Galat

Andaikan p^* adalah nilai penaksiran untuk p, maka galatnya adalah $|p - p^*|$ dan galat relatifnya adalah $\left|\frac{p-p^*}{p}\right|$. Bilangan p^* dikatakan mendekati p sampai dengan t angka penting jika $\left|\frac{p-p^*}{p}\right| \leq 5 \times 10^{-t}$.

Metode Bagi Dua (Bisection)

Metode bagi dua adalah metode sederhana dan stabil untuk menemukan akar suatu fungsi dalam interval $[a, b]$, di mana $f(a)$ dan $f(b)$ memiliki tanda yang berlawanan (artinya terdapat akar di dalam interval tersebut berdasarkan Teorema Nilai Antara).

1. Tentukan interval awal $[a, b]$ di mana $f(a) \cdot f(b) < 0$.
2. Hitung titik tengah $c = \frac{a+b}{2}$.
3. Evaluasi $f(c)$:
   - Jika $f(c) = 0$, maka c adalah akar.
   - Jika $f(a) \cdot f(c) < 0$, maka akar berada di interval $[a, c]$, dan $b = c$.
   - Jika $f(b) \cdot f(c) < 0$, maka akar berada di interval $[c, b]$, dan $a = c$.
4. Ulangi langkah 2 dan 3 hingga mencapai toleransi kesalahan yang diinginkan.

Metode ini konvergen secara linear, tetapi lambat dibandingkan metode lainnya.

Metode Regula Falsi

Metode Regula Falsi adalah variasi dari metode bagi dua, namun menggunakan garis lurus untuk mengaproksimasi akar lebih cepat. Sama seperti metode bagi dua, ini juga memerlukan $f(a) \cdot f(b) < 0$.

1. Tentukan interval awal $[a, b]$.
2. Hitung titik potong garis lurus antara $(a, f(a))$ dan $(b, f(b))$ sebagai c:
$$c = b - \frac{f(b) \cdot (b - a)}{f(b) - f(a)}$$
3. Evaluasi $f(c)$ dan perbarui interval $[a, b]$ seperti pada metode bagi dua.
4. Ulangi hingga mencapai toleransi kesalahan yang diinginkan.

Metode ini umumnya lebih cepat daripada metode bagi dua, tetapi bisa menjadi lambat jika salah satu sisi interval terus-menerus dipertahankan.

Metode Iterasi Titik Tetap

Metode iterasi titik tetap digunakan untuk menyelesaikan persamaan yang dapat dinyatakan dalam bentuk $x = g(x)$. Persamaan ini kemudian diiterasikan mulai dari tebakan awal x_0.

1. Pilih tebakan awal x_0.
2. Hitung iterasi berikutnya dengan rumus $x_{n+1} = g(x_n)$.
3. Ulangi hingga nilai x_{n+1} mendekati nilai x_n dengan toleransi tertentu.

Keberhasilan metode ini tergantung pada fungsi $g(x)$ yang harus memenuhi syarat tertentu agar konvergen.

Metode Newton-Raphson

Metode Newton-Raphson adalah metode yang sangat populer dan cepat untuk menemukan akar persamaan. Metode ini menggunakan turunan dari fungsi $f(x)$ untuk memperbaiki tebakan akar.

1. Pilih tebakan awal x_0.
2. Iterasi diperoleh dengan rumus:
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
3. Ulangi iterasi hingga nilai x_{n+1} mendekati nilai akar dengan toleransi yang diinginkan.

Metode ini memiliki kecepatan konvergensi kuadratik, tetapi dapat gagal jika turunan $f'(x)$ mendekati nol atau jika tebakan awal jauh dari akar sebenarnya.

Metode Secant

Metode secant adalah variasi dari metode Newton-Raphson yang tidak memerlukan perhitungan turunan. Sebagai gantinya, metode ini menggunakan aproksimasi numerik dari turunan.

1. Pilih dua tebakan awal x_0 dan x_1.
2. Iterasi dihitung sebagai berikut:
$$x_{n+1} = x_n - \frac{f(x_n) \cdot (x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$
3. Ulangi hingga mencapai toleransi kesalahan yang diinginkan.

Metode secant biasanya lebih cepat dari metode Newton-Raphson karena tidak memerlukan perhitungan turunan, tetapi bisa kurang stabil.

Metode Jacobi

Metode Jacobi adalah metode iteratif untuk menyelesaikan sistem persamaan linear dengan memisahkan setiap persamaan untuk menghitung variabel yang diinginkan.

1. Bentuk sistem persamaan dalam bentuk diagonal dominan:
$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}\right)$$
2. Pilih tebakan awal untuk setiap variabel.
3. Hitung solusi untuk iterasi berikutnya berdasarkan nilai variabel pada iterasi sebelumnya.
4. Ulangi proses hingga solusi konvergen atau mencapai toleransi yang diinginkan.

Metode Jacobi mudah diterapkan, namun lambat dalam konvergensi dibandingkan metode lain.

Metode Gauss-Seidel

Metode Gauss-Seidel adalah modifikasi dari metode Jacobi. Pada setiap iterasi, solusi variabel segera diperbarui dan digunakan untuk menghitung variabel lain pada iterasi yang sama.

1. Bentuk sistem persamaan dalam bentuk diagonal dominan.
2. Pilih tebakan awal.
3. Gunakan iterasi:
$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}\right)$$
4. Perbarui nilai variabel saat iterasi berlangsung.

Metode Gauss-Seidel biasanya lebih cepat dari Jacobi karena memanfaatkan nilai yang telah diperbarui secara langsung pada iterasi yang sama.

Dekomposisi LU

Dekomposisi LU adalah metode langsung untuk menyelesaikan sistem persamaan linear dengan memecah matriks A menjadi dua matriks segitiga, L (lower) dan U (upper):
$$A = LU$$
Solusi kemudian diperoleh dengan menyelesaikan dua sistem persamaan linear:
$$L\mathbf{y} = \mathbf{b} \quad \text{dan} \quad U\mathbf{x} = \mathbf{y}$$

Metode Dekomposisi LU Doolittle

Algoritma Doolittle memfaktorkan matriks A menjadi dua matriks, yaitu matriks segitiga bawah L dengan elemen diagonal sama dengan 1 dan matriks segitiga atas U. Langkah-langkah algoritma Doolittle adalah sebagai berikut:

1. Berikan input matriks A yang berukuran $n \times n$.
2. Tentukan matriks L dan U sedemikian rupa sehingga $A = LU$.
3. Untuk setiap $k = 1, 2, \ldots, n$:
   a) Hitung elemen-elemen pada $U[k, i]$ untuk $i = k, k+1, \ldots, n$ dengan rumus:
   $$U[k, i] = A[k, i] - \sum_{j=1}^{k-1} L[k, j]U[j, i]$$
   b) Hitung elemen-elemen pada $L[i, k]$ untuk $i = k+1, k+2, \ldots, n$ dengan rumus:
   $$L[i, k] = \frac{A[i, k] - \sum_{j=1}^{k-1} L[i, j]U[j, k]}{U[k, k]}$$
4. Ulangi langkah ini sampai seluruh elemen L dan U terhitung.

Metode Dekomposisi LU Crout

Berbeda dengan metode Doolittle, metode Crout menghasilkan matriks U dengan elemen diagonal 1. Matriks L memiliki elemen non-diagonal di bawah diagonal, sedangkan U memiliki elemen di atas diagonal.

1. Berikan input matriks A yang berukuran $n \times n$.
2. Tentukan matriks L dan U sedemikian rupa sehingga $A = LU$.
3. Untuk setiap $k = 1, 2, \ldots, n$:
   a) Hitung elemen-elemen pada $L[i, k]$ untuk $i = k, k+1, \ldots, n$ dengan rumus:
   $$L[i, k] = A[i, k] - \sum_{j=1}^{k-1} L[i, j]U[j, k]$$
   b) Hitung elemen-elemen pada $U[k, i]$ untuk $i = k+1, k+2, \ldots, n$ dengan rumus:
   $$U[k, i] = \frac{A[k, i] - \sum_{j=1}^{k-1} L[k, j]U[j, i]}{L[k, k]}$$
4. Ulangi langkah ini sampai seluruh elemen L dan U terhitung.

Metode Newton-Raphson (SPNL)

Metode Newton-Raphson adalah metode iteratif yang menggunakan pendekatan linier untuk menyelesaikan sistem persamaan non-linear. Misalkan terdapat sistem persamaan non-linear:
$$F(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = 0$$
Langkah-langkah Metode Newton-Raphson untuk sistem persamaan non-linear adalah sebagai berikut:

1. Mulai dengan tebakan awal $x^{(0)} = \left[x_i^{(0)}\right]$.
2. Hitung matriks Jacobian $J(x)$, yaitu turunan parsial dari setiap f_i terhadap x_j:
$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$
3. Perbarui solusi menggunakan iterasi Newton-Raphson:
$$x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})F(x^{(k)})$$
4. Ulangi langkah 2 dan 3 sampai konvergensi, yaitu hingga perubahan antara iterasi $x^{(k)}$ dan $x^{(k+1)}$ lebih kecil dari toleransi yang ditentukan.

Metode Iterasi (SPNL)

Tebakan awal yang diberikan sebagai:
$$\mathbf{x}^{(0)} = \left[x_i^{(0)}\right]$$
Sistem disusun ulang menjadi bentuk iterasi:
$$\begin{gathered} x_1 = F_1(x_1, x_2, \ldots, x_n) \\ x_2 = F_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ x_n = F_n(x_1, x_2, \ldots, x_n) \end{gathered}$$
Seperti pada metode Gauss-Seidel untuk sistem persamaan linear, metode iterasi ini juga memperbaharui setiap variabel x_i satu per satu. Hasil baru yang dihitung langsung digunakan dalam iterasi berikutnya, tanpa menunggu hasil iterasi seluruh variabel. Proses iterasi dapat dituliskan sebagai berikut:
$$\begin{gathered} x_1^{(k+1)} = F_1(x_1^{(k+1)}, x_2^{(k)}, \ldots, x_n^{(k)}) \\ x_2^{(k+1)} = F_2(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k)}) \\ \vdots \\ x_n^{(k+1)} = F_n(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}) \end{gathered}$$

1. Tentukan tebakan awal $\mathbf{x}^{(0)}$ untuk solusi.
2. Susun sistem persamaan non-linear ke dalam bentuk iteratif seperti yang dituliskan di atas.
3. Hitung iterasi berikutnya $\mathbf{x}^{(k+1)}$ menggunakan rumus:
$$x_i^{(k+1)} = F_i(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k)})$$
4. Ulangi proses ini sampai konvergensi dicapai, yaitu saat $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$ lebih kecil dari toleransi yang ditentukan ϵ.

Interpolasi Lagrange/Polinomial

Polinomial interpolasi Lagrange-nya adalah

$$f(x) = L_0(x)y_0 + L_1(x)y_1 + \cdots + L_n(x)y_n$$
$$= \sum_{k=0}^{n} L_k(x)y_k$$

dengan

$$L_k(x) = \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{(x - x_j)}{(x_k - x_j)}$$
$$= \frac{(x - x_0)}{(x_k - x_0)} \cdot \frac{(x - x_1)}{(x_k - x_1)} \cdot \ldots$$

Interpolasi Newton (Metode Beda)
Polinomial interpolasi Newton adalah

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots$$

dengan

$$a_0 = y_0,$$
$$a_1 = [x_1, x_0],$$
$$a_2 = \frac{[x_2, x_1] - [x_1, x_0]}{x_2 - x_0},$$
$$a_3 = \frac{[x_3, x_2, x_1] - [x_2, x_1, x_0]}{x_3 - x_0},$$
$$\vdots$$

Didefinisikan $s = \frac{x-x_0}{h}$

- **Maju**:

$$P_n(x) = f(x_0) + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f(x_0)$$

- **Mundur**:

$$P_n(x) = f[x_n] + \sum_{k=1}^{n} (-1)^k \binom{-s}{k} \nabla^k f(x_n)$$

Diferensiasi Numerik

RK-4 (Runge-Kutta Orde 4) (Optional: Pak Basuki)

- Diberikan nilai awal (x_0, y_0), nilai y_1 dapat dicari dengan formula:

$$y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

di mana:

$$k_1 = hf(x_0, y_0),$$
$$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right),$$
$$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right),$$
$$k_4 = hf(x_0 + h, y_0 + k_3),$$

dengan $h = x_1 - x_0$.

Integrasi Numerik

Kita definisikan $h = \frac{b-a}{n}$ dan n adalah banyaknya partisi yang kita inginkan.
Berturut-turut untuk hampiran kiri, kanan dan tengah

$$\int_a^b f(x)\,dx \approx h\left[y_0 + y_1 + \cdots + y_{n-1}\right] \quad \text{(Kiri)}$$

$$\int_a^b f(x)\,dx \approx h\,[y_1 + y_2 + \cdots + y_n] \quad \text{(Kanan)}$$

$$\int_a^b f(x)\,dx \approx h\left[y_1 + y_2^* + \cdots + y_{n-1}^*\right] \quad \text{(Tengah)}$$

dengan $y_i^* = f\left(\frac{x_k - x_{k-1}}{2}\right) = f(m_i)$.
Perkiraan error — untuk hampiran tengah — dapat dicari dengan cara,

$$|E_T| \leq \frac{(b-a)^3 K_2}{24n^2}$$

di mana K_2 adalah nilai maksimum dari turunan kedua fungsi yang diintegrasikan.
Untuk hampiran trapezium diperoleh

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\,[y_0 + 2y_1 + y_2] \quad (\text{jika } 0 \leq x \leq 2)$$

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[y_0 + 2\left(y_1 + y_2 + \cdots + y_{n-1}\right) + y_n\right]$$

Perkiraan error — untuk hampiran trapezoidal — dapat dicari dengan cara,

$$|E_T| \leq \frac{(b-a)^3 K_2}{12n^2}$$

di mana K_2 adalah nilai maksimum dari turunan kedua fungsi yang diintegrasikan.
Aturan Simpson 1/3 dan 3/8

$$S_{1/3} = \int_a^b f(x)\,dx \approx \frac{h}{3}\,[y_0 + 4y_1 + y_2] \quad (\text{jika } 0 \leq x \leq 2)$$

$$= \frac{h}{3}\left[y_0 + 4(y_1 + y_3 + y_5 + \cdots + y_{2n-1}) + 2(y_2 + y_4 + y_6 + \cdots + y_{2n-2}) + y_n\right]$$

Simpson 3/8

$$S_{3/8} = \int_a^b f(x)\,dx \approx \frac{3h}{8}\,[y_0 + 3y_1 + 3y_2 + y_3] \quad (\text{jika } 0 \leq x \leq 3)$$

$$= \frac{3h}{8}\left[y_0 + 3(y_1 + y_4 + y_7 + \ldots) + 3(y_2 + y_5 + y_8 + \ldots) + 2(y_3 + y_6 + y_9 + \ldots) + y_n\right]$$

Perkiraan error untuk hampiran Simpson 1/3 dapat dicari dengan cara:

$$\left|E_{S_{1/3}}\right| \leq \frac{(b-a)^5 K_2}{180n^4}$$

di mana K_2 adalah nilai maksimum dari turunan kedua fungsi yang diintegrasikan.
Kuadratur Gauss Diberikan,

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f(t)\,dt \quad \text{dengan } x = \frac{b-a}{2}t + \frac{a+b}{2}$$

serta $dx = \frac{b-a}{2}\,dt$. Sehingga, integralnya dapat diestimasi menggunakan:

$$\int_{-1}^{1} f(t)\,dt = 2f(0) \qquad (1)$$
$$= f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \qquad (2)$$
$$= \frac{5}{9}f\left(-\frac{\sqrt{3}}{5}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\frac{\sqrt{3}}{5}\right) \qquad (3)$$

Least Square Method atau Metode Regresi Untuk jarak kuadrat terkecil dicapai melalui:

$$S = \sum_{i=0}^{n} e_i^2 = \sum_{i=0}^{n} (y_i - \bar{y})^2$$

dengan (x_i, y_i) adalah pasangan data yang ada di indeks i dari total n pasangan data n buah data, sementara $\bar{y}$ adalah fungsi regresi yang ingin dicari.

Regresi Linear

Regresinya adalah $y = ax + b$ dan rumus awal menjadi:

$$S = \sum_{i=0}^{n} (y_i - ax_i - b)^2$$

Sehingga, supaya S minimum, kita buat

$$\frac{\partial S}{\partial a} = 0 \quad \text{dan} \quad \frac{\partial S}{\partial b} = 0$$

$$\frac{\partial S}{\partial a} = 2\sum_{i=0}^{n} (y_i - ax_i - b)(-x_i) = \sum x_i y_i = a\sum x_i^2 + b\sum x_i$$

$$\frac{\partial S}{\partial b} = 2\sum_{i=0}^{n} (y_i - ax_i - b)(-1) = \sum y_i = a\sum x_i + b\cdot n$$

Sehingga, kita dapat menulis matriks:

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}$$

Selanjutnya, dapat diselesaikan menggunakan aturan Cramer, sebagai berikut:

$$a = \frac{\begin{vmatrix} \sum x_i y_i & \sum x_i \\ \sum y_i & n \end{vmatrix}}{\begin{vmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{vmatrix}} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{\begin{vmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i & \sum y_i \end{vmatrix}}{\begin{vmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{vmatrix}} = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Regresi Eksponensial
Ada dua versi, yaitu:

$$y = ae^{bx} \qquad (1)$$

$$y = ax^b \qquad (2)$$

Untuk (1), kita ubah menjadi $\ln y = \ln a + bx$. Sementara (2), diubah menjadi $\ln y = \ln a + b\ln x$. Sehingga, bentuk matriks untuk (1):

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \begin{pmatrix} b \\ \ln a \end{pmatrix} = \begin{pmatrix} \sum x_i \ln y_i \\ \sum \ln y_i \end{pmatrix}$$

Sementara, untuk (2):

$$\begin{pmatrix} \sum (\ln x_i)^2 & \sum (\ln x_i) \\ \sum (\ln x_i) & n \end{pmatrix} \begin{pmatrix} b \\ \ln a \end{pmatrix} = \begin{pmatrix} \sum (\ln x_i) \ln y_i \\ \sum \ln y_i \end{pmatrix}$$

Kelanjutannya cukup menggunakan aturan Cramer.