[수치해석] Nonlinear_equation

Multi-dimensional root finding의 경우를 살펴보자.

N개의 Root를 찾기 위해선 N개의 방정식이 필요하다.

2개의 Root를 찾기 위해 2개의 Nonlinear function (방정식) $f_1(x_1^*, x_2^*) = 0, f_2(x_1^*, x_2^*)$ = 0 의 연립방정식을 푼다고 생각해보자.

- Root는 x_1^*, x_2^*이다.
- 사실 위 연립방정식 형태는 Linear 한 경우에는 A의 한 행이 Linear function인 Ax = b로 나타난다.
- L**inear funcntion인 경우 Gauss, LU, SVD 등을 통해 Ax = b를 직접 해결하는 것이 효율적**이다.
- **Newton's method를 비롯한 방법은 Non-linear일 때 사용한다고 생각**하는 것이 좋다.

1차 Taylor 전개를 수행하면 아래와 같다.

- 나머지 Higher order term은 무시한다.

$$f_1(x_{1i}^*, x_{2i}^*) = f_1(x_{1i}, x_{2i}) + \left.\frac{\partial f_1}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_1}{\partial x_2}\right|_i \Delta x_{2i} + O(\Delta \mathbf{x}_i^2) = 0$$
$$f_2(x_{1i}^*, x_{2i}^*) = f_2(x_{1i}, x_{2i}) + \left.\frac{\partial f_2}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_2}{\partial x_2}\right|_i \Delta x_{2i} + O(\Delta \mathbf{x}_i^2) = 0$$

- Non-linear function을 linear function으로 근사하는 과정이다.
- $\Delta x = x^* - x$이다.

위 식을 정리하면 아래와 같다.

$$\left.\frac{\partial f_1}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_1}{\partial x_2}\right|_i \Delta x_{2i} = -f_{1i}$$
$$\left.\frac{\partial f_2}{\partial x_1}\right|_i \Delta x_{1i} + \left.\frac{\partial f_2}{\partial x_2}\right|_i \Delta x_{2i} = -f_{2i}$$

4개의 편미분을 행렬 형태로 나타내보자.

- 그 행렬을 **Jacobian Matrix**라고 한다.
- $J(A), \nabla f$라고 부른다.

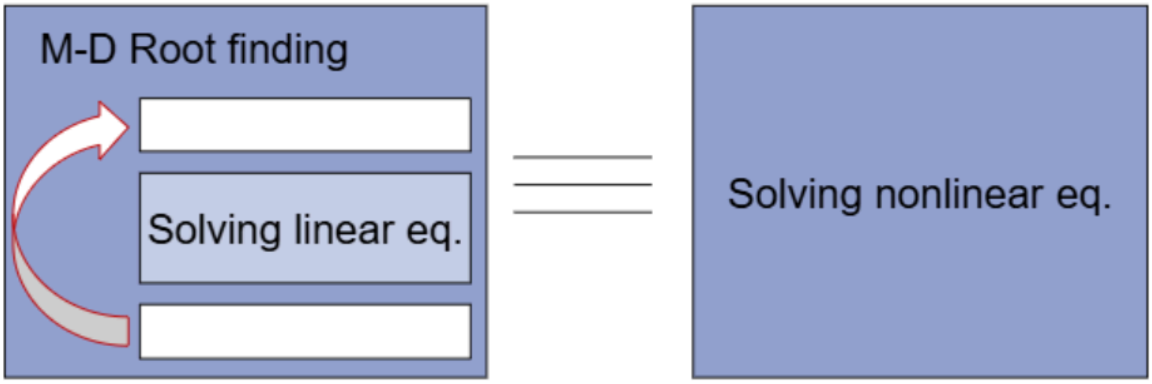
$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

그럼 위 연립방정식을 행렬과 벡터의 곱 형태로 나타낼 수 있다.

- $J(x)\Delta x = f$, $\nabla f \Delta x = f$

Nonlinaer equation을 푸는 것은 Multi-dimensional root finding을 수행하는 것과 같다.

Nonlinear equation을 푸는 것은 **Linaer Approximation → Linear equation 해결 → X를 Update** 하는 과정을 반복하는 것과 동일하다.



Nonlinear equation을 푸는 방법론은 세 가지가 있다.

Newton's method

과정

1. $x_0 = [x_{10}, x_{20}, \dots, x_{n0}]$을 초기값으로 잡는다.
2. x_i에 대해 Jacobian과 함수값 f를 계산한다.
3. $\nabla f \Delta x = f$를 푼다.
4. New x_i = $x_i + \Delta x_i$로 업데이트한다.
5. 수렴하면 멈추고 아니라면 (2) ~ (4) 과정을 반복한다**.**

업데이트 공식

$$P^{(k)} = P^{(k-1)} - J(p^{(k-1)})^{-1} F(p^{(k-1)})$$

각 Iteration마다 $J(P^{(k-1)})y^{(k-1)} = -F(P^{(k-1)})$를 풀어야 한다.

Quasi-Newton method - Brayden's method

각 반복 단계마다 **Jacobian의 역행렬을 계산하지 말고 Jacobian을 근사한다.**

$$f'(P_i) = \frac{f(P_1) - f(P_0)}{P_1 - P_0}$$

- Root finding에서의 Secant method와 비슷하다.
- Brayden's method를 **Multidimensional secant method**라고도 부른다.

Jacobian matrix를 A로 근사한다.

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

- $\mathbf{y}_i - A_{i-1}\mathbf{s}_i$는 실제 함수값이랑 추정한 값의 오차이다.
- $\frac{s_i^t}{||s_i|_2^2|}$은 보정항이다.
- $s_i = p^{(i)} - p^{(i-1)}$, $y_i = F(P^{(i)} - F(P^{(i-1)})$이다.

이후 다음의 업데이트 공식을 사용한다.

$$P^{(k)} = P^{(k-1)} - A_i^{-1}F(p^{(k-1)}$$

A의 역행렬을 구하면 다음과 같다.

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

- 원래 Jacobian의 역행렬을 구하려면 행렬 x 행렬 연산으로 인해 $O(N^3)$의 연산량이 들었다.
- 위 A의 역행렬을 구하는 과정은 행렬 x 벡터 연산으로만 이루어져 $O(N^2)$의 연산량이 든다.

Steepest Descent Method

함수 f(x)에 대해 **g(x)를 아래와 같이 정의하고 Local minimum을 찾는 문제로 변환하면 f(x) = 0인 x를 찾는 문제와 동일**해진다.

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

과정

1. 초기값 $P^{(0)} = (P_1^{(0)}, P_2^{(0)} \dots P_n^{(0)})$
2. $p^{(i)}$에서 g(x)의 방향이 **가장 크게 감소하는 방향**을 찾는다.
    a. Gradient의 반대 방향이다.
3. $p^{(i)}$를 해당 방향으로 움직이도록 Update한다.
    a. $P^{(k)} = P^{(k-1)} - \hat{\alpha}\nabla g(p^{(0)})$
4. 수렴할 때까지 (2) ~ (3) 과정을 반복한다.

보통 **Steepest Descant method를 통해 f(x) = 0의 Solution x 근처까지만 이동하구, 근처에서는 Newton's method를 이용하여 정밀하게 탐색**한다.