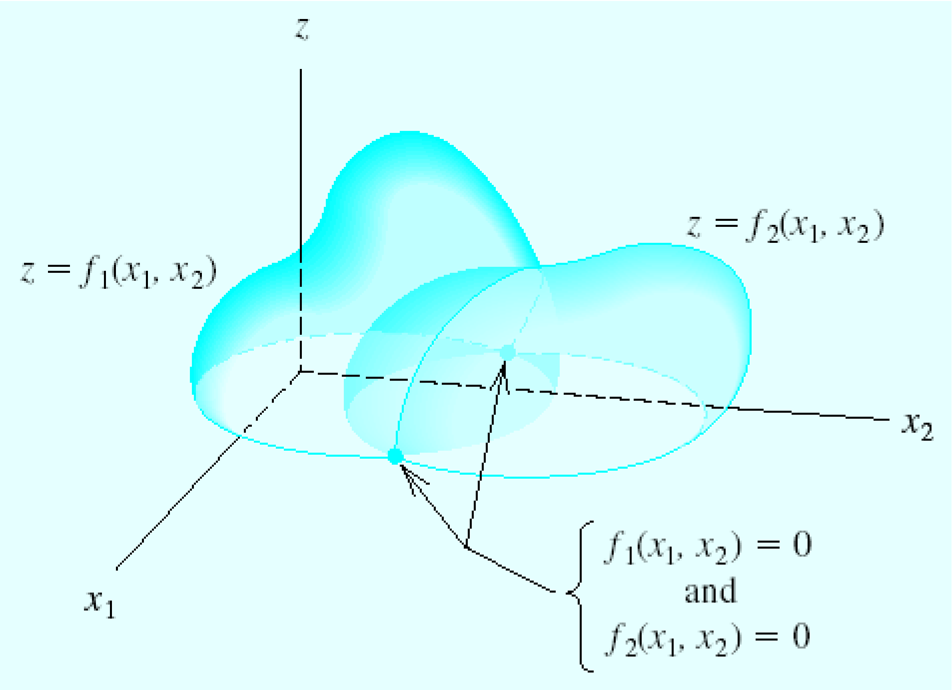
Numerical Analysis – Solving Nonlinear Equations

Hanyang University

Jong-Il Park

Nonlinear Equations



Nonlinear functions f_1 and f_2 of x_1 and x_2.

$$
\begin{aligned}
f_1(x_1^*, x_2^*) &= 0 \\
f_2(x_1^*, x_2^*) &= 0
\end{aligned}
$$

where x_1^*, x_2^* are true solution.

Newton's method(I)

Expanding the equations at (x_{1i}, x_{2i}) by Taylor series,

$$f_1(x_1^*, x_2^*) = f_1(x_{1i}, x_{2i}) + \left.\frac{\partial f_1}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_1}{\partial x_2}\right|_i \Delta x_{2i} + O(\ \boldsymbol{\Delta x}_i^{\ 2}\ ) = 0$$

$$f_2(x_1^*, x_2^*) = f_2(x_{1i}, x_{2i}) + \left.\frac{\partial f_2}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_2}{\partial x_2}\right|_i \Delta x_{2i} + O(\ \boldsymbol{\Delta x}_i^{\ 2}\ ) = 0$$

where $\Delta x_{ki} = x_k^* - x_{ki}, \quad k = 1, 2.$

Ignoring higher order terms,

$$\left.\frac{\partial f_1}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_1}{\partial x_2}\right|_i \Delta x_{2i} = -f_{1i}$$

$$\left.\frac{\partial f_2}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_2}{\partial x_2}\right|_i \Delta x_{2i} = -f_{2i}$$

Newton's method(II)

Rewriting the equations into a matrix form,

$$\nabla \boldsymbol{F} \; \Delta \boldsymbol{x} = - \boldsymbol{f}$$

where $\nabla \boldsymbol{F}$ is the Jacobian.

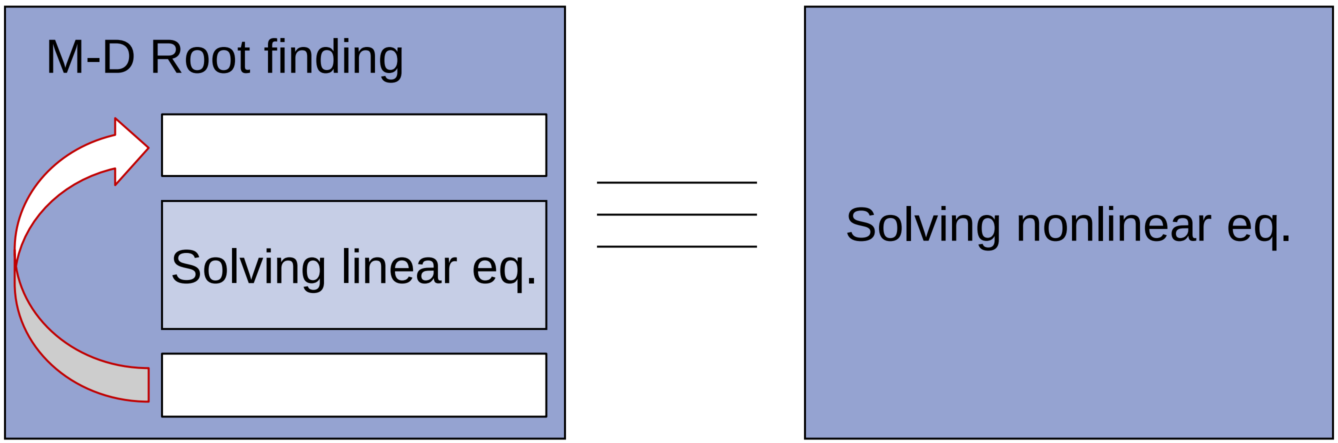
- Generalization to n-dimension

$$J(\mathbf{x})\Delta\mathbf{x} = -\mathbf{f}$$

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Solving nonlinear equation

- Multi-dimensional root finding



Newton's method - Algorithm

1) Initial guess $\boldsymbol{x}_0 = (x_{10}, x_{20}, \cdots, x_{n0})$, Set i=0.

2) Calculate the Jacobian and $\boldsymbol{f}$ at $\boldsymbol{x}_i$.

3) Solve the linear equation

$$\nabla \boldsymbol{F} \, \Delta \boldsymbol{x} = - \boldsymbol{f}$$

to get the update term $\Delta \boldsymbol{x}$

4) Update the solution

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta \boldsymbol{x}_i$$

5) If(convergence) -> stop

else set i=i+1 and goto 2).

Eg. Newton's method(I)

$$\mathbf{p}^{(k)} = \mathbf{p}^{(k-1)} - [J(\mathbf{p}^{(k-1)})]^{-1}\mathbf{F}(\mathbf{p}^{(k-1)}), \quad \text{for } k \geq 1,$$

Eg.

$$3x_1 - \cos(x_2 x_3) - \tfrac{1}{2} = 0,$$

$$x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0,$$

$$e^{-x_1 x_2} + 20x_3 + \tfrac{1}{3}(10\pi - 3) = 0$$

$$\mathbf{p}^{(0)} = (0.1, 0.1, -0.1)^t$$

Sol.

$$J(x_1, x_2, x_3) = \begin{bmatrix} 3 & x_3 \sin x_2 x_3 & x_2 \sin x_2 x_3 \\ 2x_1 & -162(x_2 + 0.1) & \cos x_3 \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix}$$

Eg. Newton's method(II)

$$\begin{bmatrix} p_1^{(k)} \\ p_2^{(k)} \\ p_3^{(k)} \end{bmatrix} = \begin{bmatrix} p_1^{(k-1)} \\ p_2^{(k-1)} \\ p_3^{(k-1)} \end{bmatrix} + \begin{bmatrix} y_1^{(k-1)} \\ y_2^{(k-1)} \\ y_3^{(k-1)} \end{bmatrix},$$

$$\begin{bmatrix} y_1^{(k-1)} \\ y_2^{(k-1)} \\ y_3^{(k-1)} \end{bmatrix} = -\left(J(p_1^{(k-1)}, p_2^{(k-1)}, p_3^{(k-1)})\right)^{-1} \mathbf{F}\left(p_1^{(k-1)}, p_2^{(k-1)}, p_3^{(k-1)}\right)$$

At each step

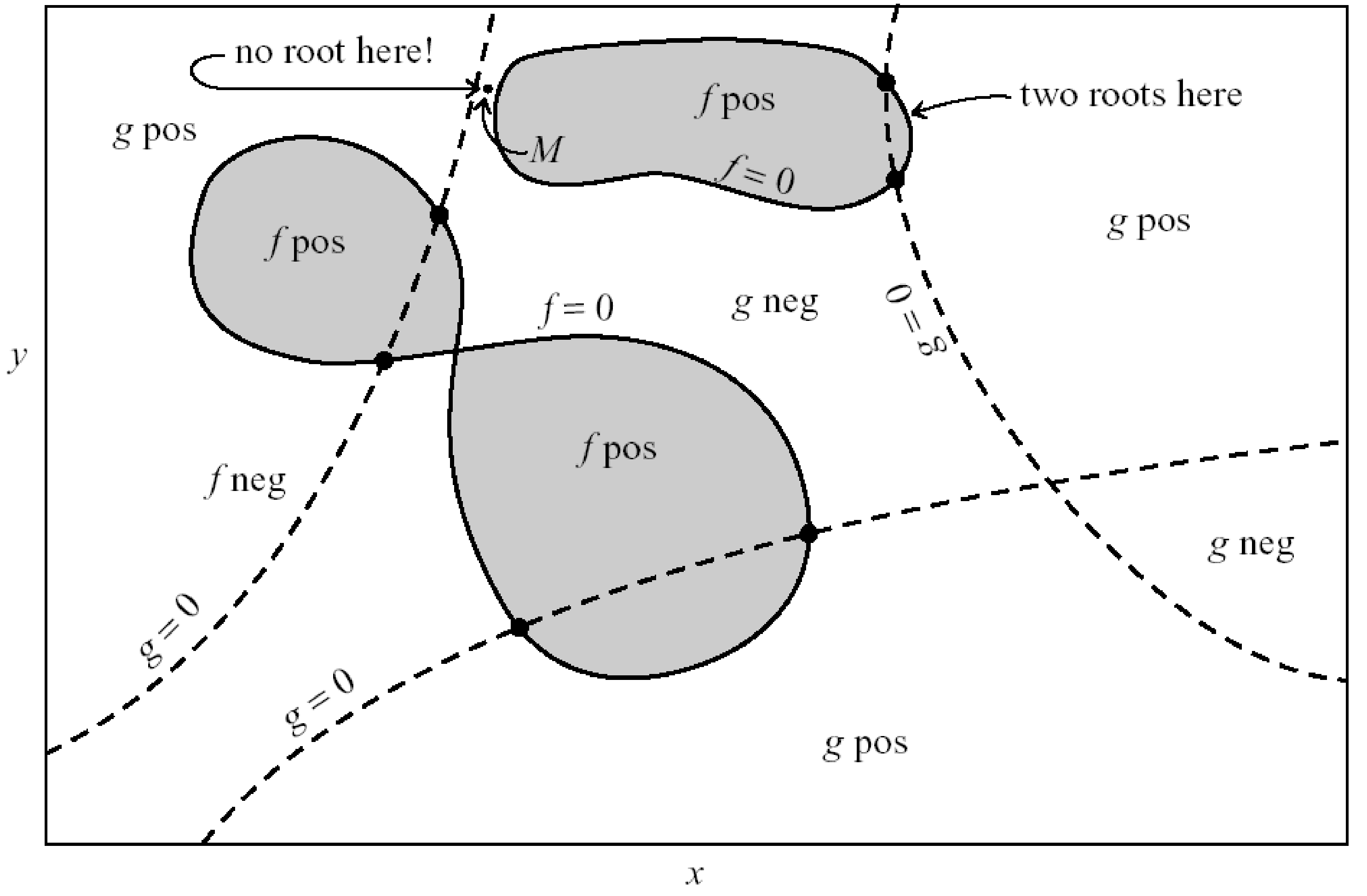
the linear system $J(\mathbf{p}^{(k-1)})\mathbf{y}^{(k-1)} = -\mathbf{F}(\mathbf{p}^{(k-1)})$ must be solved

Eg. Newton's method(II)

- Result:

k	$p_1^{(k)}$	$p_2^{(k)}$	$p_3^{(k)}$	$\|\mathbf{p}^{(k)} - \mathbf{p}^{(k-1)}\|_\infty$
0	0.10000000	0.10000000	−0.10000000	
1	0.49986967	0.01946684	−0.52152047	0.422
2	0.50001424	0.00158859	−0.52355696	1.79×10^{-2}
3	0.50000011	0.00001244	−0.52359845	1.58×10^{-3}
4	0.50000000	0.00000000	−0.52359877	1.24×10^{-5}
5	0.50000000	0.00000000	−0.52359877	8.04×10^{-10}

Discussion


no root here!
two roots here
M
f pos
f = 0
g pos
f pos
f = 0
g neg
g = 0
g pos
f pos
f neg
g neg
g = 0
g = 0
g pos
x
y

Quasi-Newton method(I)

Broyden's method

- Without calculating the Jacobian at each iteration
- Using approximation:

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

- Analogy
  - Root finding: Newton vs. Secant
  - Nonlinear eq.: Newton vs. Broyden

➔ Broyden's method is called **"multidimensional secant method"**

* Read Section 10.3, *Numerical Methods, 3rd ed.* by Faires and Burden

Quasi-Newton method(II)

- Replacing the Jacobian with the matrix A

$$A_i = A_{i-1} + \frac{\mathbf{y}_i - A_{i-1}\mathbf{s}_i}{\|\mathbf{s}_i\|_2^2}\mathbf{s}_i^t$$

$$\mathbf{p}^{(i+1)} = \mathbf{p}^{(i)} - A_i^{-1}\mathbf{F}\left(\mathbf{p}^{(i)}\right)$$

where the notation $\mathbf{s}_i = \mathbf{p}^{(i)} - \mathbf{p}^{(i-1)}$ and $\mathbf{y}_i = \mathbf{F}(\mathbf{p}^{(i)}) - \mathbf{F}(\mathbf{p}^{(i-1)})$

- Important property of calculating A_i^{-1}

$$A_i^{-1} = A_{i-1}^{-1} + \frac{\left(\mathbf{s}_i - A_{i-1}^{-1}\mathbf{y}_i\right)\mathbf{s}_i^t A_{i-1}^{-1}}{\mathbf{s}_i^t A_{i-1}^{-1}\mathbf{y}_i}.$$

This update involves only matrix-vector multiplication!

Eg. Broyden's method

- Results:

k	$p_1^{(k)}$	$p_2^{(k)}$	$p_3^{(k)}$	$\|\mathbf{p}^{(k)} - \mathbf{p}^{(k-1)}\|_2$
0	0.1000000	0.1000000	−0.1000000	
1	0.4998697	-1.946685×10^{-2}	−0.5215205	5.93×10^{-1}
2	0.4999863	8.737833×10^{-3}	−0.5231746	2.83×10^{-2}
3	0.5000066	8.672215×10^{-4}	−0.5236918	7.89×10^{-3}
4	0.5000005	6.087473×10^{-5}	−0.5235954	8.12×10^{-4}
5	0.5000002	-1.445223×10^{-6}	−0.5235989	6.24×10^{-5}

Slightly less accurate than Newton's method.

k	$p_1^{(k)}$	$p_2^{(k)}$	$p_3^{(k)}$	$\|\mathbf{p}^{(k)} - \mathbf{p}^{(k-1)}\|_\infty$
0	0.10000000	0.10000000	−0.10000000	
1	0.49986967	0.01946684	−0.52152047	0.422
2	0.50001424	0.00158859	−0.52355696	1.79×10^{-2}
3	0.50000011	0.00001244	−0.52359845	1.58×10^{-3}
4	0.50000000	0.00000000	−0.52359877	1.24×10^{-5}
5	0.50000000	0.00000000	−0.52359877	8.04×10^{-10}

Steepest Descent Method(I)

- Finding a local minimum for a multivariable function of the form $g : \mathcal{R}^n \to \mathcal{R}$

$$g(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} [f_i(x_1, x_2, \ldots, x_n)]^2$$

- Algorithm

  - Evaluate g at an initial approximation $\mathbf{p}^{(0)} = (p_1^{(0)}, p_2^{(0)}, \ldots, p_n^{(0)})^t$.
  - Determine a direction from $\mathbf{p}^{(0)}$ that results in a decrease in the value of g.
  - Move an appropriate amount in this direction and call the new value $\mathbf{p}^{(1)}$.
  - Repeat the steps with $\mathbf{p}^{(0)}$ replaced by $\mathbf{p}^{(1)}$.

$$\mathbf{p}^{(1)} = \mathbf{p}^{(0)} - \hat{\alpha} \nabla g\left(\mathbf{p}^{(0)}\right)$$

where $\nabla g(\mathbf{x}) = \left(\frac{\partial g}{\partial x_1}(\mathbf{x}), \frac{\partial g}{\partial x_2}(\mathbf{x}), \ldots, \frac{\partial g}{\partial x_n}(\mathbf{x}) \right)^t$

Steepest Descent Method(II)

• Mostly used for finding an appropriate initial value of Newton's methods etc.