metode numerik - all in one

edisi sesatmaxxing vol. 2

the one and only @kafeyangasli

June 7, 2025

1 Aproksimasi MNA pada PDB

Masalah Nilai Awal (*Initial Value Problem*) mencari persamaan eksak dari diketahui suatu persamaan diferensial. Terdapat beberapa metode yang dapat digunakan:

1. Metode Taylor

   Cara menggunakan metode ini adalah dengan menjabarkan deret Taylor sampai orde tertentu:

   $$y_{n+1} = y_n + \frac{f(x_n, y_n)}{1!} \cdot \Delta x + \frac{f'(x_n, y_n)}{2!} \cdot \Delta x^2 + ...$$

   Jika digunakan metode ini sampai orde ke-n, maka kita perlu mencari juga turunan ke-$n-1$ dari persamaan diferensial tersebut juga.

   Misal dari persamaan diferensial $P'(t) = 0.1 \cdot P(t)$, kita akan mengaproksimasi IVP menggunakan Taylor Orde 2. Maka, perlu mencari turunan pertama dari PD tersebut, yaitu:

   $$\begin{aligned} P'(t) &= 0.1 \cdot P(t) \\ P''(t) &= 0.1 \cdot P'(t) \\ &= 0.1 \cdot (0.1 \cdot P(t)) \\ &= 0.1^2 \cdot P(t) \end{aligned}$$

2. Metode Euler / Runge Kutta 1

   Metode Euler adalah Metode Taylor yang dipotong sampai orde 1.

   $$y_{n+1} = y_n + f(x_n, y_n) \cdot \Delta x$$

3. Metode Heun / RK2 / Euler's Predictor Corrector (EPC)

   Metode Heun menggunakan konsep **prediktor** dan **korektor**, dimana dua prediktor akan dihitung terlebih dahulu menggunakan metode Euler, lalu menghitung rata-rata garis miring prediktor tersebut untuk menghasilkan korektor.

   Korektor didapatkan dari:

   $$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \Delta x, y_n + \Delta x \cdot k_1) \\ m &= \frac{(k_1 + k_2)}{2} \\ y_{n+1} &= y_n + m \cdot \Delta x \end{aligned}$$

4. Metode Runge-Kutta 3 dan 4

   Metode RK3 dan RK4 menggunakan konsep yang sama dengan RK2, namun dengan jumlah korektor yang lebih banyak.

   - RK3

   $$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \frac{\Delta x}{2}, y_n + \frac{\Delta x \cdot k_1}{2}) \\ k_3 &= f(x_n + \Delta, y_n - k_1 \cdot \Delta x + 2k_2 \Delta x) \\ m &= \frac{(k_1 + 4k_2 + k_3)}{6} \\ y_{n+1} &= y_n + m \cdot \Delta x \end{aligned}$$

- RK4

$$
\begin{aligned}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{\Delta x}{2}, y_n + \frac{\Delta x \cdot k_1}{2}) \\
k_3 &= f(x_n + \frac{\Delta x}{2}, y_n + \frac{\Delta x \cdot k_2}{2}) \\
k_4 &= f(x_n + \Delta x, y_n + k_3 \cdot \Delta x) \\
m &= \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6} \\
y_{n+1} &= y_n + m \cdot \Delta x
\end{aligned}
$$

2 Aproksimasi Sistem Persamaan Linear

Bentuk umum dari SPL adalah:

$$A \cdot X = B$$

Misal diberikan SPL:

$$
\begin{aligned}
10x + 5y + 2z &= 75 \\
x + 7y + 5z &= 101 \\
6x + 3y + 9z &= 123
\end{aligned}
$$

Matriks SPL yang terbentuk dalam bentuk $A \cdot X = B$ adalah:

$$
\begin{bmatrix} 10 & 5 & 2 \\ 1 & 7 & 5 \\ 6 & 3 & 9 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} =
\begin{bmatrix} 75 \\ 101 \\ 123 \end{bmatrix}
$$

Sebelum memulai perhitungan, **USAHAKAN** matriks A yang terbentuk memiliki diagonal yang dominan atau dalam kata lain, nilai elemen diagonal lebih besar atau sama dengan dibandingkan hasil jumlah semua elemen yang lain dalam satu baris. Matriks di atas sudah memiliki diagonal yang dominan, dimana:

- Elemen $A_{1,1}$ bernilai lebih besar dibandingkan $A_{1,2} + A_{1,3}$
- Elemen $A_{2,2}$ bernilai lebih besar dibandingkan $A_{2,1} + A_{2,3}$
- Elemen $A_{3,3}$ bernilai sama dengan $A_{3,1} + A_{3,2}$

Jika belum memiliki diagonal yang dominan, maka lakukan pertukaran baris apabila memungkinkan. Misal dari persamaan SPL dibawah ini yang belum memiliki diagonal yang dominan:

$$
\begin{bmatrix} 1 & 7 & 1 \\ 1 & 1 & 8 \\ 3 & 1 & 1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} =
\begin{bmatrix} 9 \\ 12 \\ 5 \end{bmatrix}
\xrightarrow{H_{12}}
\begin{bmatrix} 1 & 1 & 8 \\ 1 & 7 & 1 \\ 3 & 1 & 1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} =
\begin{bmatrix} 12 \\ 9 \\ 5 \end{bmatrix}
\xrightarrow{H_{13}}
\begin{bmatrix} 3 & 1 & 1 \\ 1 & 7 & 1 \\ 1 & 1 & 8 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} =
\begin{bmatrix} 5 \\ 9 \\ 12 \end{bmatrix}
$$

Terdapat beberapa metode yang dapat digunakan:

1. Metode Jacobian

   Metode Jacobian didefinisikan dengan:

$$x_i^{(k+1)} = \frac{1}{A_{i,i}} \left(B_i - \sum_{i \neq j} A_{i,j} \cdot x_j^{(k)} \right)$$

   Dari SPL diatas, maka persamaan lelarannya adalah:

$$
\begin{aligned}
x^{(k+1)} &= \frac{1}{10} \left(75 - 5y^{(k)} - 2z^{(k)} \right) \\
y^{(k+1)} &= \frac{1}{7} \left(101 - x^{(k)} - 5z^{(k)} \right) \\
z^{(k+1)} &= \frac{1}{9} \left(123 - 6x^{(k)} - 3y^{(k)} \right)
\end{aligned}
$$

2. Metode Gauss-Seidel

   Secara cara mengerjakan, Gauss-Seidel mirip dengan Jacobian, dengan perbedaan pada saat iterasi selanjutnya, dia akan menggunakan nilai sebelumnya yang sudah terhitung.

$$x_i^{(k+1)} = \frac{1}{A_{i,i}} \left(B_i - \sum_{i>j} A_{i,j} \cdot x_j^{(k+1)} - \sum_{i<j} A_{i,j} \cdot x_j^{(k)} \right)$$

   Dengan sedikit modifikasi, maka persamaan lelarannya menjadi:

$$\begin{aligned} x^{(k+1)} &= \frac{1}{10} \left(75 - 5y^{(k)} - 2z^{(k)} \right) \\ y^{(k+1)} &= \frac{1}{7} \left(101 - x^{(k+1)} - 5z^{(k)} \right) \\ z^{(k+1)} &= \frac{1}{9} \left(123 - 6x^{(k+1)} - 3y^{(k+1)} \right) \end{aligned}$$

3. Metode *Successive Over-Relaxation*

   Untuk Metode SOR, biasanya akan diberikan faktor relaksasi (ω) yang akan digunakan di persamaan lelaran:

$$x_i^{(k+1)} = (1-\omega) x_i^{(k)} + \omega \cdot G.S$$

   dimana G.S adalah persamaan Gauss-Seidel biasa.

   Ketika menggunakan SOR, terdapat 3 kemungkinan:

   (a) $\omega = 1$: stabil layaknya Gauss-Seidel

   (b) $\omega > 1$: konvergensi cepat tapi cenderung tidak stabil

   (c) $\omega < 1$: konvergensi lambat tapi stabil

3 Aproksimasi Fungsi Rasional

Terdapat beberapa metode yang dapat digunakan:

1. Aproksimasi Pade

   Bentuk umum aproksimasi Pade berderajat-N dimana $N = n + m$ adalah:

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1 x + p_2 x^2 + ... + p_n x^n}{1 + q_1 x + q_2 x^2 + ... + q_m x^m}$$

   Untuk mendapatkan koefisien p dan q, biasanya $r(x)$ akan ditempatkan oleh ekspansi dari fungsi yang dicari menggunakan Deret Maclaurin, misal:

$$r(x) = e^x = \sum_{n=0}^{N} \frac{x^{(n)}}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + ... + \frac{x^N}{N!}$$

   Lalu, dengan $r(x)$ yang didapat, kalikan kedua ruas dengan $q(x)$ sehingga terbentuk:

$$r(x)q(x) = p(x)$$

   Setelahnya, ekspansikan $r(x)q(x)$ jika memungkinkan untuk mendapatkan sistem persamaan untuk mencari koefisien p.

   Untuk mempermudah, gunakan fungsi persamaan yang ada pada kalkulator (jika ada) untuk mencari koefisien p dan q!

2. Aproksimasi Chebyshev / Pade-Chebyshev

   Cara mendapatkan aproksimasi fungsi rasional menggunakan Pade-Chebyshev hampir mirip dengan Pade, dengan perbedaan pada $r(x)$ yang menggunakan polinomial Chebyshev dan koefisien x^n berubah menjadi $T_n(x)$, sehingga:

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 T_0(x) + p_1 T_1(x) + p_2 T_2(x) + ... + p_n T_n(x)}{T_0(x) + q_1 T_1(x) + q_2 T_2(x) + ... + q_m T_m(x)}$$

Konversi polinomial Chebyshev ke polinomial biasa didefinisikan secara rekursif:

$$T_n(x) = 2x \cdot T_{n-1}(x) - T_{n-2}(x)$$

dengan basis:

$$T_0 = 1,\ T_1 = x$$

Lalu pada saat mengekspansi $r(x)q(x)$, gunakan sifat:

$$T_i \cdot T_j = \frac{1}{2}\left(T_{i+j} + T_{|i-j|}\right)$$

Dalam soal, seharusnya polinomial Chebyshev akan diberikan. Namun, untuk berjaga-jaga, saya lampirkan polinomial Chebyshev e^{-x} sampai suku ke-lima:

$$\begin{aligned} e^{-x} = {} & 1.266066T_0(x) - 1.130318T_1(x) + 0.271495T_2(x) - \\ & 0.044337T_3(x) + 0.005474T_4(x) - 0.000543T_5(x) \end{aligned}$$

Dan cara mengonversi polinomial Maclaurin menjadi Chebyshev sampai suku ke-lima didapatkan dari SPL:

$$\begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & 0 & 0 \\ 0 & -3 & 0 & 4 & 0 & 0 \\ 1 & 0 & -8 & 0 & 8 & 0 \\ 0 & 5 & 0 & -20 & 0 & 16 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \end{bmatrix}$$

dan menginversnya menjadi:

$$\begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{3}{4} & 0 & \frac{1}{4} & 0 & 0 \\ \frac{3}{8} & 0 & \frac{1}{2} & 0 & \frac{1}{8} & 0 \\ 0 & \frac{5}{8} & 0 & \frac{5}{16} & 0 & \frac{1}{16} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}$$

Sehingga:

$$x^0 = 1 = T_0,\ x^1 = T_1,\ x^2 = \frac{1}{2}T_0 + \frac{1}{2}T_2,\ dst..$$

Jika ingin mempelajari lebih lanjut, maka bisa melihat sumber dibawah ini:
https://homepages.math.uic.edu/~jan/mcs471/chebyshev.pdf

4 Aproksimasi Polinomial Trigonometri

Secara singkat, aproksimasi Polinomial Trigonometri di interval $-\pi < x < \pi$ dihitung dengan rumus:

$$S_n(x) = a_0 + \sum_{k=1}^{n} (a_k \cos(kx)) + \sum_{k=1}^{n-1} (b_k \sin(kx))$$

dimana:

$$\begin{aligned} a_0 &= \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\,dx \\ a_k &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cdot\cos(kx)\,dx \\ b_k &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cdot\sin(kx)\,dx \end{aligned}$$

Tampak familiar? Jika $n \to \infty$, maka polinomial tersebut akan menjadi **Deret Fourier** yang sudah kita kenal di Matematika II. Sehingga, semua sifat-sifat yang berkaitan dengannya akan juga berlaku di sini!

5 *Discrete Fourier Transform* (DFT)

Fourier Transform adalah salah satu teknik untuk mengubah citra dari domain spasial ke domain frekuensi. Rumus untuk menghitung DFT adalah:

$$F(u) = \sum_{x=0}^{N-1} f(x) \cdot e^{-i \cdot \frac{2\pi \cdot u \cdot x}{N}}$$

dan inversnya:

$$f(x) = \frac{1}{N} \sum_{x=0}^{N-1} F(u) \cdot e^{i \cdot \frac{2\pi \cdot u \cdot x}{N}}$$

Ketika kita melakukan komputasi Discrete Fourier Transform, terutama pada $e^{i \cdot \frac{2\pi \cdot u \cdot x}{N}}$, dia akan membutuhkan tenaga dan waktu yang cukup boros. Sehingga, muncullah kehadiran Algoritma FFT, yang dimana ia melakukan komputasi bagian tersebut secara cepat.

Secara singkat, FFT menghitung bagian imajiner dari transformasi menggunakan Twiddle Factor:

$$W_N = e^{-i\frac{2\pi}{N}}$$

.

Setelahnya, FFT akan memisah indeks ganjil dan genap dari data yang ada, lalu menghitung koefisien setiap data tersebut dengan:

$$E_k = \sum_{n=genap}^{N} x_n \cdot W_N^{nk}$$

dan

$$O_k = \sum_{n=ganjil}^{N} x_n \cdot W_N^{nk}$$

Setelah itu, menghitung nilai transformasinya dengan:

$$X_k = E_k + W_N^k \cdot O_{k\%2}$$

6 Aproksimasi Eigenvalue

Terdapat beberapa metode yang dapat digunakan:

1. Metode Power

   Jika diberikan sebuah matriks dan suatu vektor tebakan, maka Metode Power bisa digunakan dengan cara:

   $$y^{(k+1)} = A \cdot x^{(k)}, \text{lalu menghitung } x \text{ selanjutnya dengan: } x^{(k+1)} = \frac{1}{||y^{(k+1)}||_\infty} \cdot y^{(k+1)}$$

   Semakin banyak iterasi yang dilakukan, maka akan menghasilkan Eigenvalue ($||y^{(k)}||_\infty$) dan Eigenvector ($x^{(k)}$) terbesar dari matriks tersebut.

2. Metode Inverse-Power

   Metode Inverse Power memberikan konvergensi yang lebih cepat namun sedikit lebih rumit jika dikerjakan secara manual. Daripada menggunakan matriks A secara langsung, dia mengubah matriks A menjadi dalam bentuk:

   $$(A - qI)$$

   dimana q didapatkan dari:

   $$q = \frac{x^{(0)t} \cdot A \cdot x^{(0)}}{x^{(0)t} \cdot x^{(0)}}$$

   Lalu, mencari $y^{(k+1)}$ dari persamaan:

   $$(A - qI)(y^{(k+1)}) = x^{(k)}$$
   $$y^{(k+1)} = (A - qI)^{-1}(x^{(k)})$$

   dan $x^{(k+1)}$ dari persamaan di atas sebelumnya.

3. Metode QR Iteration dengan QR Decomposition Gram-Schmidt

   Konsep dasar dari metode ini adalah memecah matriks A menjadi dua matriks Q dan R yang memenuhi persamaan

   $$A = Q \cdot R$$

   , lalu mengalikan kembali dua matriks tersebut dalam urutan yang berkebalikan.

   Ada banyak metode yang bisa digunakan untuk memecah matriks menjadi bentuk QR. Namun, untuk mempermudah, maka metode Gram-Schmidt akan digunakan, berhubung konsepnya sudah sekilas dikenal dari mata kuliah Aljabar Linier masa kala.

   Diberikan matriks A awal:

   $$A^{(k)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{\text{dalam bentuk himpunan vektor}} A = \left\{ \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}, \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix}, \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix} \right\}$$

   Lalu, dengan proses Gram-Schmidt:

   $$v_1 = a_{x1}$$
   $$v_2 = a_{x2} - \frac{\langle a_{x2}, v_1 \rangle}{||v_1||^2} \cdot v_1$$
   $$v_3 = a_{x3} - \frac{\langle a_{x3}, v_1 \rangle}{||v_1||^2} \cdot v_1 - \frac{\langle a_{x3}, v_2 \rangle}{||v_2||^2} \cdot v_2$$

   Dari vektor yang didapatkan, maka akan menghasilkan matriks Q:

   $$Q^{(k)} = \left\{ \frac{v_1}{||v_1||^2}, \frac{v_2}{||v_2||^2}, \frac{v_3}{||v_3||^2} \right\}$$

   Dari matriks Q yang didapatkan, kita bisa mencari R dengan:

   $$Q^{T(k)} \cdot A^{(k)} = R^{(k)}$$

   Setelah mendapatkan Q dan R, maka untuk mencari nilai Eigenvaluenya, kali kembali kedua matriks tersebut dalam urutan terbalik:

   $$A^{(k+1)} = R^{(k)} \cdot Q^{(k)}$$

   Matriks A baru tersebut akan digunakan kembali untuk QR Decomposition lalu QR Iteration. Semakin banyaknya iterasi yang dilakukan, maka akan menghasilkan matriks diagonal yang berisikan Eigenvalue dari matriks A.

4. Metode Singular Value Decomposition

   Sebenarnya, metode SVD ini digunakan untuk mencari *singular values* dari suatu matriks non-persegi. Singular Value adalah akar dari Eigenvalue suatu matriks.

   Tetapi, matriks non-persegi tidak memiliki Eigenvalue!

   Memang, namun setiap matriks **pasti** memiliki Singular Value, dan ada kalanya peran Singular Value pada matriks non-persegi hampir mirip dengan Eigenvalue pada matriks persegi. Sehingga, kita juga perlu mengetahui bagaimana cara mencarinya!

   Untuk mendapatkan Singular Value dari matriks non-persegi, kita perlu membuat matriks simetrisnya, yaitu dengan cara:

   $$S = A^T \cdot A$$

   Lalu, menyelesaikannya dengan metode pencarian Eigenvalue biasa:

   $$det(S - \lambda I) = 0$$

7 Aproksimasi Sistem Persamaan Non-Linear

Sistem Persamaan Non-Linear memiliki bentuk:

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

yang bisa direpresentasikan sebagai fungsi koordinat F:

$$F(x_1,x_2,\ldots,x_n) = (f_1(x_1,x_2,\ldots,x_n), f_2(x_1,x_2,\ldots,x_n),\ldots,f_n(x_1,x_2,\ldots,x_n))^T$$

Terdapat beberapa metode yang dapat digunakan:

1. Metode Titik Tetap (*Fixed Point*)

   Metode Titik Tetap mengubah $f(x_1,x_2,\ldots,x_n) = 0$ menjadi dalam bentuk $x = g(x)$.

   Misal diberikan SPNL:

$$x_1^2 - 10x_1 + x_2^2 + 8 = 0 \quad \xrightarrow{\textit{dalam bentuk } x = g(x)} \quad x_1 = g_1(x_1,x_2) = \frac{1}{10}\left(x_1^2 + x_2^2 + 8\right)$$
$$x_1x_2^2 + x_1 - 10x_2 + 8 = 0 \quad \xrightarrow{\textit{dalam bentuk } x = g(x)} \quad x_2 = g_2(x_1,x_2)\frac{1}{10}\left(x_1x_2^2 + x_1 + 8\right)$$

   Lalu menggunakan vektor tebakan $x^{(0)}$, iterasikan seperti SPL biasa.

   Ingat, perubahan menjadi bentuk $x = g(x)$ menjadi peran penting dalam memastikan konvergensi iterasi. Cara untuk mengecek apakah persamaan yang telah dibuat akan konvergen adalah dengan Matriks Jacobian, dimana:

$$J(x^{(n)}) = \begin{bmatrix} \frac{\partial g_1}{x_1} & \frac{\partial g_1}{x_2} & \cdots & \frac{\partial g_1}{x_n} \\ \frac{\partial g_2}{x_1} & \frac{\partial g_2}{x_2} & \cdots & \frac{\partial g_2}{x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial g_n}{x_1} & \frac{\partial g_n}{x_2} & \cdots & \frac{\partial g_n}{x_n} \end{bmatrix}$$

   Lalu, memasukkan nilai dari vektor tebakan awal ke dalam matriks Jacobian. Jika $||J(x^{(0)})||_\infty < 1$, maka dia akan konvergen, dan sebaliknya jika > 1.

2. Metode Newton

   Metode Newton menggunakan matriks Jacobian dan fungsi koordinat dari SPNL yang diberikan untuk mengaproksimasi nilai x_n, sehingga:

$$\Delta x^{(k)} = J^{-1}(x^{(k)}) \cdot -F(x^{(k)})$$

   dan

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

3. Metode Quasi-Newton (Broyden)

   Metode Quasi-Newton sekilas mirip dengan Metode Newton, dengan perbedaan pada metode ini, invers dari matriks Jacobiannya secara langsung tanpa melalui metode Determinan Adjoin.

   Cara menghitungnya adalah sebagai berikut:

   1 Definisikan $F(x^{(0)})$ dan $A_0 = J(x^{(0)})$
   2 Invers A_0 menjadi A_0^{-1}
   3 Hitung $x^{(1)} = x^{(0)} - A_0^{-1} \cdot F(x^{(0)})$
   4 Langkah Iteratif, $k = 1,2,\ldots$
      5 Definisikan $F(x^{(k)})$
      6 Hitung $y^{(k)} = F(x^{(k)}) - F(x^{(k-1)})$
      7 Hitung $s^{(k)} = x^{(k)} - x^{(k-1)}$
      8 Hitung $p = s_k^T \cdot A_{k-1}^{-1} \cdot y_k$
      9 Hitung Matriks $A_k^{-1} = A_{k-1}^{-1} + \frac{1}{p}\left[\left(s_k - A_{k-1}^{-1} \cdot y_k\right) \cdot s_1^T \cdot A_{k-1}^{-1}\right]$
      10 Hitung $\Delta x^{(k)} = A_k^{-1} \cdot F(x^{(k)})$
      11 Terakhir, hitung $x^{(k+1)} = x^{(k)} - \Delta x^{(k)}$
      12 Jika $||\Delta x^{(k)}||_\infty > \varepsilon$, ulangi langkah 5 - 12.

4. Metode *Steepest Descent*

   Cara menghitungnya adalah sebagai berikut:

1 Definisikan $F(x^{(k)})$ dan $J(x^{(k)})$

2 Hitung $\Delta g(x^{(k)}) = J(x^{(k)})^T \cdot F(x^{(k)})$

3 Hitung $z = -\Delta g(x^{(k)}) \cdot \alpha$, dimana α adalah *learning rate / stepsize*.

4 Terakhir, hitung $x^{(k+1)} = x^{(k)} + z$

5 Jika $||z||_\infty > \varepsilon$, ulangi iterasi.