Iterative Methods

Hsiao-Lung Chan
Dept Electrical Engineering
Chang Gung University, Taiwan
chanhl@mail.cgu.edu.tw

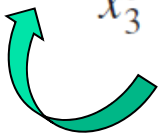
Gauss-Seidel method

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$

$$x_1^j = \frac{b_1 - a_{12}x_2^{j-1} - a_{13}x_3^{j-1}}{a_{11}}$$

$$x_2^j = \frac{b_2 - a_{21}x_1^j - a_{23}x_3^{j-1}}{a_{22}}$$

$$x_3^j = \frac{b_3 - a_{31}x_1^j - a_{32}x_2^j}{a_{33}}$$



Iterative until convergence $\varepsilon_{a,i} = \left| \frac{x_i^j - x_i^{j-1}}{x_i^j} \right| \times 100\% \leq \varepsilon_s$

An example using Gauss-Seidel method

$$
\begin{aligned}
3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\
0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\
0.3x_1 - 0.2x_2 + 10x_3 &= 71.4
\end{aligned}
$$

True solution:

$x_1 = 3,\ x_2 = -2.5,\ x_3 = 7$

Iterative equations

$$x_1 = \frac{7.85 + 0.1x_2 + 0.2x_3}{3}$$

$$x_2 = \frac{-19.3 - 0.1x_1 + 0.3x_3}{7}$$

$$x_3 = \frac{71.4 - 0.3x_1 + 0.2x_2}{10}$$

1st iteration

$$x_1 = \frac{7.85 + 0.1(0) + 0.2(0)}{3} = 2.616667$$

$$x_2 = \frac{-19.3 - 0.1(2.616667) + 0.3(0)}{7} = -2.794524$$

$$x_3 = \frac{71.4 - 0.3(2.616667) + 0.2(-2.794524)}{10} = 7.005610$$

2nd iteration

$$x_1 = \frac{7.85 + 0.1(-2.794524) + 0.2(7.005610)}{3} = 2.990557$$

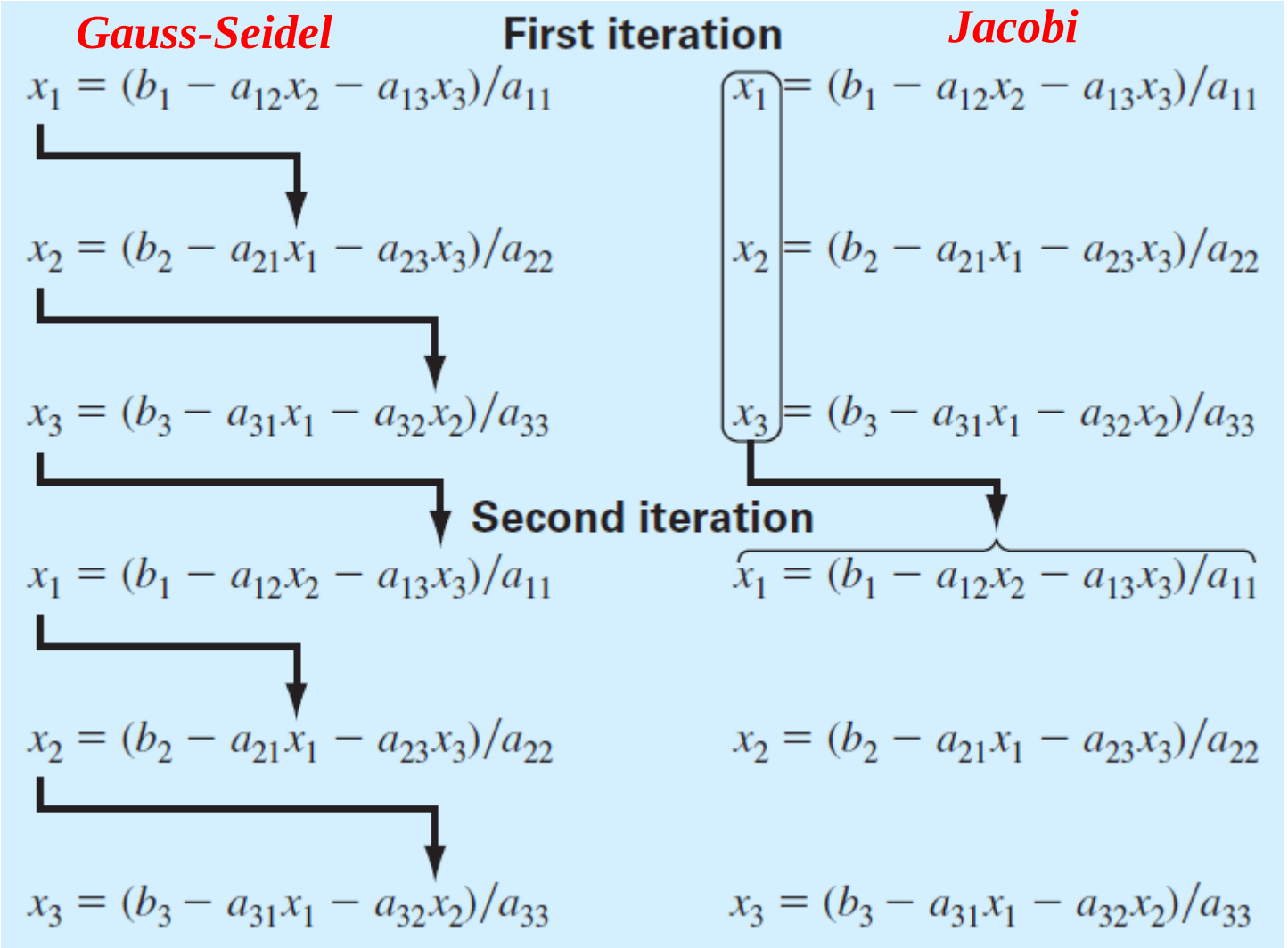
$$x_2 = \frac{-19.3 - 0.1(2.990557) + 0.3(7.005610)}{7} = -2.499625$$

$$x_3 = \frac{71.4 - 0.3(2.990557) + 0.2(-2.499625)}{10} = 7.000291$$

Error estimation

$$\varepsilon_{a,1} = \left|\frac{2.990557 - 2.616667}{2.990557}\right| \times 100\% = 12.5\% \qquad \varepsilon_{a,2} = 11.8\% \qquad \varepsilon_{a,3} = 0.076\%$$

Gauss-Seidel vs. Jacobi iterative methods



Diagonal dominance

- The Gauss-Seidel method may diverge
- If the system is *diagonally dominant*, it will definitely converge.
- Diagonal dominance

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

M-file to implement Gauss-Seidel

$$x_1^{\text{new}} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2^{\text{old}} - \frac{a_{13}}{a_{11}}x_3^{\text{old}}$$

$$x_2^{\text{new}} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1^{\text{new}} - \frac{a_{23}}{a_{22}}x_3^{\text{old}}$$

$$x_3^{\text{new}} = \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}}x_1^{\text{new}} - \frac{a_{32}}{a_{33}}x_2^{\text{new}}$$

Implemented by matrix operations

$$\{x\} = \{d\} - [C]\{x\}$$

where

$$\{d\} = \begin{Bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ b_3/a_{33} \end{Bmatrix} \qquad [C] = \begin{bmatrix} 0 & a_{12}/a_{11} & a_{13}/a_{11} \\ a_{21}/a_{22} & 0 & a_{23}/a_{22} \\ a_{31}/a_{33} & a_{32}/a_{33} & 0 \end{bmatrix}$$

M-file to implement Gauss-Seidel (cont.)

```
function x = GaussSeidel(A,b,es,maxit)
  if nargin<4|isempty(maxit), maxit=50; end
  if nargin<3|isempty(es), es=0.00001; end
  C = A;
  n=length(A);
  for i = 1:n
     C(i,i) = 0;
     x(i) = 0;
  end
  x = x';
  for i = 1:n
    C(i,1:n) = C(i,1:n)/A(i,i);
  end
  for i = 1:n
    d(i) = b(i)/A(i,i);
  end
```

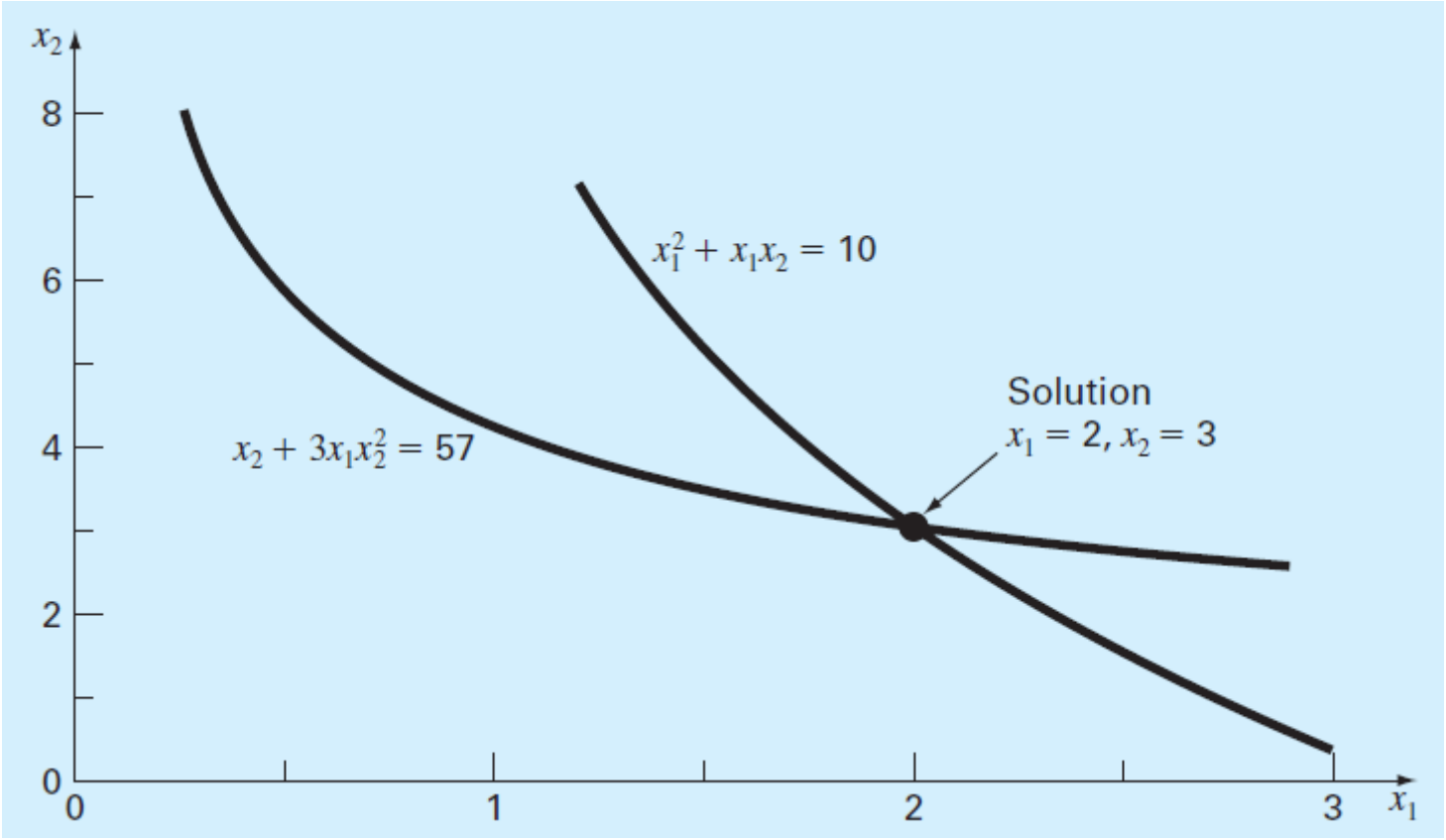
M-file to implement Gauss-Seidel (cont.)

```
iter = 0;
while (1)
  xold = x;
  for i = 1:n
    x(i) = d(i)-C(i,:)*x;
    if x(i) ~= 0
      ea(i) = abs((x(i) - xold(i))/x(i)) * 100;
    end
  end
  iter = iter+1;
  if max(ea)<=es | iter >= maxit, break, end
end
```

Nonlinear system

$$x_1^2 + x_1 x_2 = 10$$

$$x_2 + 3x_1 x_2^2 = 57$$



Solving nonlinear systems by successive substitution, the same strategy as the Gauss-Seidel method

$$x_1^2 + x_1 x_2 = 10$$

$$x_2 + 3x_1 x_2^2 = 57$$

Iterative equations

$$x_1 = \frac{10 - x_1^2}{x_2}$$

$$x_2 = 57 - 3x_1 x_2^2$$

Initial guesses: $x_1 = 1.5, x_2 = 3.5$

$$x_1 = \frac{10 - (1.5)^2}{3.5} = 2.21429$$

$$x_2 = 57 - 3(2.21429)(3.5)^2 = -24.37516$$

Next iteration

$$x_1 = \frac{10 - (2.21429)^2}{-24.37516} = -0.20910$$

$$x_2 = 57 - 3(-0.20910)(-24.37516)^2 = 429.709$$

Diverging !!

Alternative format of iteration equation

$$x_1^2 + x_1 x_2 = 10$$

$$x_2 + 3x_1 x_2^2 = 57$$

Iterative equations

$$x_1 = \sqrt{10 - x_1 x_2}$$

$$x_2 = \sqrt{\frac{57 - x_2}{3x_1}}$$

Initial guesses: $x_1 = 1.5, x_2 = 3.5$

$$x_1 = \sqrt{10 - 1.5(3.5)} = 2.17945$$

$$x_2 = \sqrt{\frac{57 - 3.5}{3(2.17945)}} = 2.86051$$

Next iteration

$$x_1 = \sqrt{10 - 2.17945(2.86051)} = 1.94053$$

$$x_2 = \sqrt{\frac{57 - 2.86051}{3(1.94053)}} = 3.04955$$

Converging !!

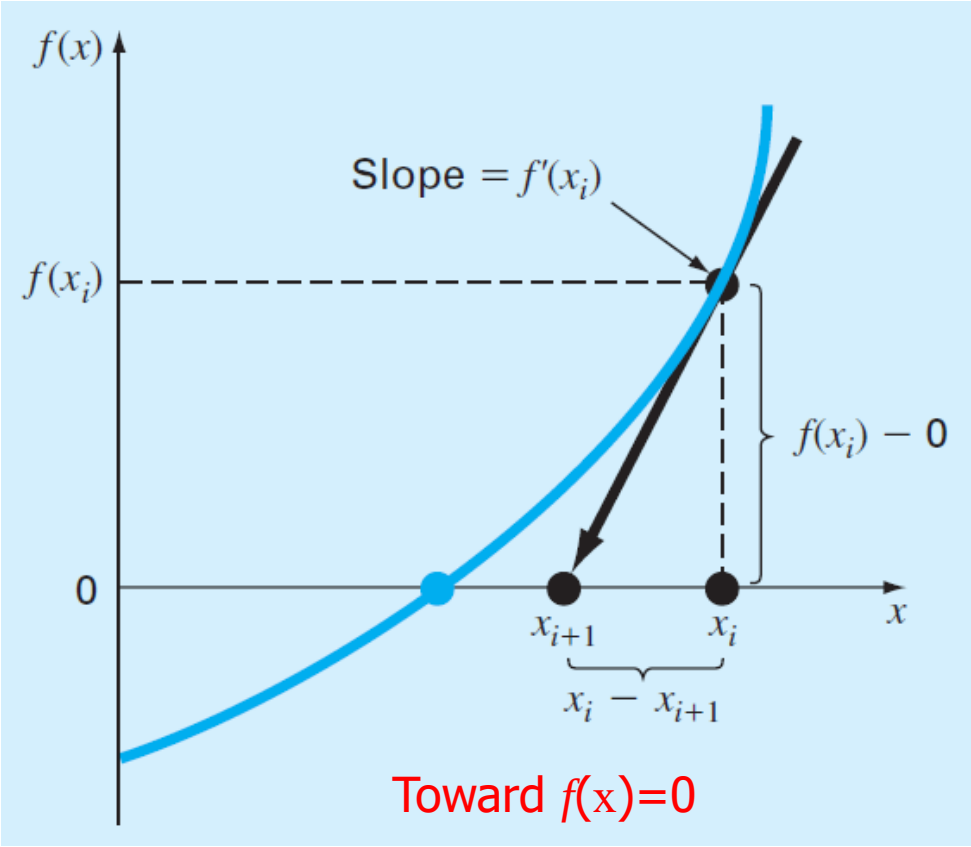
Shortcoming of successive substitution: convergence often depends on the format of iteration equations.

Reform of nonlinear equation solving by root finding of nonlinear equations

$$
\begin{aligned}
f_1(x_1, x_2, \ldots, x_n) &= 0 \\
f_2(x_1, x_2, \ldots, x_n) &= 0 \\
&\vdots \\
f_n(x_1, x_2, \ldots, x_n) &= 0
\end{aligned}
$$

Recall of Newton-Raphson method for finding a root

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i) f'(x_i)$$



$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Solving nonlinear systems by Newton-Raphson

- A first-order Taylor series

$$f_{1,i+1} = f_{1,i} + (x_{1,i+1} - x_{1,i})\frac{\partial f_{1,i}}{\partial x_1} + (x_{2,i+1} - x_{2,i})\frac{\partial f_{1,i}}{\partial x_2}$$

$$f_{2,i+1} = f_{2,i} + (x_{1,i+1} - x_{1,i})\frac{\partial f_{2,i}}{\partial x_1} + (x_{2,i+1} - x_{2,i})\frac{\partial f_{2,i}}{\partial x_2}$$

$f_{1,i+1}$ and $f_{2,i+1}$ equal zero

$$\frac{\partial f_{1,i}}{\partial x_1}x_{1,i+1} + \frac{\partial f_{1,i}}{\partial x_2}x_{2,i+1} = -f_{1,i} + x_{1,i}\frac{\partial f_{1,i}}{\partial x_1} + x_{2,i}\frac{\partial f_{1,i}}{\partial x_2}$$

$$\frac{\partial f_{2,i}}{\partial x_1}x_{1,i+1} + \frac{\partial f_{2,i}}{\partial x_2}x_{2,i+1} = -f_{2,i} + x_{1,i}\frac{\partial f_{2,i}}{\partial x_1} + x_{2,i}\frac{\partial f_{2,i}}{\partial x_2}$$

J, Jacobian matrix

Solution

$$[J]\{x_{i+1}\} = -\{f\} + [J]\{x_i\}$$

$$\{x_{i+1}\} = \{x_i\} - [J]^{-1}\{f\}$$

Nonlinear systems by Newton-Raphson (cont.)

Extending to *n* nonlinear equations

$$\frac{\partial f_{k,i}}{\partial x_1}x_{1,i+1} + \frac{\partial f_{k,i}}{\partial x_2}x_{2,i+1} + \cdots + \frac{\partial f_{k,i}}{\partial x_n}x_{n,i+1} = -f_{k,i} + x_{1,i}\frac{\partial f_{k,i}}{\partial x_1} + x_{2,i}\frac{\partial f_{k,i}}{\partial x_2} + \cdots + x_{n,i}\frac{\partial f_{k,i}}{\partial x_n}$$

Jacobian matrix

$$[J] = \begin{bmatrix} \frac{\partial f_{1,i}}{\partial x_1} & \frac{\partial f_{1,i}}{\partial x_2} & \cdots & \frac{\partial f_{1,i}}{\partial x_n} \\ \frac{\partial f_{2,i}}{\partial x_1} & \frac{\partial f_{2,i}}{\partial x_2} & \cdots & \frac{\partial f_{2,i}}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_{n,i}}{\partial x_1} & \frac{\partial f_{n,i}}{\partial x_2} & \cdots & \frac{\partial f_{n,i}}{\partial x_n} \end{bmatrix}$$

$$\{x_i\}^T = \lfloor x_{1,i} \quad x_{2,i} \quad \cdots \quad x_{n,i} \rfloor$$

$$\{x_{i+1}\}^T = \lfloor x_{1,i+1} \quad x_{2,i+1} \quad \cdots \quad x_{n,i+1} \rfloor$$

$$\{f\}^T = \lfloor f_{1,i} \quad f_{2,i} \quad \cdots \quad f_{n,i} \rfloor$$

$$[J]\{x_{i+1}\} = -\{f\} + [J]\{x_i\} \quad \Rightarrow \quad \{x_{i+1}\} = \{x_i\} - [J]^{-1}\{f\}$$

M-file to implement Newton-Raphson method for nonlinear systems of equations

```
function [x,f,ea,iter]=newtmult(func,x0,es,maxit)
  if nargin<3|isempty(es), es=0.0001; end
  if nargin<4|isempty(maxit), maxit=50; end
  iter = 0;
  x=x0;
  while (1)
    [J,f]=func(x);
    dx=J\f;
    x=x-dx;
    iter = iter + 1;
    ea=100*max(abs(dx./x));
    if iter>=maxit|ea<=es, break, end
  end
```

Example of nonlinear equations

- Determining the roots of two nonlinear equations:

$$f_1(x_1, x_2) = \frac{5 + x_1 + x_2}{(50 - 2x_1 - x_2)^2(20 - x_1)} - 4 \times 10^{-4}$$

$$f_2(x_1, x_2) = \frac{(5 + x_1 + x_2)}{(50 - 2x_1 - x_2)(10 - x_2)} - 3.7 \times 10^{-2}$$

The partial derivatives comprising the Jacobian

$$\frac{\partial f_1}{\partial x_1} = \frac{f_1(x_1 + \delta x_1, x_2) - f_1(x_1, x_2)}{\delta x_1} \qquad \frac{\partial f_1}{\partial x_2} = \frac{f_1(x_1, x_2 + \delta x_2) - f_1(x_1, x_2)}{\delta x_2}$$

$$\frac{\partial f_2}{\partial x_1} = \frac{f_2(x_1 + \delta x_1, x_2) - f_2(x_1, x_2)}{\delta x_1} \qquad \frac{\partial f_2}{\partial x_2} = \frac{f_2(x_1, x_2 + \delta x_2) - f_2(x_1, x_2)}{\delta x_2}$$

M-file to compute function values and Jacobian

```
function [J,f]=jf_2eq(x)
del=0.000001;
df1dx1=(u(x(1)+del*x(1),x(2))-u(x(1),x(2)))/(del*x(1));
df1dx2=(u(x(1),x(2)+del*x(2))-u(x(1),x(2)))/(del*x(2));
df2dx1=(v(x(1)+del*x(1),x(2))-v(x(1),x(2)))/(del*x(1));
df2dx2=(v(x(1),x(2)+del*x(2))-v(x(1),x(2)))/(del*x(2));
J=[df1dx1 df1dx2;df2dx1 df2dx2];
f1=u(x(1),x(2));
f2=v(x(1),x(2));
f=[f1;f2];

function f=u(x,y)
f = (5 + x + y) / (50 - 2 * x - y) ^ 2 / (20 - x) - 0.0004;

function f=v(x,y)
f = (5 + x + y) / (50 - 2 * x - y) / (10 - y) - 0.037;
```

Example of nonlinear equations (cont.)

Employ the Newton-Raphson method to derive the x_1 and x_2

```
format short e
x0=[3; 3];
[x,f,ea,iter]=newtmult(@jf_2eq,x0)
```

After 4 iterations, a solution of x_1=3.3366 and x_2=2.6772 is obtained.

Reference

- Steven C. Chapra "Applied Numerical Methods with MATLA B", 3rd ed., McGraw Hill, 2012.