Lecture Note-Numerical Analysis (4): Roots of Nonlinear Algebraic Equations

1. Background information on the numerical approximation of the derivative of a function

FDM: Finite Difference Method (유한차분법)을 이용한 도함수 계산

- **Taylor series expansion of f(x+h) for a small value h around a given point x**

$$f(x+h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f^{(3)}(x)h^3 + O(h^4)$$

$$f(x-h) \approx f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 - \frac{1}{6}f^{(3)}(x)h^3 + O(h^4)$$

- **The 1st order approximation of the derivative of f(x) using one of the above equation**

$$f'(x) \approx \frac{1}{h}\{f(x+h) - f(x)\} + \frac{1}{2}f''(x)h + \frac{1}{6}f^{(3)}(x)h^2 + O(h^3) \approx \frac{1}{h}\{f(x+h) - f(x)\} + O(h)$$

or

$$f'(x) \approx \frac{1}{h}\{f(x) - f(x-h)\} + \frac{1}{2}f''(x)h - \frac{1}{6}f^{(3)}(x)h^2 + O(h^3) \approx \frac{1}{h}\{f(x+h) - f(x)\} + O(h)$$

Therefore, the first order numerical approximation becomes

$$\boxed{f'(x) \approx \frac{1}{h}\{f(x+h) - f(x)\}} \quad \text{or} \quad \boxed{f'(x) \approx \frac{1}{h}\{f(x) - f(x-h)\}}$$

,which is called by the forward/backward difference formula

○ **The 2nd order approximation of the derivative of f(x) by subtracting the above equation**

$$f(x+h) - f(x-h) \approx 2f'(x)h + \frac{2}{6} f^{(3)}(x)h^3 + O(h^4)$$

$$f'(x) \approx \frac{1}{2h}\{f(x+h) - f(x-h)\} - \frac{1}{6} f^{(3)}(x)h^2 + O(h^3) \approx \frac{1}{2h}\{f(x+h) - f(x-h)\} + O(h^2)$$

Therefore, the 2nd order numerical approximation becomes

$$\boxed{f'(x) \approx \frac{1}{2h}\{f(x+h) - f(x-h)\}}$$

,which is called by the central difference formula

(Example) calculate $f'(1)$ for $f(x) = x^5$ with varying h : True value is $f'(1) = 5.0$

h	$f'(1)$ with 1st order	$f'(1)$ with 2nd order
1	31	16
0.5	13.1875	7.5625
0.25	8.20703125	5.62890625
0.125	6.416259766	5.156494141
0.0625	5.665298462	5.039077759
0.03125	5.322419167	5.009766579
0.015625	5.158710539	5.002441466
0.0078125	5.078737739	5.000610355
0.00390625	5.039215386	5.000152588
0.001953125	5.019569434	5.000038147

2. Definition of Jacobian and numerical approximation of Jacobian

Multi-variable function: $\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} \in R^n, \quad \mathbf{x} \in R^m$

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

Example: $\mathbf{f} \in R^2, \quad \mathbf{x} \in R^3$

$$\mathbf{f}(x, y, z) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} x^2 + y^2 + z^2 + 2xy - 3yz \\ 3x - 2y + 5z \end{pmatrix}$$

Definition of the Jacobian of the multi-variable function:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_m} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_m} \end{pmatrix}$$

Example:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \frac{\partial f_1(\mathbf{x})}{\partial x_3} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \frac{\partial f_2(\mathbf{x})}{\partial x_3} \end{pmatrix}$$

$$= \begin{pmatrix} 2x + 2y & 2y + 2x - 3z & 2z - 3y \\ 3 & -2 & 5 \end{pmatrix}$$

Numerical approximation of the Jacobian using the finite difference formula

$$\frac{\partial f_k(\mathbf{x})}{\partial x_j} \approx \frac{f_k(x_1, \cdots, x_{j-1}, x_j + \Delta x_j, x_{j+1}, \cdots, x_m) - f_k(x_1, \cdots, x_{j-1}, x_j - \Delta x_j, x_{j+1}, \cdots, x_m)}{2\Delta x_j}$$

Remark: The formula has the perturbed values only for the x_j with $x_j \pm \Delta x_j$

(Example) Jacobian computing using the central difference formula

$$\mathbf{f}(x,y,z)=\begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{pmatrix}=\begin{pmatrix} x^2+y^2+z^2+2xy-3yz \\ 3x-2y+5z \end{pmatrix} \text{ at } x=y=z=1 \text{ with } \Delta x=\Delta y=\Delta z=0.1$$

(i) Perturbation in x

Positive perturbation with $x=1.1, y=z=1$

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p=\begin{pmatrix} x^2+y^2+z^2+2xy-3yz \\ 3x-2y+5z \end{pmatrix}=\begin{pmatrix} 2.4100 \\ 6.3000 \end{pmatrix}$$

Negative perturbation with $x=0.9, y=z=1$

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_n=\begin{pmatrix} x^2+y^2+z^2+2xy-3yz \\ 3x-2y+5z \end{pmatrix}=\begin{pmatrix} 1.6100 \\ 5.7000 \end{pmatrix}$$

Jacobian component due to variable x

$$\frac{\partial}{\partial x}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\approx\frac{1}{2\times 0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p-\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_n\right\}=\frac{1}{2\times 0.1}\left\{\begin{pmatrix} 2.4100 \\ 6.3000 \end{pmatrix}-\begin{pmatrix} 1.6100 \\ 5.7000 \end{pmatrix}\right\}=\frac{1}{2\times 0.1}\begin{pmatrix} 0.8000 \\ 0.6000 \end{pmatrix}=\begin{pmatrix} 4.0000 \\ 3.0000 \end{pmatrix}$$

(ii) Perturbation in y with the fixed values of $x=z=1$

Jacobian component due to variable y

$$\frac{\partial}{\partial y}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\approx\frac{1}{2\times 0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p-\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_n\right\}=\frac{1}{2\times 0.1}\left\{\begin{pmatrix} 2.1100 \\ 5.8000 \end{pmatrix}-\begin{pmatrix} 1.9100 \\ 6.2000 \end{pmatrix}\right\}=\begin{pmatrix} 1.0000 \\ -2.0000 \end{pmatrix}$$

(iii) Perturbation in z with the fixed values of $x = y = 1$

Jacobian component due to variable y

$$\frac{\partial}{\partial y}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \approx \frac{1}{2\times 0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p - \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_n\right\} = \frac{1}{2\times 0.1}\left\{\begin{pmatrix} 1.9100 \\ 6.5000 \end{pmatrix} - \begin{pmatrix} 2.1100 \\ 5.5000 \end{pmatrix}\right\} = \begin{pmatrix} -1.0000 \\ 5.0000 \end{pmatrix}$$

(iv) Approximated Jacobian

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} 4 & 1 & -1 \\ 3 & -2 & 5 \end{pmatrix}$$

(v) Exact Jacobian

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} 2x+2y & 2y+2x-3z & 2z-3y \\ 3 & -2 & 5 \end{pmatrix} = \begin{pmatrix} 4 & 1 & -1 \\ 3 & -2 & 5 \end{pmatrix}$$

(Example) Jacobian computing using the forward difference formula

$$\mathbf{f}(x,y,z) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} x^2 + y^2 + z^2 + 2xy - 3yz \\ 3x - 2y + 5z \end{pmatrix} \text{ at } x = y = z = 1 \text{ with } \Delta x = \Delta y = \Delta z = 0.1$$

$$\mathbf{f}(1,1,1) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$$

(Example): $\mathbf{f}(x,y,z)=\begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{pmatrix}=\begin{pmatrix} x^2+y^2+z^2+2xy-3yz \\ 3x-2y+5z \end{pmatrix}$ at $x=y=z=1$ with $\Delta x=\Delta y=\Delta z=0.1$

(i) Perturbation in x

Jacobian component due to variable x

$$\frac{\partial}{\partial x}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\approx\frac{1}{0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p-\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\right\}=\frac{1}{0.1}\left\{\begin{pmatrix} 2.4100 \\ 6.3000 \end{pmatrix}-\begin{pmatrix} 2 \\ 6 \end{pmatrix}\right\}=\begin{pmatrix} 5.0 \\ 3.0 \end{pmatrix}$$

(ii) Perturbation in y with the fixed values of $x=z=1$

$$\frac{\partial}{\partial y}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\approx\frac{1}{0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p-\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\right\}=\frac{1}{0.1}\left\{\begin{pmatrix} 2.1100 \\ 5.8000 \end{pmatrix}-\begin{pmatrix} 2 \\ 6 \end{pmatrix}\right\}=\begin{pmatrix} 1.1 \\ -2.0 \end{pmatrix}$$

(iii) Perturbation in z with the fixed values of $x=y=1$

Jacobian component due to variable y

$$\frac{\partial}{\partial y}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\approx\frac{1}{0.1}\left\{\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}_p-\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\right\}=\frac{1}{0.1}\left\{\begin{pmatrix} 1.9100 \\ 6.5000 \end{pmatrix}-\begin{pmatrix} 2 \\ 6 \end{pmatrix}\right\}=\begin{pmatrix} -0.9 \\ 5.0 \end{pmatrix}$$

(iv) Approximated Jacobian

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}=\begin{pmatrix} 5.0 & 1.1 & -0.9 \\ 3.0 & -2.0 & 5.0 \end{pmatrix}$$

(v) Exact Jacobian

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} 2x+2y & 2y+2x-3z & 2z-3y \\ 3 & -2 & 5 \end{pmatrix} = \begin{pmatrix} 4 & 1 & -1 \\ 3 & -2 & 5 \end{pmatrix}$$

Tips

1) **Use a small value for the perturbations** $\Delta x, \Delta y, \Delta z$
2) **Use the 2nd order central difference formula to enhance accuracy**
3) **Use the 1st order difference formula to save computing time with small values for the perturbations**

3. Taylor series expansion of the multi-variable functions

(3-1) Two-variable scalar function $f(x,y) \in R$

$$f(x+h_x, y+h_y) \approx f(x,y) + \frac{\partial f}{\partial x} h_x + \frac{\partial f}{\partial y} h_y + \frac{1}{2}\frac{\partial^2 f}{\partial x^2} h_x^2 + \frac{\partial^2 f}{\partial x \partial y} h_x h_y + \frac{1}{2}\frac{\partial^2 f}{\partial y^2} h_y^2 + O(h^3)$$

(3-2) Two-variable vector function $\mathbf{f}(x,y) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} \in R^2$

$$\mathbf{f}(x+h_x, y+h_y) = \begin{pmatrix} f_1(x+h_x, y+h_y) \\ f_2(x+h_x, y+h_y) \end{pmatrix}$$

$$\approx \begin{pmatrix} f_1(x,y) + \dfrac{\partial f_1}{\partial x} h_x + \dfrac{\partial f_1}{\partial y} h_y \\ f_2(x,y) + \dfrac{\partial f_2}{\partial x} h_x + \dfrac{\partial f_2}{\partial y} h_y \end{pmatrix} = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} + \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{pmatrix} \begin{pmatrix} h_x \\ h_y \end{pmatrix}$$

$$= \mathbf{f}(x,y) + \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \Delta\mathbf{x} \leftarrow \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{pmatrix}, \quad \Delta\mathbf{x} = \begin{pmatrix} h_x \\ h_y \end{pmatrix}$$

(3-3) Three-variable scalar function $f(x,y,z)\in R$

$$f(x+h_x, y+h_y, z+h_z) \approx f(x,y) + \frac{\partial f}{\partial x}h_x + \frac{\partial f}{\partial y}h_y + \frac{\partial f}{\partial z}h_z + O(h^2)$$

(3-4) Three-variable vector function $\mathbf{f}(x,y) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} \in R^2$

$$\mathbf{f}(x+h_x, y+h_y, z+h_z) = \begin{pmatrix} f_1(x+h_x, y+h_y, z+h_z) \\ f_2(x+h_x, y+h_y, z+h_z) \end{pmatrix}$$

$$\approx \begin{pmatrix} f_1(x,y,z) + \dfrac{\partial f_1}{\partial x}h_x + \dfrac{\partial f_1}{\partial y}h_y + \dfrac{\partial f_1}{\partial z}h_z \\ f_2(x,y,z) + \dfrac{\partial f_2}{\partial x}h_x + \dfrac{\partial f_2}{\partial y}h_y + \dfrac{\partial f_2}{\partial z}h_z \end{pmatrix}$$

$$= \begin{pmatrix} f_1(x,y,z) \\ f_2(x,y,z) \end{pmatrix} + \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} \end{pmatrix} \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix}$$

$$= \mathbf{f}(x,y,z) + \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \Delta\mathbf{x} \leftarrow \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial z} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial z} \end{pmatrix}, \quad \Delta\mathbf{x} = \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix}$$

4. Newton-Raphson Method: One of the most popular iterative method

- The 1st order Taylor series approximation of a function can be written as

$$f(x_{j+1}) \approx f(x_j) + f'(x)(x_{j+1} - x_j)$$

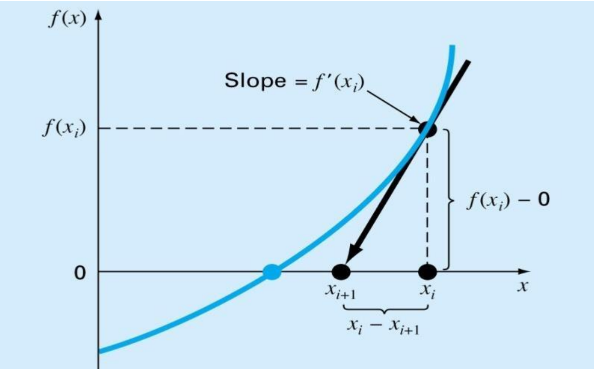
The Newton-Raphson method approximate the root with x_{j+1} satisfying $f(x_{j+1}) \approx f(x_j) + f'(x)(x_{j+1} - x_j) = 0$

Therefore,

$$\rightarrow \; f(x_j) + f'(x)(x_{j+1} - x_j) = 0$$

$$\rightarrow \; x_{j+1} - x_j = -\frac{f(x_j)}{f'(x_j)} \rightarrow \; x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}$$

- Graphical depiction of the Newton-Raphson method



```
Function NEWTON(x, ITmax, h,epsilon)
!Pseudo code for Newton_Raphson method to find x satisfying f(x)=0 with a given initial point x
!Given variables: x(initial-value), h(small increment to calculate derivative)
!                             ITmax (number of maximum iteration allowed)
! Given tolerance: epsilon <<1
! Given external function: f(x)
! Using numerical calculation of derivative information of the function f(x)
        Do j=1,2,3, …….,ITmax
! estimation of derivative using the central difference formula
!
                fzero = f(x)  xpl                   !function value at zero perturbation
                us = x+h  fplus                    ! positive perturvation
                = f(xplus)                                   ! function value at xplus
!
                xminus = x-h  fmin                 ! negative perturvation
                us = f(xminus)                               ! function value at xminus
!
                gradf = 0.5*(fplus-fminus)/h                 ! derivative(gradient) estimation
```

```
!Newton-Raphson method
                    x0=x
            x ← x – fzero/gradf          ! for the nonlinear system: gradf is a matrix


!convergence test
            If abs(fzero) < epsilon, exit      !converged solution
            If abs(x-x0)<epsilon, exit         !converged solution
 !
        End do  NEW
        TON = x                            !similar to return value in C/C++
!
 End NEWTON
```

5. The Secant Method

- In the Secant method the derivative is approximated using the 1st order finite difference formula with the following increment condition

$$x_j = x_{j-1} + h \rightarrow h = x_j - x_{j-1}$$

$$\boxed{f'(x) \approx \frac{1}{h}\{f(x) - f(x-h)\}} \rightarrow \boxed{f'(x_j) \approx \frac{f(x_j) - f(x_{j-1})}{x_j - x_{j-1}}}$$

Then the Newton-Raphson formula can be represented by the formula for the Secant Method

$$\boxed{\begin{aligned} x_{j+1} &\approx x_j - \frac{f(x_j)}{f'(x_j)} \\ &= x_j - \frac{f(x_j)}{f(x_j) - f(x_{j-1})}(x_j - x_{j-1}) \end{aligned}}$$

- Modified Secant method by directly using the backward difference formula (1st order)

$$\boxed{f'(x) \approx \frac{1}{h}\{f(x) - f(x-h)\}} \rightarrow \boxed{f'(x_j) \approx \frac{f(x_j) - f(x_j - h)}{h}}$$

$$\boxed{\begin{aligned} x_{j+1} &\approx x_j - \frac{f(x_j)}{f'(x_j)} \\ &= x_j - \frac{hf(x_j)}{f(x_j) - f(x_j - h)} \end{aligned}}$$

○ **Definition of open method**

- It needs functional information such as function value and its gradient at one points to find a root:Find x satisfying the nonlinear equation $\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} \in R^n, \quad \mathbf{x} \in R^n$

-Bracketing methods are always convergent. However, the convergence of open methods highly depend on the initial estimation of the root, where function value and its gradient are calculated.

6. Newton-Raphson Method for the System of Nonlinear Equations

- Definition of the system of nonlinear equations

$\boxed{\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} \in R^n, \quad \mathbf{x} \in R^n}$, which has n unknowns $\mathbf{x} \in R^n$ and n nonlinear equations $\mathbf{f} \in R^n$

Expanded form

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \in R^n, \quad \mathbf{f} = \begin{pmatrix} f_1(x_1, x_2, x_3, \cdots, x_n) \\ f_2(x_1, x_2, x_3, \cdots, x_n) \\ f_3(x_1, x_2, x_3, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, x_3, \cdots, x_n) \end{pmatrix} = \mathbf{0} \in R^n$$

- Jacobean of the system of nonlinear equations

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \mathbf{G} = \begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \cdots & \frac{df_1}{dx_n} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \cdots & \frac{df_2}{dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{df_n}{dx_1} & \frac{df_n}{dx_2} & \cdots & \frac{df_n}{dx_n} \end{pmatrix} \in R^{n \times n}$$

- 1^{st} Order approximation of function value around x

$$\begin{aligned}\mathbf{f}(\mathbf{x}+\mathbf{h}) &\approx \mathbf{f}(\mathbf{x}) + \frac{d\mathbf{f}}{d\mathbf{x}}\mathbf{h} \\ &\approx \mathbf{f}(\mathbf{x}) + \mathbf{G}\mathbf{h}\end{aligned}$$

- Newton-Raphson Method for the system of Nonlinear Equations

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \mathbf{h}$$

The Newton-Raphson Method approximate the root with the vector satisfying

$$\begin{aligned}\mathbf{f}(\mathbf{x}_{j+1}) &\approx \mathbf{f}(\mathbf{x}_j) + \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_{j+1} - \mathbf{x}_j) \\ &\approx \mathbf{f}(\mathbf{x}_j) + \mathbf{G}(\mathbf{x}_{j+1} - \mathbf{x}_j)\end{aligned} \quad \rightarrow \quad \begin{aligned}&\mathbf{f}(\mathbf{x}_j) + \mathbf{G}(\mathbf{x}_{j+1} - \mathbf{x}_j) = 0 \\ &\rightarrow \mathbf{G}\mathbf{x}_{j+1} = \mathbf{G}\mathbf{x}_j - \mathbf{f}(\mathbf{x}_j) \\ &\rightarrow \mathbf{x}_{j+1} = \mathbf{G}^{-1}(\mathbf{G}\mathbf{x}_j - \mathbf{f}(\mathbf{x}_j)) \\ &\rightarrow \mathbf{x}_{j+1} = \mathbf{x}_j - \mathbf{G}^{-1}\mathbf{f}(\mathbf{x}_j)\end{aligned}$$

(Example) Newton-Raphson Method for the system of Nonlinear Equations

$$f_1(x,y) = x_2 + xy - 10 = 0$$

$$f_2(x,y) = 3xy_2 + y - 57 = 0$$

$$\frac{\partial f_1(x,y)}{\partial x} = 2x + y \qquad \frac{\partial f_1(x,y)}{\partial y} = x$$

$$\frac{\partial f_2(x,y)}{\partial x} = 3y^2 \qquad \frac{\partial f_2(x,y)}{\partial y} = 6xy + 1$$

$$\rightarrow \boxed{\begin{array}{l} \mathbf{G} = \begin{pmatrix} 2x+y & x \\ 3y^2 & 6xy+1 \end{pmatrix} \\ \mathbf{G}^{-1} = \dfrac{1}{(2x+y)(6xy+1) - 3xy^2} \begin{pmatrix} 6xy+1 & -x \\ -3y^2 & 2x+y \end{pmatrix} \end{array}} \quad \text{for } \mathbf{x} = \begin{pmatrix} x_j \\ y_j \end{pmatrix}$$

By using $\boxed{\mathbf{x}_{j+1} = \mathbf{x}_j - \mathbf{G}^{-1}\mathbf{f}(\mathbf{x}_j)}$

$$\boxed{\begin{pmatrix} x_{j+1} \\ y_{j+1} \end{pmatrix} = \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \frac{1}{(2x_j + y_j)(6x_j y_j + 1) - 3x_j y_j^2} \begin{pmatrix} 6x_j y_j + 1 & -x_j \\ -3y_j^2 & 2x_j + y_j \end{pmatrix} \begin{pmatrix} x_j^2 + x_j y_j - 10 \\ 3x_j y_j^2 + y_j - 57 \end{pmatrix}}$$

```
Function NEWTON_SYS(n, x,    ITmax,    h, epsilon, error)
!-------------------------------------------------------------------------------------------
!Pseudo code for Newton_Raphson method to find x satisfying f(x)=0 with a given initial point x
!Given variables:
!    n        : (input) number of equations
!    x(1:n): (input/output) initial-value
!    ITmax:(input) maximum allowed iteration
!    h(1:n): (input) small increments to calculate derivative
!    epsilon: (input) given tolerance (<<1 )
!    error:(output) norm of function residual
!Be careful when we calculate the roots of nonlinear system of equations
!      x(1:n), h(1:n), f(1:n), gradf(1:n, 1:n)
!-------------------------------------------------------------------------------------------
        Do j=1,2,3, …….,ITmax
                fzero(1:n) = f(x)                            !function value at zero perturbation
! estimation of derivative using the central difference formula
                do k = 1, n
                    xplus(1:n) = x(1:n)                      ! positive perturvation
                    xplus(k)     = xplus(k) + h(k)
                    fplus(1:n)    = f(xplus)                 ! function value at xplus
!
                    xminus (1:n) = x(1:n)                    ! negative perturvation
```

```
                xminus (k)      = xminus (k) - h(k)
                fminus (1:n)    = f(xminus)              ! function value at xminus
!
                gradf (1:n,k)= 0.5*(fplus(1:n)-fminus(1:n))/h(k) ! derivative(gradient) estimation
            end do
!Newton-Raphson method
                 x0(1:n)=x(1:n)
            x(1:n) ← x(1:n) – (gradf)^-1 *fzero(1:n)       ! for the nonlinear system: gradf is a matrix
!convergence test
            If norm(fzero) < epsilon, exit         !converged solution
            If norm(x-x0)<epsilon, exit            !converged solution
        End do
!
        error = norm(f(x)) ! residual in function value
!
 End NEWTON_SYS
```

Appendix: Problem set for the System of Nonlinear Equations

General Problem Statements

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{f}, \mathbf{x} \in R^n \rightarrow \mathbf{f} = \begin{pmatrix} f_1(x_1, x_2, \cdots, x_n) \\ f_2(x_1, x_2, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

1. Problem #1

$$\mathbf{f} = \begin{pmatrix} x_1 + 3\ln(x_1) - x_2^2 \\ 2x_1^2 - x_1x_2 - 5x_1 + 1.0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 1.0 \\ -2.0 \end{pmatrix}$$

2. Problem #2

$$\mathbf{f} = \begin{pmatrix} x_1^2 + x_1x_2^2 - 9 \\ 3x_1^2x_2 - x_2^3 - 4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 1.2 \\ 2.5 \end{pmatrix}$$

3. Problem #3

$$\mathbf{f} = \begin{pmatrix} x_1 + 2x_2 - 3.0 \\ 2x_1^2 + x_2^2 - 5.0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix}$$

4. **Problem #4**

$$\mathbf{f} = \begin{pmatrix} 3x_1^2 + 4x_2^2 - 1.0 \\ x_2^3 - 8x_1^3 - 1.0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} -0.5 \\ 0.25 \end{pmatrix}$$

5. **Problem #5**

$$\mathbf{f} = \begin{pmatrix} 4x_1^2 + x_2^2 - 4.0 \\ x_1 + x_2 - \sin(x_1 - x_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix}$$

6. **Problem #6 : Not converged**

$$\mathbf{f} = \begin{pmatrix} x_1^5 + x_2^3 x_3^4 + 1.0 \\ x_1^2 x_2 x_3 \\ x_3^4 - 1.0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} -10 \\ -10 \\ -10 \end{pmatrix}$$

7. **Problem #7**

$$\mathbf{f} = \begin{pmatrix} x_1^2 + x_2 - 37 \\ x_1 - x_2^2 - 5.0 \\ x_1 + x_2 + x_3 - 3.0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 5.0 \\ 0.0 \\ -2.0 \end{pmatrix}$$

8. Problem #8

$$\mathbf{f} = \begin{pmatrix} 12x_1 - 3x_2^2 - 4x_3 - 7.17 \\ x_1^2 + 10x_2 - x_3 - 11.54 \\ x_2^3 + 7x_3 - 7.631 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 3.0 \\ 0.0 \\ 1.0 \end{pmatrix}$$

9. Problem #9

$$\mathbf{f} = \begin{pmatrix} 10x_2 - 10x_1^2 \\ 1 - x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} -1.2 \\ 1.0 \end{pmatrix}$$

10. Problem #10: Not converged

$$\mathbf{f} = \begin{pmatrix} -13.0 + x_1 - x_2^3 + 5x_2^2 - 2x_2 \\ -29.0 + x_1 + x_2^3 + x_2^2 - 14x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 15.0 \\ 1.0 \end{pmatrix}$$

11. Problem #11

$$\mathbf{f} = \begin{pmatrix} 10.0x_2 - 10x_1^2 \\ 1.0 - x_1 \\ 10.0x_4 - 10x_3^2 \\ 1.0 - x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

12.Problem #12 : Poor convergence

$$\mathbf{f} = \begin{pmatrix} x_1 + 10x_2 \\ \sqrt{5}x_3 - \sqrt{5}x_4 \\ (x_2 - 2x_3)^2 \\ \sqrt{10}(x_1 - x_4) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

13.Problem #13 : Poor convergence

$$\mathbf{f} = \begin{pmatrix} x_1^2 - x_2 - 1 \\ x_2^2 - x_1 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

14.Problem #14 : Poor convergence

$$\mathbf{f} = \begin{pmatrix} x_1 - x_2^2 \\ (x_2 - 1)^2 (x_2 - 2)^2 + (x_1 - x_2^2)^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

```
!----------------------------------------------------------------
!   Problem Set for Nonlinear Algebraic Equations
!----------------------------------------------------------------
      SUBROUTINE NAE_Problems(IND_PROBLEM,IND_case,No_variable,X,Fun,No_fun_call);
!--------------------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
!--------------------------------------------------------------------------
!     Input:
!               IND_PROBLEM: Problem Number
!               IND_Case
!                         = 1: initialization routine
!                         = 2: evaluation of function vectors
!--------------------------------------------------------------------------
      DIMENSION X(*),Fun(*)
!----------------------------------------------------------------
    IF(IND_case==2) No_fun_call = No_fun_call + 1
!----------------------------------------------------------------
      NSELECT = 0
      SELECT CASE (IND_PROBLEM)
!----------------------------------------------------------------
            CASE (1) ;
!----------------------------------------------------------------
                 IF(IND_Case==1) THEN
                       No_variable = 2;
!
                       X(1) =    1.0
                       X(2) = -2.0
                       NSELECT = 1;
!
 ----------------------------------------------------------------
                 ELSEIF(IND_Case==2) THEN
                       Fun(1) = X(1) + 3.0*LOG(X(1))-X(2)*X(2)
                       Fun(2) = 2*X(1)*X(1) - X(1)*X(2) - 5.0*X(1) + 1.0
                       NSELECT = 1;
                 END IF
!
 ----------------------------------------------------------------
```

```
!-----------------------------------------------------------------
          CASE (2) ;
!-----------------------------------------------------------------
              IF(IND_Case==1) THEN
                   No_variable = 2;
!
                   X(1) =    1.2
                   X(2) =    2.5
                   NSELECT = 1;
!-----------------------------------------------------------------
              ELSEIF(IND_Case==2) THEN
                   Fun(1) = X(1)*X(1) + X(1)*X(2)*X(2) -9.0
                   Fun(2) = 3*X(1)*X(1)*X(2) -X(2)**3 - 4.0
                   NSELECT = 1;
              END IF
!-----------------------------------------------------------------


!-----------------------------------------------------------------
          CASE (3) ;
!-----------------------------------------------------------------
              IF(IND_Case==1) THEN
                   No_variable = 2;
!
                   X(1) =    1.5
                   X(2) =    1.0
                   NSELECT = 1;
!-----------------------------------------------------------------
              ELSEIF(IND_Case==2) THEN
                   Fun(1) = X(1) + 2.0*X(2) - 3.0
                   Fun(2) = 2*X(1)*X(1) + X(2)*X(2) - 5.0
```

```
            NSELECT = 1;
        END IF
!------------------------------------------------------------



!------------------------------------------------------------
    CASE (4) ;
!------------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 2;
!
            X(1) = -0.5
            X(2) =    0.25
            NSELECT = 1;
!------------------------------------------------------------
        ELSEIF(IND_Case==2) THEN
            Fun(1) = 3.0*X(1)*X(1) + 4*X(2)*X(2) - 1.0
            Fun(2) = X(2)**3 - 8.0*X(1)**3 - 1.0
            NSELECT = 1;
        END IF
!------------------------------------------------------------



!------------------------------------------------------------
    CASE (5) ;
!------------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 2;
!
            X(1) =    1.0
            X(2) =    0.0
            NSELECT = 1;
!------------------------------------------------------------
```

```
        ELSEIF(IND_Case==2) THEN
            Fun(1) = 4.0*X(1)*X(1) + X(2)*X(2) - 4.0
            Fun(2) = X(1) + X(2) - SIN(X(1)-X(2))
            NSELECT = 1;
        END IF
!-----------------------------------------------------------


!-----------------------------------------------------------
    CASE (6) ;
!-----------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 3;
!
            X(1) = -10.0
            X(2) = -10.0
            X(3) = -10.0
            NSELECT = 1;
!-----------------------------------------------------------
        ELSEIF(IND_Case==2) THEN
            Fun(1) = X(1)**5 + (X(2)**3)*(X(3)**4) + 1.0
            Fun(2) = X(1)*X(1)*X(2)*X(3)
            Fun(3) = X(3)**4 - 1.0  NSELE
            CT = 1;
        END IF
!-----------------------------------------------------------


!-----------------------------------------------------------
    CASE (7) ;
!-----------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 3;
```

```
            X(1) =   5.0
            X(2) =   0.0
            X(3) = -2.0
            NSELECT = 1;
!------------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            Fun(1) = X(1)**2 + X(2) - 37.0
            Fun(2) = X(1) - X(2)*X(2) - 5.0
            Fun(3) = X(1) + X(2) + X(3) - 3.0
            NSELECT = 1;
         END IF
!------------------------------------------------------------



!------------------------------------------------------------
      CASE (8) ;
!------------------------------------------------------------
         IF(IND_Case==1) THEN
            No_variable = 3;
!
            X(1) =   3.0
            X(2) =   0.0
            X(3) =   1.0
            NSELECT = 1;
!------------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            Fun(1) = 12.0*X(1) - 3.0*X(2)**2 - 4.0*X(3) - 7.17
            Fun(2) = X(1)**2 + 10.0*X(2) - X(3) - 11.54
            Fun(3) = X(2)**3 + 7.0*X(3) - 7.631
            NSELECT = 1;
         END IF
!------------------------------------------------------------
```

```
!-----------------------------------------------------------
        CASE (9) ; ! Ref Broyden A Class of Methods for Solving Nonlinear Simultaneous Equations
!-----------------------------------------------------------
            IF(IND_Case==1) THEN
                No_variable = 2;
!
                X(1) = -1.2
                X(2) =    1.0
!
                NSELECT = 1;
!-----------------------------------------------------------
            ELSEIF(IND_Case==2) THEN  Fu
                n(1) = 10.0*(X(2) - X(1)**2) F
                un(2) = 1.0 - X(1)
!
                NSELECT = 1;
            END IF
!-----------------------------------------------------------

!-----------------------------------------------------------
        CASE (10) ; ! Ref Broyden A Class of Methods for Solving Nonlinear Simultaneous Equations
!-----------------------------------------------------------
            IF(IND_Case==1) THEN
                No_variable = 2;
!
                X(1) =    15.0
                X(2) =      1.0
!
                NSELECT = 1;
!-----------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                Fun(1) = -13.0 + X(1) + ((-X(2) + 5.0)*X(2) -    2.0)*X(2)
                Fun(2) = -29.0 + X(1) + (( X(2) + 1.0)*X(2) - 14.0)*X(2)
```

```
!
                NSELECT = 1;
            END IF
!-------------------------------------------------------------

!-------------------------------------------------------------
        CASE (11) ; ! Ref Shanghai Multi-step Nonlinear ABS Methods and Their Efficiency Analysis 1991 (Extended Rosenbloack function)
!-------------------------------------------------------------
            IF(IND_Case==1) THEN
                No_variable = 4;
!
                X(1:4) = 0.5
!
                NSELECT = 1;
!-------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN  Fu
                n(1) = 10.0*(X(2) - X(1)**2) F
                un(2) = 1.0 - X(1)
                Fun(3) = 10.0*(X(4) - X(3)**2)
                Fun(4) = 1.0 - X(3)
!
                NSELECT = 1;
            END IF
!
 -------------------------------------------------------------

!
 -------------------------------------------------------------
        CASE (12) ; ! ! Ref Shanghai Multi-step Nonlinear ABS Methods and Their Efficiency Analysis 1991 (Extended Powell singular function)
!
 -------------------------------------------------------------
            IF(IND_Case==1) THEN
                No_variable = 4;
!
                X(1:4) = 0.5
!
                NSELECT = 1;
!
 -------------------------------------------------------------
```

```
        ELSEIF(IND_Case==2) THEN
            Fun(1) = X(1) + 10.0*X(2)
            Fun(2) = SQRT(5.0)*(X(3) - X(4))
            Fun(3) = (X(2) - 2.0*X(3))**2
            Fun(4) = SQRT(10.0)*(X(1) - X(4))**2
!
            NSELECT = 1;
        END IF
!----------------------------------------------------------


!----------------------------------------------------------
    CASE (13) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #1
!----------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 2;
!
            X(1:2) = 0.5
!
            NSELECT = 1;
!----------------------------------------------------------
        ELSEIF(IND_Case==2) THEN
            Fun(1) = X(1)**2 - X(2)-1.0
            Fun(2) = X(2)**2 - X(1)-1.0
!
            NSELECT = 1;
        END IF
!
 ----------------------------------------------------------


!
 ----------------------------------------------------------
    CASE (14) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #2
!
 ----------------------------------------------------------
        IF(IND_Case==1) THEN
            No_variable = 2;
```

```
!
                    X(1:2) = 0.5
!
                    NSELECT = 1;
!---------------------------------------------------------------
               ELSEIF(IND_Case==2) THEN
                    Fun(1) = X(1) - X(2)**2
                    Fun(2) = ((X(2)-1.0)**2)*((X(2)-2.0)**2) + (X(1)-X(2)**2)**2
!
                    NSELECT = 1;
               END IF
!---------------------------------------------------------------


!---------------------------------------------------------------
          CASE (15) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #3
!---------------------------------------------------------------
               a1 = 25.0;     b1 = 1.0;          c1 = 2.0;
               a2 = 3.0 ;     b2 = 4.0;          c2 = 5.0;
               IF(IND_Case==1) THEN
                    No_variable = 2;
!
                    X(1) =-10.0
                    X(2) = -1.0
!
                    NSELECT = 1;
!---------------------------------------------------------------
               ELSEIF(IND_Case==2) THEN
                    Fun(1) = X(1)**3 -3.0*X(1)*X(2)**2 +a1*(2.0*X(1)**2 +        X(1)*X(2)) + b1*X(2)**2 + c1*X(1) + a2*X(2)
                    Fun(2) =-X(2)**3 +3.0*X(2)*X(1)**2 +a1*(        X(2)**2 -4.0*X(1)*X(2)) + b2*X(1)**2 + c2
!
                    NSELECT = 1;
               END IF
!---------------------------------------------------------------
```

```
!-----------------------------------------------------------
      CASE (16) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #4
!-----------------------------------------------------------
         IF(IND_Case==1) THEN
            No_variable = 3;
!
            X(1:3) = 0.1
!
            NSELECT = 1;
!-----------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            Fun(1) = X(1) + X(2) +X(3) - 3.0
            Fun(2) = X(1)*X(2) + 2.0*X(2)**2 + 4.0*X(3)**2 - 7.0
            Fun(3) = X(1)**8 + X(2)**4 + X(3)**9 - 3.0
!
            NSELECT = 1;
         END IF
!
-----------------------------------------------------------




!-----------------------------------------------------------
!-----------------------------------------------------------
!   Large Scale Problem by Adjusting N the number of equations
!-----------------------------------------------------------
!-----------------------------------------------------------
      CASE (101) ; ! Ref An Autoadatative limited memory Broyden's Method to Solve Systems of NEs :: Broyden banded function
!-----------------------------------------------------------
         N = 10
         IF(IND_Case==1) THEN
            No_variable = N;
!
```

```
                X(1:N) =    0.0
                NSELECT = 1;
!------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                Fun(1) = X(1)*(2.0+5.0*X(1)**2)+1.0 - X(2)*(1.0+X(2))
                Fun(2) = X(2)*(2.0+5.0*X(2)**2)+1.0 - X(1)*(1.0+X(1)) - X(3)*(1.0+X(3))
                Fun(3) = X(3)*(2.0+5.0*X(3)**2)+1.0 - X(1)*(1.0+X(1)) - X(2)*(1.0+X(2)) - X(4)*(1.0+X(4))
                Fun(4) = X(4)*(2.0+5.0*X(4)**2)+1.0 - X(1)*(1.0+X(1)) - X(2)*(1.0+X(2)) - X(3)*(1.0+X(3)) - X(5)*(1.0+X(5))
                Fun(5) = X(5)*(2.0+5.0*X(5)**2)+1.0 - X(1)*(1.0+X(1)) - X(2)*(1.0+X(2)) - X(3)*(1.0+X(3)) - X(4)*(1.0+X(4)) - X(6)*(1.0+X(6))
!
                DO    J = 6, N-1
                    Fun(J) = X(J)*(2.0+5.0*X(J)**2)+1.0
                    DO    K = J-5, J-1
                        Fun(J) = Fun(J) - X(K)*(1.0+X(K))
                    END DO
                    Fun(J) = Fun(J) - X(J+1)*(1.0+X(J+1))
                END DO
!
                Fun(N) = X(N)*(2.0+5.0*X(N)**2)+1.0
                DO    K = N-5, N-1
                        Fun(N) = Fun(N) - X(K)*(1.0+X(K))
                END DO
!
                NSELECT = 1;
            END IF
!
 ------------------------------------------------------------


!------------------------------------------------------------
        CASE (102) ; ! Ref An Autoadatative limited memory Broyden's Method to Solve Systems of NEs :: Martinez function
!------------------------------------------------------------
            N = 10
            IF(IND_Case==1) THEN
                No_variable = N;
!
```

```
                X(1:N) =    0.0
                NSELECT = 1;
!------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                Fun(1) = (3.0 - 0.1*X(1))*X(1) + 1.0 - 2.0*X(2) + X(1)
                DO    J = 2, N-1
                    Fun(J) = (3.0 - 0.1*X(J))*X(J) + 1.0 - X(J-1) - 2.0*X(J+1) + X(J)
                END DO
                Fun(N) = (3.0 - 0.1*X(N))*X(N) + 1.0 - 2.0*X(N-1) + X(N)
!
                NSELECT = 1;
            END IF
!------------------------------------------------------------


!------------------------------------------------------------
        CASE (103) ; ! Ref An Autoadatative limited memory Broyden's Method to Solve Systems of NEs :: Broyden tridiagonal function
!------------------------------------------------------------
            N = 10
            IF(IND_Case==1) THEN
                No_variable = N;
!
                X(1:N) =    0.0
                NSELECT = 1;
!------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                Fun(1) = (3.0 - 2.0*X(1))*X(1) + 1.0 - 2.0*X(2)
                Fun(N) = (3.0 - 2.0*X(N))*X(N) + 1.0 -          X(N-1)
                DO    J = 2, N-1
                    Fun(J) = (3.0 - 2.0*X(J))*X(J) + 1.0 - X(J-1) - 2.0*X(J+1)
                END DO
!
                NSELECT = 1;
            END IF
!
 ------------------------------------------------------------
```

```
!---------------------------------------------------------------
      CASE (104) ; ! Ref An Autoadatative limited memory Broyden's Method to Solve Systems of NEs :: Spedicato function 4
!---------------------------------------------------------------
         N = 10
         IF(IND_Case==1) THEN
            No_variable = N;
!
            X(1:N-1) = -1.2
            X(N)       =   1.0
            NSELECT   =   1;
!---------------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            K = 0
            DO   J = 1, N
               K = K + 1
               IF(K.EQ.1) THEN ; ! Odd case of J F
                  un(J) = 1.0 - X(J)
               ELSE                    ; ! Even case of J
                  Fun(J) = 100.0*(X(J) - X(J-1)**2)
                  K = 0
               END IF
            END DO
!
            NSELECT = 1;
         END IF
!
 ---------------------------------------------------------------

!
 ---------------------------------------------------------------
      CASE (105) ; ! Ref An Autoadaptative limited memory Broyden's Method to Solve Systems of NEs :: Discrete integral equation function
!
 ---------------------------------------------------------------
         N = 10
         H = 1.0/FLOAT(N+1)
```

```
            HH= 0.5*H
            IF(IND_Case==1) THEN
                No_variable = N;
!
                DO   J = 1, N
                    TJ = H*FLOAT(J)
                    X(J) = TJ*(TJ-1.0)
                END DO
!
                NSELECT = 1;
!------------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                DO J = 1, N
                    TJ         = H*FLOAT(J)
!
                    Sum1 = 0.0
                    DO   K = 1, J
                        TK = H*FLOAT(K)
                        Sum1 = Sum1 + TK*(X(K)+TK+1.0)**3
                    END DO
!
                    Sum2 = 0.0
                    DO   K = J+1, N
                        TK = H*FLOAT(K)
                        Sum2 = Sum2 + (1.0 - TK)*(X(J)+TK+1.0)**3
                    END DO
!
                    Fun(J) = X(J) + HH*((1.0- TJ)*Sum1 + TJ*sum2)
                END DO
!
                NSELECT = 1;
            END IF
!------------------------------------------------------------------
```

```
!-----------------------------------------------------------
        CASE (106) ; ! Ref Shanghai Multi-step Nonlinear ABS Methods and Their Efficiency Analysis 1991 (Brown Problem)
!-----------------------------------------------------------
            N = 10
            IF(IND_Case==1) THEN
                No_variable = N;
!
                X(1:N) = 0.5
!
                NSELECT = 1;
!-----------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
               Pr1 = 1.0
               DO   K = 1, N
                    Pr1 = Pr1*X(K)
               END DO
               Fun(N) = -1.0 + Pr1
!
               DO   J = 2, N-1
                    Fun(J) = -FLOAT(N+1)
                    DO   K = J+1, N
                         Fun(J) = Fun(J) + X(K)
                    END DO
                END DO
!
                NSELECT = 1;
            END IF
!
-----------------------------------------------------------

!
-----------------------------------------------------------
        CASE (107) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #5
!
-----------------------------------------------------------
            N = 10
            X0=  0.0
```

```
            XN= 20.0
            IF(IND_Case==1) THEN
                No_variable = N;
                X(1:N) =    1.0
!
                NSELECT = 1;
!-----------------------------------------------------------------------
            ELSEIF(IND_Case==2) THEN
                DO J = 1, N
                    IF(J==1) THEN
                        Xm  = X0
                        Xx = X(J)
                        Xp = X(J+1)
                    ELSE IF(J==N) THEN
                        Xm = X(J-1)
                        Xx = X(J)  X
                        p = XN
                    ELSE
                        Xm = X(J-1)
                        Xx = X(J)  X
                        p = X(J+1)
                    END IF
!
                    Fun(J) = 3.0*Xx*(Xp -2.0*Xx + Xm) + 0.25*(Xp-Xm)**2
                END DO
!
                NSELECT = 1;
            END IF
!
-----------------------------------------------------------------------


!
-----------------------------------------------------------------------
        CASE (108) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #6
!
-----------------------------------------------------------------------
            IF(IND_Case==1) THEN
```

```
            No_variable = 10;
            X(1:N) = -0.1
!
            NSELECT = 1;
!------------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            Fun(1) = (3.0-5.0*X(1) )*X(1)    + 1.0 -2.0*X(2)
            Fun(10)= (3.0-5.0*X(10))*X(10) + 1.0 -        X(9)
            DO    J = 2, 9
               Fun(J)= (3.0-5.0*X(J))*X(J) - X(J-1) - 2.0*X(J+1)
            END DO
!
            NSELECT = 1;
         END IF
!------------------------------------------------------------


!------------------------------------------------------------
      CASE (109) ; ! ! Ref Atluri, A Modified Newton Method for Solving NAEs Example #7
!------------------------------------------------------------
         N = 10
         H = 1.0/FLOAT(N+1)
         H2= 1.0/H**2
!
         X0= 4.0
         XN= 1.0
!
         IF(IND_Case==1) THEN
            No_variable = N;
            X(1:N)          = 0.5
!
            NSELECT = 1;
!------------------------------------------------------------
         ELSEIF(IND_Case==2) THEN
            DO J = 1, N
```

```
                    IF(J==1) THEN
                        Xm  = X0
                        Xx = X(J)
                        Xp = X(J+1)
                    ELSE IF(J==N) THEN
                        Xm = X(J-1)
                        Xx = X(J)  X
                        p = XN
                    ELSE
                        Xm = X(J-1)
                        Xx = X(J)  X
                        p = X(J+1)
                    END IF
!
                    Fun(J) = H2*(Xp -2.0*Xx + Xm) - 1.5*Xx**2
                END DO
!
                NSELECT = 1;
            END IF
!-----------------------------------------------------------


!-----------------------------------------------------------
    END SELECT
!
    IF(NSELECT.EQ.0) THEN
        PRINT*,'NO PROBLEM IS SELECTED, SEE SUBROUTINE NLP_Problems for IND_PROBLEM = ', IND_PROBLEM
        STOP
    END IF
!
RETURN
END
!=================================================================================
```

Np	It_newt	Nf_newt	Fn_newt	Dx_newt	It_brdn	Nf_brdn	Fn_brdn	Dx_Brdn	It_Tmas	Nf_Tmas	Fn_Tmas	Dx_Tmas	It_mart	Nf_mart	Fn_mart	Dx_mart				
1	11	55	0.314018E-15	0.728109E-16	14	14	0.210486E-13	0.490187E-14	19	19	0.289978E-12	0.135531E-12	15	15	0.378128E-14	0.127834E-14	500	505	0.173576E-02	0.616325E-05
2	10	50	0.237783E-12	0.217161E-13	14	14	0.120334E-12	0.118316E-13	13	13	0.564844E-12	0.596211E-13	14	14	0.118498E-13	0.103636E-14	500	505	0.587316E-02	0.555262E-05
3	10	50	0.439626E-14	0.945924E-15	13	13	0.595989E-12	0.221141E-12	12	12	0.106766E-13	0.229617E-14	15	15	0.309272E-14	0.154266E-14	500	505	0.397494E-03	0.219393E-05
4	10	50	0.157009E-15	0.493563E-16	20	20	0.388806E-12	0.734353E-13	11	11	0.551088E-12	0.205790E-12	11	11	0.414695E-13	0.662863E-14	500	505	0.577830E-05	0.498691E-07
5	10	50	0.157009E-15	0.107252E-15	12	12	0.415029E-12	0.283751E-12	13	13	0.180706E-13	0.301705E-14	13	13	0.351984E-13	0.444482E-14	500	505	0.943471E-04	0.981775E-06
6	35	245	0.963470E-12	0.372914E-06	101	101	0.230735E+85	0.788668E+10	101	101	NaN	NaN	101	101	0.194128E+27	0.102558E-13	500	505	NaN	NaN
7	11	77	0.000000E+00	0.000000E+00	16	16	0.876837E-12	0.235614E-12	17	17	0.637615E-13	0.193977E-13	14	14	0.705456E-12	0.319848E-12	500	505	0.526664E-02	0.873318E-05
8	11	77	0.725195E-15	0.702075E-16	17	17	0.211677E-13	0.197035E-14	15	15	0.957971E-14	0.109693E-14	18	18	0.644159E-13	0.736436E-14	500	505	0.110273E-01	0.127884E-04
9	10	50	0.000000E+00	0.000000E+00	16	16	0.392523E-14	0.395443E-15	13	13	0.785046E-15	0.784943E-16	19	19	0.628086E-14	0.467219E-15	500	505	0.155750E-02	0.459616E-04
10	101	500	0.784772E+01	0.121673E+02	101	101	0.177408E+02	0.136895E+02	101	101	0.924939E+01	0.148465E+02	101	101	0.385372E+03	0.615735E+02	500	505	NaN	NaN
11	9	81	0.000000E+00	0.000000E+00	77	77	0.209302E-12	0.162038E-13	20	20	0.858842E-12	0.864412E-13	41	41	0.111022E-14	0.117153E-15	500	505	0.159514E-02	0.942575E-05
12	101	900	NaN	NaN	101	101	0.611603E-04	0.280025E-02	101	101	0.667863E-06	0.232192E-03	101	101	0.125840E-04	0.130658E-02	500	505	0.391605E-02	0.201710E-04
13	101	500	NaN	NaN	12	12	0.239142E-12	0.106927E-12	12	12	0.239142E-12	0.106927E-12	12	12	0.155431E-14	0.695109E-15	500	505	0.426203E-02	0.289752E-04
14	101	500	0.826539E-11	0.366395E-05	78	78	0.982218E-12	0.300409E-07	101	101	0.997873E-10	0.211875E-06	35	35	0.405667E-12	0.386207E-06	500	505	0.416466E-03	0.258069E-03
15	101	500	0.447148E+01	0.294290E+00	101	101	0.409754E+01	0.150103E-02	101	101	0.198576E+02	0.191478E+00	101	101	0.334405E+04	0.548425E+01	500	505	0.116989E+09	0.193125E+01
16	30	210	0.904843E-12	0.271993E-12	89	89	0.173068E-12	0.612027E-13	23	23	0.213220E-12	0.675918E-13	41	41	0.317583E-12	0.677076E-13	500	505	0.419002E-02	0.251980E-04
101	17	357	0.276556E-12	0.376083E-13	101	101	0.919831E+13	0.132828E+04	101	101	0.165520E+00	0.317172E-01	101	101	0.835198E+01	0.296900E+00	500	505	0.728118E-01	0.130043E-03
102	10	210	0.289468E-14	0.212347E-14	34	34	0.900751E-12	0.346585E-12	22	22	0.340257E-13	0.133692E-13	30	30	0.314446E-12	0.866142E-13	500	505	NaN	NaN
103	11	231	0.352834E-15	0.764035E-16	48	48	0.853439E-12	0.297119E-12	23	23	0.159601E-12	0.396714E-13	34	34	0.271327E-12	0.495800E-13	500	505	NaN	NaN
104	9	189	0.000000E+00	0.000000E+00	101	101	0.192036E-03	0.289833E-05	39	39	0.306079E-13	0.380775E-15	30	30	0.213685E-12	0.414231E-14	500	505	0.594295E-01	0.480422E-04
105	10	210	0.202302E-16	0.173801E-16	19	19	0.482188E-12	0.479602E-12	18	18	0.315927E-12	0.312288E-12	16	16	0.303084E-12	0.307928E-12	500	505	0.700192E-04	0.813046E-06
106	101	2100	NaN	NaN	101	101	0.316228E+00	0.123955E-13	101	101	0.316228E+00	0.471344E-07	101	101	0.316228E+00	0.931649E-12	500	505	0.316749E+00	0.361314E-02
107	22	462	0.315274E-12	0.426300E-10	101	101	0.553139E+12	0.145194E+06	101	101	0.136526E-03	0.260155E-01	101	101	0.207699E-01	0.181703E-01	500	505	0.114560E-01	0.146905E-03
108	11	231	0.464440E-16	0.938605E-17	33	33	0.724831E-12	0.216322E-12	22	22	0.650093E-12	0.150471E-12	31	31	0.169978E-12	0.382370E-13	500	505	0.298611E-03	0.918865E-06
109	11	231	0.333990E-13	0.202870E-15	101	101	0.245979E+04	0.117144E+02	34	34	0.330289E-12	0.113646E-13	101	101	0.637111E+05	0.157089E+03	500	505	0.171403E+01	0.871518E-04