdio.h>
#include <sys/time.h>
#include <math.h>

main()
{
   int k, max_num=10;
   /*double t, x, y, zx, zy */
   float t, x, y , zx, zy;

   x=7.0; y=8.0;
   zx=x*x*x-0.5*(exp(x)-exp(-x))+4.0*x*x+6.0*x+9.0;
   zy=y*y*y-0.5*(exp(y)-exp(-y))+4.0*y*y+6.0*y+9.0;

   printf("\t %2d %11.6f, %11.6f\n",k,x,zx);

   for (k=1; k<max_num; k++)
   {
    printf("\t %2d %11.6f, %11.6f\n",k,y,zy);
    t=y-zy*(y-x)/(zy-zx);
    x=y;
    zx=zy;
    y=t;
    zy=y*y*y-0.5*(exp(y)-exp(-y))+4.0*y*y+6.0*y+9.0;
    }
}
```

Fixed Point and Functional Iteration

$$x_{n+1} = F(x_n) = x_n - \frac{f(x_n)}{f'(x_n)} \quad \forall\ n \geq 0 \qquad (Newton)$$

$$F(x) = x - \frac{f(x)}{f'(x)}$$

Definition: A mapping $F:\ X\ \rightarrow\ Y$ is said to be contractive if $\exists$ a number $\lambda < 1$ such that

$$|F(x) - F(y)|\ <\ \lambda|x - y|\ \ \forall\ x, y \in X$$

Theorem: Let F be a contractive mapping of a compact set $\mathbf{S}$ of $\mathbf{R}^n$ into $\mathbf{S}$. Then F has a unique fixed point. Moreover, this fixed point is the limit of every sequence obatined from $\mathbf{x}_{n+1} = F(\mathbf{x}_n)$ with any starting point $\mathbf{x}_0 \in \mathbf{S}$.

Hint of Proof
show every sequence in the contractive mapping theorem satisfies Cauchy criterion for convergence . For any $\varepsilon > 0,\ \exists$ an N such that if $n \geq m \geq N$, we have

$$||\mathbf{x}_n - \mathbf{x}_m|| \leq ||\mathbf{x}_n - \mathbf{x}_{n-1}|| + ||\mathbf{x}_{n-1} - \mathbf{x}_{n-2}|| + \ldots + ||\mathbf{x}_{m+1} - \mathbf{x}_m||$$

$$\leq \lambda^{n-1}||\mathbf{x}_1 - \mathbf{x}_0|| + \lambda^{n-2}||\mathbf{x}_1 - \mathbf{x}_0|| + \ldots + \lambda^m||\mathbf{x}_1 - \mathbf{x}_0||$$

$$\leq \frac{\lambda^N}{1-\lambda}||\mathbf{x}_1 - \mathbf{x}_0|| \leq \varepsilon$$

Example 1. $F(x) = 3 + \frac{1}{2}sinx$

Example 2. $F(x) = 4 + \frac{1}{3}sin(2x)$

Error Analysis and Accelerating Convergence

A sequence $\{p_n\}$ of approximations to a number p converges

$$linearly \;\; if \;\; |p_{n+1} - p| \leq M|p_n - p| \;\; for \; large \; n \; and \; 0 < M < 1$$

$$quadratically \;\; if \;\; |p_{n+1} - p| \leq M|p_n - p|^2 \;\; for \; large \; n \; and \; 0 < M < 1$$

- *Aitken's Δ^2 method* and *Convergence*
  If $\{p_n\}_0^\infty$ is a sequence that converges linearly to p and $(p_{n+1} - p)(p_n - p) > 0$, define

$$q_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n} \; or \; q_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n},$$

  then $limit_{n\to\infty} \frac{q_n - p}{p_n - p} = 0$ that is, $\{q_n\}_0^\infty$ converges to p more rapidly.

Example 1. $\{p_n = \cos(1/n)\}_{n=1}^\infty$.

n	$p_n = \cos(\frac{1}{n})$	q_n
1	0.54030	0.96178
2	0.87758	0.98213
3	0.94496	0.98979
4	0.96891	0.99342
5	0.98007	0.99541
6	0.98614	
7	0.98981	

Table 1: **The Speed of Convergence for** $\cos(\frac{1}{n})$

Proof from P.260, Kincaid

Let $h_n = p_n - p$, then

$$\begin{aligned} q_n &= p_n - \frac{(p_{n+1}-p_n)^2}{p_{n+2}-2p_{n+1}+p_n} \\ &= \frac{(p_n p_{n+1}-2p_n p_{n+1}+p_n^2)-(p_{n+1}^2-2p_n p_{n+1}+p_n^2)}{p_{n+2}-2p_{n+1}+p_n} \\ &= \frac{(p+h_n)(p+h_{n+1})-(p+h_{n+1})^2}{(p+h_{n+2})-2(p+h_{n+1})+(p+h_n)} \\ &= \frac{p(h_n+h_{n+2}-2h_{n+1})+h_n h_{n+1}-h_{n+1}^2}{h_{n+2}-2h_{n+1}+h_n} \\ &= p + \frac{h_n h_{n+1}-h_{n+1}^2}{h_{n+2}-2h_{n+1}+h_n} \end{aligned}$$

Since $|p_{n+1} - p| \leq M|p_n - p|$ with $0 < M < 1$ for large n, then $\exists\ \delta_n \geq 0$ with $limit_{n\to\infty}\delta_n = 0$ such that

$$h_{n+1} = (M + \delta_n)h_n, \quad h_{n+2} = (M + \delta_{n+1})(M + \delta_n)h_n.$$

Then

$$\begin{aligned} q_n - p &= \frac{h_n^2[(M+\delta_{n+1})(M+\delta_n)-(M+\delta_n)^2]}{[(M+\delta_{n+1})(M+\delta_n)-2(M+\delta_n)+1]h_n} \\ &= h_n \cdot \frac{(M+\delta_{n+1})(M+\delta_n)-(M+\delta_n)^2}{(M+\delta_{n+1})(M+\delta_n)-2(M+\delta_n)+1} \\ &= (p_n - p) \cdot \frac{(M+\delta_{n+1})(M+\delta_n)-(M+\delta_n)^2}{(M+\delta_{n+1})(M+\delta_n)-2(M+\delta_n)+1} \end{aligned}$$

Thus

$$\frac{q_n - p}{p_n - p} = \frac{M(\delta_{n+1} - \delta_n) + \delta_{n+1}\delta_n - \delta_n^2}{(M-1)^2 + M(\delta_{n+1} + \delta_n) + \delta_{n+1}\delta_n - 2\delta_n} \ \to\ 0 \ \ as\ n \ \to\ 0$$

Müller's Method

It attempts to overcome two problems, in particular, occurred in Newton's method,

(a) $f(x_n) \approx 0$ and $f'(x_n) \approx 0$ for some steps.

(b) unable to find a complex root if starting with a real initial number.

Given 3 initial approximations p_0, p_1, p_2 for a solution of $f(x) = 0$. Consider a quadratic polynomial passing through $[p_0, f(p_0)]$, $[p_1, f(p_1)]$, $[p_2, f(p_2)]$

$$f(x) = a(x - p_2)^2 + b(x - p_2) + c$$

where a, b, c are to be determined, the next approximation p_3 is defined to be one of the roots of $f(x) = 0$ computed by

$$p_3 = p_2 - \frac{2c}{b + sgn(b)\sqrt{b^2 - 4ac}} \quad (\textit{the } p_3 \textit{ closest to } p_2)$$

where $c = f(p_2)$

$$b = \frac{(p_0 - p_2)^2[f(p_1) - f(p_2)] - (p_1 - p_2)^2[f(p_0) - f(p_2)]}{(p_0 - p_1)(p_0 - p_2)(p_1 - p_2)}$$

$$a = \frac{(p_1 - p_2)[f(p_0) - f(p_2)] - (p_0 - p_2)[f(p_1) - f(p_2)]}{(p_0 - p_1)(p_0 - p_2)(p_1 - p_2)}$$

Notice that a, b, c are obtained by solving

$$\begin{aligned}
(p_0 - p_2)^2 a &+ (p_0 - p_2) b + c = f(p_0) \\
(p_1 - p_2)^2 a &+ (p_1 - p_2) b + c = f(p_1) \\
(p_2 - p_2)^2 a &+ (p_2 - p_2) b + c = f(p_2)
\end{aligned}$$

• $f(x) = 16x^4 - 40x^3 + 5x^2 + 20x + 6 = 0$ has roots 1.241677, 1.970446, $-0.356062 \pm 0.162758j$.

$(p_0, p_1, p_2) = (0.5, 1.0, 1.5)$ leads to **1.24168** in 7 iterations

$(p_0, p_1, p_2) = (2.5, 2.0, 2.25)$ leads to **1.97044** in 6 iterations

$(p_0, p_1, p_2) = (0.5, -0.5, 0)$ leads to **-0.356062+0.162758j** in 8 iterations

A System of Nonlinear Equations

A typical problem of n nonlinear equations with n variables is

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= f_1(\mathbf{x}) = \mathbf{0} \\ f_2(x_1, x_2, \ldots, x_n) &= f_2(\mathbf{x}) = \mathbf{0} \\ &\cdots\cdots \\ f_n(x_1, x_2, \ldots, x_n) &= f_n(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{1}$$

or

$$\mathbf{f(x)=0}$$

By Taylor's expansion, we have

$$f_i(\mathbf{x} + \delta\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_i(\mathbf{x})}{\partial x_j} \delta x_j + O(||\delta\mathbf{x}||^2) \tag{2}$$

or

$$f_i(\mathbf{x} + \delta\mathbf{x}) \approx f_i(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial f_i(\mathbf{x})}{\partial x_j} \delta x_j \tag{3}$$

Assume $\mathbf{x} + \delta\mathbf{x}$ is the zero of $\mathbf{f}$, then

$$\sum_{j=1}^{n} \alpha_{ij}(\delta_j) = \beta_i \qquad for \quad 1 \leq i \leq n \tag{4}$$

where

$$\alpha_{ij} \equiv \frac{\partial f_i(\mathbf{x})}{\partial x_j} \quad and \quad \beta_i \equiv -f_i(\mathbf{x}) \tag{5}$$

Once the $\delta\mathbf{x}$ is solved by using(4), the next approximated solution can be updated by

$$\mathbf{x}^{(new)} = \mathbf{x}^{(old)} + \delta\mathbf{x} \quad or \quad x_i^{(new)} = x_i^{(old)} + \delta x_i, \quad 1 \leq i \leq n \tag{6}$$

The process is iterated to convergence if an initial guess $\mathbf{x}$ is close to a zero of $\mathbf{f}$

Algorithm

Given an initial guess $\mathbf{x}$ for a zero of $\mathbf{f}$, take ntrial Newton-Raphson iterations to improve the root. Stops if $||\delta\mathbf{x}||_1 \leq tolx \quad or \quad ||\delta\mathbf{f}(\mathbf{x})||_1 \leq tolf$

```
Subroutine Mnewton(ntrial, x, n, tolx, tolf, alpha, beta, indx )

real x(n), alpha(n, n), beta(n)
integer indx(n)
for k=1, ..., ntrial

  call userfun( x, alpha, beta)
  errf=0.
  for k=1, ..., n
    errf=errf + abs(beta(i))
  endfor
  if (errf. le. tolf) return
  call LUdecomp(alpha, n, indx,det )
  call LUbacksb(alpha, n, indx, beta)
  errx=0.
  for k=1, ..., n
    errx = errx + abs(beta(i))
    x(i) = x(i)+ beta(i)
  endfor
  if ( errx .le. tolx  )return
endfor
Return
End
```

Example 1. Find a solution of the following equations by Newton-Raphson method using $\mathbf{x}^{(0)} = [2, 1, 1]^t$ as an initial guess.

$$\begin{aligned} xy - z^2 &= 1 \\ xyz - x^2 + y^2 &= 2 \\ e^x - e^y - z &= 3 \end{aligned}$$

```
%
% The solution is [1.1659, 1.4493, 1.4625]
%        xy  - z^2 = 1
%  xyz - x^2 + y^2 = 2
%  e^x - e^y - z   = 3
% Newton-Raphson's method for Nrun iterations
% X=[x y z]'
n=3; Nrun=50;
X=[2.0 1.0 1.0]';       % initial guess
for k=1:Nrun,
  F=[X(1)*X(2)-X(3)*X(3)-1;
     X(1)*X(2)*X(3)-X(1)*X(1)+X(2)*X(2)-2.0;
     exp(X(1))-exp(X(2))-X(3)-3.0];
  A=[X(2), X(1), -2.0*X(3);
     X(2)*X(3)-2.0*X(1), X(1)*X(3)+2.0*X(2), X(1)*X(2);
     exp(X(1)), -exp(X(2)), -1.0];

  dX=(A+0.005*eye(n))\F;
  X=X-dX;
end
format short
X',F'
```

Example 2. Find a solution of the following equations by Newton-Raphson method using $\mathbf{x}^{(0)} = [4.8, 3.3, 1.5, 1.3, 0.8, 1.2]^t$ as an initial guess, where $\mathbf{f}(\mathbf{x}) = \mathbf{C} = [2, 2, 2, 2, 2, 2]^t$.

$$
\begin{aligned}
x_1 - x_2 &= C_1 \\
x_1x_3 - x_2x_5 &= C_2 \\
x_1x_3^2 - x_2x_5^2 &= C_3 \\
x_1x_4 - x_2x_6 &= C_4 \\
x_1x_4^2 - x_2x_6^2 &= C_5 \\
x_1x_3x_4 - x_2x_5x_6 &= C_6
\end{aligned}
$$

```
% Newton-Raphson's method with Nrun iterations to solve
% A Nonlinear System of Equations from Professor Cheng-Yan Kao at NTU
% [C1,C2,C3,C4,C5,C6]=[2 2 2 2 2 2]
% [X1,X2,X3,X4,X5,X6]=[144.6946, 142.6946, 1.0000, 1.0001, 1.0000, 1.0001];
% n=6;
% fin=fopen('dataC.txt');
% header=fgetL(fin);      % read off the header line
% C=fscanf(fin,'%d',n);   % read in coefficients C1 ~ C6
% X=[5.3; 3.8; 2.2; 1.2; 1.2; 1.3];       % initial guess
%
X=[4.8; 3.3; 1.5; 1.3; 0.8; 1.2];       % initial guess
C=[2 2 2 2 2 2]';
Nrun=50;  format long
for k=1:Nrun,
  F=[X(1)-X(2)-C(1); X(1)*X(3)-X(2)*X(5)-C(2); ...
     X(1)*X(3)*X(3)-X(2)*X(5)*X(5)-C(3); X(1)*X(4)-X(2)*X(6)-C(4); ...
     X(1)*X(4)*X(4)-X(2)*X(6)*X(6)-C(5); X(1)*X(3)*X(4)-X(2)*X(5)*X(6)-C(6)];
  A=[1, -1, eps, eps, eps, eps; ...
     X(3), -X(5), X(1), 0, -X(2), 0; ...
     X(3)*X(3), -X(5)*X(5), 2*X(1)*X(3), 0, -2*X(2)*X(5), 0; ...
     X(4), -X(6), 0, X(1)+eps, 0, -X(2); ...
     X(4)*X(4), -X(6)*X(6), 0, 2*X(1)*X(4), 0, -2*X(2)*X(6); ...
     X(3)*X(4), -X(5)*X(6), X(1)*X(4), X(1)*X(3), -X(2)*X(6), -X(2)*X(5)];
  dX=(A+0.005*eye(n))\F;
  X=X-dX;
end
format short
X',F'
```