DIKTAT KULIAH (3 sks)

MX 211: Metode Numerik

(*Revisi Terakhir: Juni 2009*)

Oleh:

Didit Budi Nugroho, M.Si.

Program Studi Matematika
Fakultas Sains dan Matematika
Universitas Kristen Satya Wacana

KATA PENGANTAR

Naskah ini ditulis ketika penulis mengajar Metode Numerik (MX 211) di Universitas Kristen Satya Wacana pada Semester 2 tahun 2008-2009 dan juga Trimester 2 tahun 2008-2009 di Soe, Nusa Tenggara Timur. Catatan ini membentuk naskah dasar untuk kuliah "Metode Numerik".

Tujuan yang ingin dicapai dari mata kuliah Metode Numerik adalah memahami konsep dasar metode numeris, mengetahui kekurangan dan kelebihan dari setiap metode, dan juga mampu untuk mengaplikasikannya. Karena itu, naskah ini difokuskan pada pemahaman konsep matematis dasar dan penyelesaian masalah menggunakan metode numerik dengan bantuan program MatLab. Beberapa metode disajikan dengan algoritma dalam *pseudocode* sehingga pembaca atau pengguna bisa mengimplementasikan dalam bahasanya sendiri. Di sini, analisis (seperti untuk galat) tidak diberikan secara mendalam.

Naskah ini memerlukan masukan dan saran dari pembaca demi perbaikan dan pengembangan secara terus menerus. Harapannya adalah bahwa naskah ini memberikan manfaat yang lebih dalam pengajaran Metode Numerik.

Salatiga, Juli 2009

Didit B. Nugroho

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab 1

Pendahuluan

Tujuan Pembelajaran:
- Mengetahui apa yang dimaksud dengan metode numerik.
- Mengetahui kenapa metode numerik perlu dipelajari.
- Mengetahui langkah-langkah penyelesaian persoalan numerik.

Metode numerik merupakan teknik untuk menyelesaikan masalah matematika dengan pengoperasian aritmatika (hitungan).

Beberapa alasan mengapa kita harus mempelajari metode numerik:

1. Metode numerik merupakan alat bantu pemecahan masalah matematika yang sangat ampuh. Metode numerik mampu menangani sistem persamaan besar, ketidaklinearan, dan geometri yang rumit yang dalam praktek rekayasa seringkali tidak mungkin dipecahkan secara analitik.

2. Di pasaran banyak tersedia program aplikasi numerik komersil. Penggunaan aplikasi tersebut menjadi lebih berarti bila kita memiliki pengetahuan metode numerik agar kita dapat memahami cara paket tersebut menyelesaikan persoalan.

3. Kita dapat membuat sendiri program komputer tanpa harus membeli paket programnya. Seringkali beberapa persoalan matematika tidak selalu dapat diselesaikan oleh program aplikasi. Sebagai contoh, terdapat program aplikasi tertentu yang tidak dapat dipakai untuk menghitung integrasi lipat dua, atau lipat tiga. Mau tidak mau, kita harus menulis sendiri programnya. Untuk itu, kita harus mempelajari cara pemecahan integral lipat dua atau lebih dengan metode numerik.

4. Metode numerik menyediakan sarana untuk memperkuat kembali pemahaman matematika, karena metode numerik ditemukan dengan cara menyederhanakan matematika yang lebih tinggi menjadi operasi matematika yang mendasar.

Langkah-langkah penyelesaian persoalan numerik:

1. Identifikasi masalah.

2. Memodelkan masalah secara matematis.

3. Identifikasi metode numerik yang diperlukan untuk menyelesaikan masalah.

4. Implementasi metode dalam komputer.
5. Analisis hasil akhir: implementasi, metode, model dan masalah.

Jenis-jenis persoalan matematika yang akan diselesaikan secara numerik dalam naskah ini:

1. Pencarian akar-akar persamaan tak linear.
2. Metode iteratif untuk penyelesaian sistem persamaan linear
3. Interpolasi linear, kuadrat, Newton, dan spline.
4. Regresi kuadrat terkecil.
5. Diferensiasi numerik.
6. Persamaan diferensial biasa.
7. Integrasi numerik.

Bab 2

Galat

Tujuan Pembelajaran:
- Mengetahui definisi dan jenis-jenis galat.
- Mengetahui bagaimana menghitung galat.

2.1 Pengertian Galat

Penyelesaian secara numerik dari suatu persamaan matematik hanya memberikan nilai perkiraan (hampiran) yang mendekati nilai eksak (yang sebenarnya) dari penyelesaian analitis. Berarti dalam penyelesaian numerik tersebut terdapat galat terhadap nilai eksak.

Ada tiga macam galat:

1. Galat bawaan, terjadi karena kekeliruan dalam menyalin data, salah membaca skala, atau karena kurangnya pengertian mengenai hukum-hukum fisik dari data yang diukur.

2. Galat pembulatan (*round-off error*), terjadi karena tidak diperhitungkannya beberapa angka terakhir dari suatu bilangan. Sebagai contoh, 3.1415926 dapat dibulatkan menjadi 3.14.

3. Galat pemotongan (*truncation error*), terjadi karena tidak dilakukannya hitungan sesuai dengan prosedur matematik yang benar. Sebagai contoh, turunan pertama dari $V(t)$ terhadap t dihitung dengan prosedur

$$\frac{dV}{dt} = \frac{\Delta V}{\Delta t} = \frac{V(t_{i+1}) - V(t_i)}{t_{i+1} - t_i}.$$

   Contoh lain yaitu pengambilan beberapa suku awal dari deret Taylor:

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 \\ &\quad + \frac{f'''(x_i)}{3!}(x_{i+1} - x_i)^3 + \ldots + \frac{f^{(n)}(x_i)}{n!}(x_{i+1} - x_i)^n + R_n \end{aligned}$$

   dengan $R_n = \dfrac{f^{(n+1)}(c)}{(n+1)!}(x_{i+1} - x_i)^{n+1}$ dimana $c \in [x_i, x_{i+1}]$.

Dari deret Taylor di atas, dipunyai hampiran orde-0:

$$f(x_{i+1}) \approx f(x_i),$$

hampiran orde-1:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i),$$

hampiran orde-2:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2.$$

Contoh 2.1 *Diberikan fungsi*

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2.$$

Turunan pertama dan kedua dari $f(x)$ *berturut-turut yaitu*

$$\begin{aligned} f'(x) &= -0.4x^3 - 0.45x^2 - x - 0.25, \\ f''(x) &= -1.2x^2 - 0.9x - 1, \end{aligned}$$

Dimulai dari $x = 0$*, diperoleh hampiran orde-1 untuk* $f(1)$*:*

$$f(1) \approx f(0) + f'(0)(1 - 0) = 1.2 - 0.25 = 0.95,$$

dan hampiran orde-2 untuk $f(1)$*:*

$$f(1) \approx f(0) + f'(0)(1 - 0) + \frac{f''(0)}{2!}(1 - 0)^2 = 1.2 - 0.25 - 0.5 = 0.9.$$

2.2 Penghitungan Galat

Untuk galat pembulatan dan pemotongan, hubungan antara hasil yang eksak dengan hampirannya dapat dirumuskan oleh

$$\text{nilai eksak} = \text{hampiran} + \text{galat}.$$

Dengan menyusun kembali persamaan di atas, diperoleh

$$E_s = \text{galat} = \text{nilai eksak - hampiran}$$

dimana subskrip s menunjukkan bahwa galat adalah galat sejati.

Kelemahan dari definisi di atas adalah bahwa tingkat besaran dari nilai yang diperiksa sama sekali tidak diperhatikan. Sebagai contoh, galat satu sentimeter jauh lebih berarti jika yang diukur adalah paku ketimbang jembatan. Salah satu cara untuk memperhitungkan besarnya besaran yang sedang dievaluasi adalah dengan menormalkan galat terhadap nilai eksak, yaitu

$$\text{galat relatif} = \frac{\text{nilai eksak - hampiran}}{\text{nilai eksak}}.$$

Galat relatif dapat juga dikalikan dengan 100% agar dapat dinyatakan sebagai

$$\epsilon_s = \text{persen galat relatif} = \frac{\text{nilai eksak - hampiran}}{\text{nilai eksak}} \times 100\%.$$

Dicatat bahwa untuk metode numerik, nilai eksak hanya akan diketahui jika fungsi yang ditangani dapat diselesaikan secara eksak. Jika tidak demikian, maka alternatifnya adalah menormalkan galat dengan menggunakan hampiran terbaik yang tersedia dari nilai eksak, yaitu terhadap hampiran itu sendiri, seperti yang dirumuskan oleh

$$\begin{aligned} \epsilon_h &= \frac{\text{galat hampiran}}{\text{hampiran}} \times 100\% \\ &= \frac{\text{hampiran sekarang } - \text{ hampiran sebelumnya}}{\text{hampiran sekarang}} \times 100\% \end{aligned}$$

dengan subskrip h menunjukkan bahwa galat dinormalkan terhadap nilai hampiran.

Bab 3

Pencarian Akar

Tujuan Pembelajaran:
- Mengetahui metode-metode pencarian akar dari persamaan tak linear.
- Mengetahui keunggulan dan kelemahan dari setiap metode.
- Mengaplikasikan metode-metode pencarian akar untuk persamaan yang sama.

3.1 Akar-akar Persamaan

Akar atau pembuat nol dari suatu fungsi adalah nilai-nilai dari variabel bebas yang membuat fungsi bernilai nol. Sebagai contoh, penyelesaian analitik untuk fungsi kuadratik $f(x) = ax^2 + bx + c = 0$ diberikan oleh

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Contoh lain, kita tidak mungkin mendapatkan suatu penyelesaian analitik untuk fungsi $f(x) = e^x - x = 0$.

Secara umum, terdapat dua kelompok metode untuk pencarian akar dari persamaan tak linear:

1. Metode Pengurung.

   Sesuai dengan namanya, tebakan akar dalam metode ini selalu berada "dalam kurung" atau berada pada kedua sisi dari nilai akar. Karena itu, di sini diperlukan dua tebakan awal untuk akar. Metode ini mempunyai suatu keunggulan yaitu konvergen (makin lama makin mendekati nilai sebenarnya), dan mempunyai kelemahan yaitu konvergensinya relatif lambat. Contoh dari metode pengurung yaitu metode bagi dua (*bisection*) dan metode posisi palsu (*false position*).

2. Metode Terbuka.

   Dalam metode ini, pencarian dimulai dari suatu nilai tunggal variabel bebas, atau dua nilai yang tidak perlu mengurung akar. Metode ini mempunyai suatu kelemahan yaitu tidak selalu konvergen, tetapi mempunyai keunggulam yaitu jika konvergen maka konvergensinya lebih cepat daripada metode pengurung. Contoh dari metode terbuka yaitu metode iterasi titik tetap (*fixed-point iteration*), metode Newton-Raphson, dan metode garis potong (*secant*).

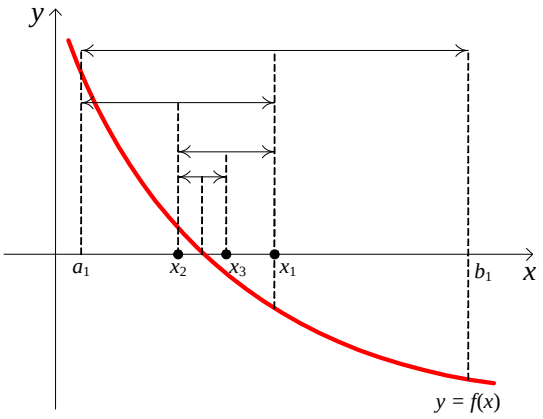
Penggunaan metode pengurung didasarkan pada teorema berikut ini.

Teorema 3.1 *Diberikan* $f : [a, b] \to \mathbb{R}$ *adalah kontinu, dimana* $a, b \in \mathbb{R}$ *dan* $a < b$. *Jika* $f(a) \cdot f(b) < 0$, *maka terdapat* $c \in (a, b)$ *sedemikian sehingga* $f(c) = 0$.

3.2 Metode Bagi Dua

Metode ini mengasumsikan bahwa fungsi $f(x)$ adalah kontinu pada interval $[a_1, b_1]$, serta $f(a_1)$ dan $f(b_1)$ mempunyai tanda berlawanan, artinya $f(a_1) \cdot f(b_1) < 0$. Karena itu terdapat minimal satu akar pada interval $[a_1, b_1]$.

Idenya adalah interval selalu dibagi dua sama lebar. Jika fungsi berubah tanda sepanjang suatu subinterval, maka letak akarnya kemudian ditentukan ada di tengah-tengah subinterval. Proses ini diulangi untuk memperoleh hampiran yang diperhalus. (Lihat Gambar 3.1.)



Gambar 3.1: Ilustrasi grafis untuk akar hampiran dalam metode bagi dua.

Dicatat bahwa terdapat beberapa kriteria penghentian pencarian akar jika diberikan suatu toleransi keakuratan ϵ, yaitu

$$\epsilon_s < \epsilon,\ \epsilon_h < \epsilon, \text{ atau } n_{\text{maks}} = N \in \mathbb{N}.$$

Proses untuk metode bagi dua diberikan seperti dalam Algoritma 1.

Contoh 3.2 *Selesaikan persamaan* $x^2 - 3 = 0$ *dalam interval* $[1, 2]$ *menggunakan metode bagi dua sampai lima iterasi.*

Penyelesaian. Proses metode bagi dua adalah seperti berikut ini.
Iterasi 1:

$$\begin{aligned} a_1 &= 1 \implies f(a_1) = -2, \\ b_1 &= 2, \\ x_1 &= \frac{a_1 + b_1}{2} = \frac{1+2}{2} = 1.5 \implies f(x_1) = -0.75. \end{aligned}$$

Algorithm 1 Algoritma Metode Bagi Dua

Masukan:

fungsi kontinu: $f(x)$

interval yang mengurung akar: $[a_1, b_1]$

maksimum iterasi: $N \in \mathbb{N}$

toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-5}$

Penghitungan Inti: Ketika $n \leq N$ dan $\epsilon_h \geq \epsilon$,

Hitung: $x_n = \dfrac{a_n + b_n}{2}$.

Tentukan subinterval mana yang akan mengurung akar:

a) Jika $f(a_n) \cdot f(x_n) < 0$, maka $a_{n+1} = a_n$, $b_{n+1} = x_n$.

b) Jika $f(a_n) \cdot f(x_n) > 0$, maka $a_{n+1} = x_n$, $b_{n+1} = b_n$.

c) Jika $f(a_n) \cdot f(x_n) = 0$, maka diperoleh akar sama dengan x_n. Berhenti.

Hitung: $\epsilon_h = \left| \dfrac{x_n - x_{n-1}}{x_n} \right| \times 100\%$, $n \geq 1$

Hasil akhir: akar x_n sedemikian sehingga $f(x_n) \approx 0$.

Iterasi 2: Diamati bahwa $f(a_1) \cdot f(x_1) > 0$, maka

$$\begin{aligned}
a_2 &= x_1 = 1.5 \implies f(a_2) = -0.75, \\
b_2 &= b_1 = 2, \\
x_2 &= \frac{a_2 + b_2}{2} = \frac{1.5 + 2}{2} = 1.75 \implies f(x_2) = 0.0625, \\
\epsilon_2 &= \left| \frac{x_2 - x_1}{x_2} \right| \cdot 100\% = \left| \frac{1.75 - 1.5}{1.75} \right| \cdot 100\% = 14.29\%.
\end{aligned}$$

Iterasi 3: Diamati bahwa $f(a_2) \cdot f(x_2) < 0$, maka

$$\begin{aligned}
a_3 &= a_2 = 1.5 \implies f(a_3) = -0.75, \\
b_3 &= x_2 = 1.75, \\
x_3 &= \frac{a_3 + b_3}{2} = \frac{1.5 + 1.75}{2} = 1.625 \implies f(x_3) = -0.3594, \\
\epsilon_3 &= \left| \frac{x_3 - x_2}{x_3} \right| \cdot 100\% = \left| \frac{1.625 - 1.75}{1.625} \right| \cdot 100\% = 7.69\%.
\end{aligned}$$

Iterasi 4: Diamati bahwa $f(a_3) \cdot f(x_3) > 0$, maka

$$\begin{aligned}
a_4 &= x_3 = 1.625 \implies f(a_4) = -0.3594, \\
b_4 &= b_3 = 1.75, \\
x_4 &= \frac{a_4 + b_4}{2} = \frac{1.625 + 1.75}{2} = 1.6875 \implies f(x_4) = -0.1523, \\
\epsilon_4 &= \left| \frac{x_4 - x_2}{x_4} \right| \cdot 100\% = \left| \frac{1.6875 - 1.625}{1.6875} \right| \cdot 100\% = 3.7\%.
\end{aligned}$$

Iterasi 5: Diamati bahwa $f(a_4) \cdot f(x_4) > 0$, maka

$$\begin{aligned} a_5 &= x_4 = 1.6875 \implies f(a_5) = -0.1523, \\ b_5 &= b_4 = 1.75, \\ x_5 &= \frac{a_5 + b_5}{2} = \frac{1.6875 + 1.75}{2} = 1.7187 \implies f(x_5) = -0.0459, \\ \epsilon_5 &= \left|\frac{x_5 - x_4}{x_5}\right| \cdot 100\% = \left|\frac{1.7187 - 1.6875}{1.7187}\right| \cdot 100\% = 1.82\%. \end{aligned}$$

Jadi pada iterasi ke-5 diperoleh akar hampiran $x = 1.7187$. ▼

Dalam MatLab, Algoritma 1 untuk metode bagi dua diimplementasikan dalam fungsi `bagidua()` berikut ini.

```
function [x,galat] = BagiDua(f,X,N,tol)
% BagiDua  Menyelesaikan persamaan f(x) = 0 menggunakan metode bagi dua.
%
% Input:  f = fungsi dari x, gunakan fungsi inline('ekspresi','x')
%         X = [a b] = vektor titik-titik ujung interval dengan a < b
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: x = akar hampiran yang memenuhi kriteria
%         galat = persen galat relatif
% ---PENGHITUNGAN INTI:
if nargin < 4, tol = 1e-3; end %dinamakan BagiDua(f,X,N)
if nargin < 3, N = 100; end %dinamakan BagiDua(f,X)
a = X(1); % batas kiri interval
b = X(2); % batas kanan interval
x = (a+b)/2; % hampiran awal
n = 1; galat = 1; % supaya iterasi pertama dapat dikerjakan
while ( n <= N & galat > tol ) % kriteria penghentian
    if f(a)*f(x)<0
        b = x;
    elseif f(a)*f(x)>0
        a = x;
    else
        break
    end
    xnew = (a+b)/2; % titik tengah interval
    galat = abs((xnew - x)/xnew)*100; % galat relatif
    x = xnew;
    n = n+1;
end
```

Dimisalkan x_s adalah akar sejati dari $f(x) = 0$. Dicatat bahwa kita dapat menurunkan batas galat untuk metode bagi dua seperti berikut ini.

$$\begin{aligned} |x_s - x_{n+1}| &\leq |b_{n+1} - a_{n+1}| = \frac{1}{2}|b_n - a_n| = \left(\frac{1}{2}\right)^2 |b_{n-1} - a_{n-1}| = \ldots \\ &= \left(\frac{1}{2}\right)^n |b_1 - a_1| \end{aligned}$$

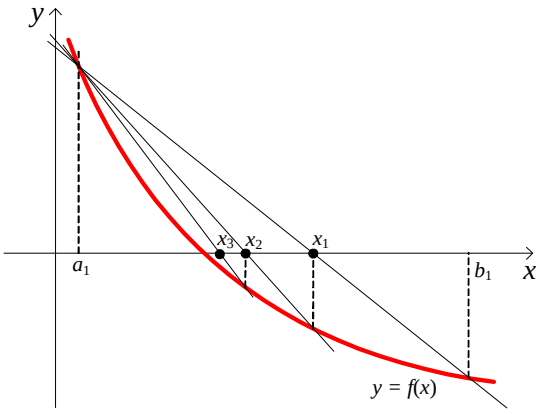
Ini berarti bahwa telah ditunjukkan $|x_s - x_{n+1}| \leq k\,|x_s - x_n|$, dengan $k = \frac{1}{2}$. Lebih lanjut, ini dinamakan konvergensi linear dan k dinamakan konstanta galat asimtotik.

Dalam contoh di atas, banyaknya langkah yang diperlukan untuk menjamin bahwa galat kurang dari 10^{-3} dihitung seperti berikut:

$$\begin{aligned}\left(\frac{1}{2}\right)^n |2-1| \quad &\leq \quad 10^{-3} \implies \left(\frac{1}{2}\right)^n \leq 10^{-3} \implies 2^n \geq 10^3 \\ &\implies \quad n\ln(2) \geq 3\ln(10) \implies n \geq 10.\end{aligned}$$

3.3 Metode Posisi Palsu

Metode posisi palsu adalah metode pencarian akar persamaan dengan memanfaatkan kemiringan dan selisih tinggi dari dua titik batas interval yang mengurung akar. Metode ini merupakan salah satu alternatif untuk mempercepat konvergensi.



Gambar 3.2: Ilustrasi grafis untuk akar hampiran dalam metode posisi palsu.

Idenya adalah menghitung akar (yang merupakan titik ujung interval baru) yang merupakan absis untuk titik potong antara sumbu x dengan garis lurus yang melalui kedua titik yang absisnya adalah titik-titik ujung interval lama. (Lihat Gambar 3.2) Rumus untuk mencari akar adalah sebagai berikut. Diasumsikan bahwa fungsi $f(x)$ adalah kontinu pada interval $[a_n, b_n]$, dan $f(a_n) \cdot f(b_n) < 0$. Garis yang melalui titik $(a_n, f(a_n))$ dan $(b_n, f(b_n))$ mempunyai persamaan

$$y - f(a_n) = \frac{f(b_n) - f(a_n)}{b_n - a_n}(x - a_n).$$

Garis memotong sumbu x jika $y = 0$, sehingga diperoleh titik absis sebagai hampiran akar yaitu

$$x_n = a_n - \frac{b_n - a_n}{f(b_n) - f(a_n)} f(a_n). \tag{3.1}$$

Proses untuk metode posisi palsu adalah seperti metode bagi dua tetapi penghitungan x_n menggunakan rumus (3.1).

Contoh 3.3 *Selesaikan persamaan* $x^2 - 3 = 0$ *dalam interval* $[1, 2]$ *menggunakan metode posisi palsu sampai lima iterasi.*

Penyelesaian. Berikut ini adalah tabel penyelesaian sampai lima iterasi.

n	a_n	b_n	x_n	$f(x_n)$	$\epsilon_n = \left\lvert\frac{x_n - x_{n-1}}{x_n}\right\rvert \%$
1	1	2	1.6667	−0.22222222	–
2	1.66667	2	1.7273	−0.01652892	3.50877192
3	1.72727	2	1.7317	−0.00118976	0.25608194
4	1.73170	2	1.7320	−0.00008543	0.01840773
5	1.73202	2	1.7321	−0.00000613	0.00132172
6	1.73204	2	1.7321	−0.00000044	0.00009489

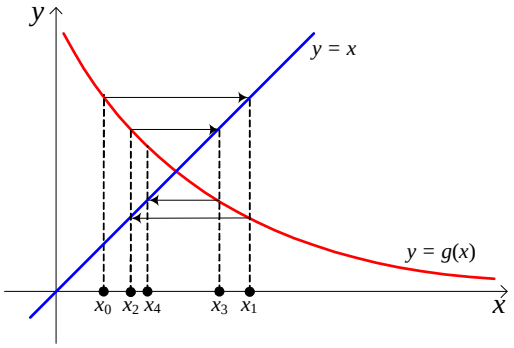
Jadi pada iterasi ke-6 diperoleh akar hampiran $x = 1.7320$ dengan $f(x) = -0.000006$. ▼

3.4 Metode Iterasi Titik Tetap

Metode iterasi titik tetap adalah metode yang memisahkan x sedemikian sehingga $f(x) = 0$ ekivalen dengan $x = g(x)$. Selanjutnya p adalah suatu akar dari $f(x)$ jika hanya jika p adalah suatu titik tetap dari $g(x)$. Kita mencoba untuk menyelesaikan $x = g(x)$ dengan menghitung

$$x_n = g(x_{n-1}), n = 1, 2, ...$$

dengan menggunakan tebakan awal x_0. Ilustrasi grafis untuk penyelesaian x_n diberikan oleh Gambar 3.3



Gambar 3.3: Ilustrasi grafis untuk akar hampiran dalam metode iterasi titik tetap $y = x$, $y = g(x)$.

Proses untuk metode bagi dua diberikan seperti dalam Algoritma 2.

Contoh 3.4 *Gunakan metode iterasi titik tetap sampai lima iterasi untuk menyelesaikan* $x^2 - 3 = 0$ *dengan tebakan awal* $x_0 = 1.5$.

Algorithm 2 Algoritma Iterasi Titik Tetap

Masukan:

fungsi kontinu: $f(x)$, $g(x)$
tebakan awal: x_0
maksimum iterasi: $N \in \mathbb{N}$
toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-5}$

Penghitungan Inti: Ketika $n \leq N$ dan $\epsilon_h \geq \epsilon$,

Hitung: $x_n = g(x_{n-1})$.
Hitung: $\epsilon_h = \left|\dfrac{x_n - x_{n-1}}{x_n}\right| \times 100\%$

Hasil akhir: akar x_n sedemikian sehingga $f(x_n) \approx 0$.

Penyelesaian. Persamaan dapat diubah ke beberapa bentuk:

$$\begin{aligned}
&\text{Kasus 1:} && x = \frac{3}{x} = g_1(x) \implies x_n = \frac{3}{x_{n-1}} \\
&\text{Kasus 2:} && x = x - \left(x^2 - 3\right) = g_2(x) \implies x_n = x_{n-1} - \left(x_{n-1}^2 - 3\right) \\
&\text{Kasus 3:} && x = x - \frac{x^2 - 3}{2} = g_3(x) \implies x_n = x_{n-1} - \frac{x_{n-1}^2 - 3}{2}
\end{aligned}$$

Penyelesaian untuk Kasus 3:
Iterasi 1:

$$\begin{aligned}
x_1 &= x_0 - \frac{x_0^2 - 3}{2} = 1.5 - \frac{1.5^2 - 3}{2} = 1.875, \\
\epsilon_1 &= \left|\frac{1.875 - 1.5}{1.875}\right| \cdot 100\% = 20\%.
\end{aligned}$$

Iterasi 2:

$$\begin{aligned}
x_2 &= x_1 - \frac{x_1^2 - 3}{2} = 1.875 - \frac{1.875^2 - 3}{2} = 1.6172, \\
\epsilon_2 &= \left|\frac{1.6172 - 1.875}{1.6172}\right| \cdot 100\% = 15.94\%.
\end{aligned}$$

Iterasi 3:

$$\begin{aligned}
x_3 &= x_2 - \frac{x_2^2 - 3}{2} = 1.6172 - \frac{1.6172^2 - 3}{2} = 1.8095, \\
\epsilon_3 &= \left|\frac{1.8095 - 1.6172}{1.8095}\right| \cdot 100\% = 10.63\%.
\end{aligned}$$

Iterasi 4:

$$\begin{aligned}
x_4 &= x_3 - \frac{x_3^2 - 3}{2} = 1.8095 - \frac{1.8095^2 - 3}{2} = 1.6723, \\
\epsilon_4 &= \left|\frac{1.6723 - 1.8095}{1.6723}\right| \cdot 100\% = 8.20\%.
\end{aligned}$$

Iterasi 5:

$$\begin{aligned} x_5 &= x_4 - \frac{x_4^2 - 3}{2} = 1.6723 - \frac{1.6723^2 - 3}{2} = 1.7740, \\ \epsilon_4 &= \left| \frac{1.7740 - 1.6723}{1.7740} \right| \cdot 100\% = 5.73\%. \end{aligned}$$

Berikut ini adalah penyelesaian sampai lima iterasi untuk ketiga kasus.

n	x_n		
	Kasus 1	Kasus 2	Kasus 3
0	1.5	1.5000	1.5
1	2	2.2500	1.875
2	1.5	0.1875	1.6172
3	2	3.1523	1.8095
4	1.5	−3.7849	1.6723
5	2	−15.1106	1.7740

Terlihat bahwa Kasus 1 dan Kasus 2 adalah divergen, tetapi Kasus 3 adalah konvergen. ▼

Berikut ini diberikan kriteria kekonvergenan dari metode iterasi titik tetap.

Teorema 3.5 *Diambil* $k = \max_{a \le x \le b} |g'(x)|$. *Metode iterasi titik tetap adalah konvergen jika hanya jika* $k < 1$.

Bukti. Dimisalkan x_s adalah akar sejati dari $f(x) = 0$, maka

$$|x_s - x_n| = |g(x_s) - g(x_{n-1})| = \left| g'(\zeta)(x_s - x_{n-1}) \right| \le k |x_s - x_{n-1}|$$

berdasarkan Teorema Nilai Rata-rata. Karena itu

$$|x_s - x_n| \le k |x_s - x_{n-1}| \le k^2 |x_s - x_{n-2}| \le \cdots \le k^n |x_s - x_0| .$$

■

Kita mencatat:

1. Telah ditunjukkan bahwa $|x_s - x_n| \le k |x_s - x_{n-1}|$, yang berarti metode iterasi titik tetap adalah konvergen secara linear.

2. Karena $k \approx |g'(x_s)|$, kita akan memilih fungsi iterasi $g(x_s)$ sedemikian sehingga $|g'(x_s)|$ adalah kecil.

   Dari contoh sebelumnya dipunyai $x_s = \sqrt{3} = 1.73205$, maka

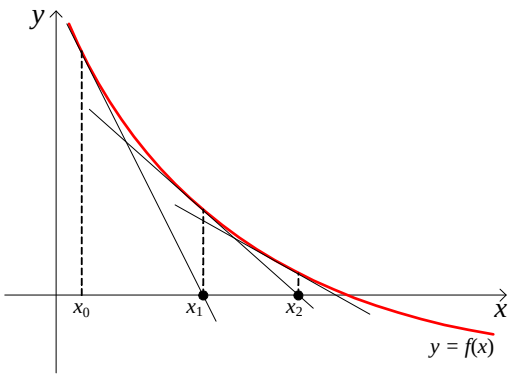
   $$\begin{aligned} &\text{Kasus 1:} && g'(x) = -\frac{3}{x^2} \implies \left| g'(x_s) \right| = 1 : \text{divergen}, \\ &\text{Kasus 2:} && g'(x) = 1 - 2x \implies \left| g'(x_s) \right| = 2.4641 : \text{divergen}, \\ &\text{Kasus 3:} && g'(x) = 1 - x \implies \left| g'(x_s) \right| = 0.73025 : \text{konvergen}. \end{aligned}$$

Dalam MatLab, Algoritma 2 untuk metode iterasi titik tetap diimplementasikan dalam fungsi `TitikTetap()` berikut ini.

```
function [x,galat] = TitikTetap(g,x0,N,tol)
% TitikTetap  Menyelesaikan persamaan x = g(x), ekivalen dengan f(x)=0,
%             menggunakan metode iterasi titik tetap.
%
% Input:  g = fungsi dari x, gunakan fungsi inline('ekspresi','x')
%         x0 = tebakan awal
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: x = akar hampiran yang memenuhi kriteria
%         galat = persen galat relatif
% ---PENGHITUNGAN INTI:
if nargin < 4, tol = 1e-3; end
if nargin < 3, N = 100; end
n = 1; galat = 1;
while ( n <= N & galat > tol )
    x = g(x0);
    galat = abs((x - x0)/x)*100;
    x0 = x;
    n = n+1;
end
```

3.5 Metode Newton-Raphson

Metode Newton-Raphson adalah metode pendekatan yang menggunakan satu titik awal dan mendekatinya dengan memperhatikan kemiringan kurva pada titik tersebut. Penjelasan grafis mengenai metode ini adalah seperti dalam Gambar 3.4.



Gambar 3.4: Ilustrasi grafis untuk akar hampiran dalam metode Newton-Raphson.

Diasumsikan bahwa fungsi $f(x)$ adalah kontinu. Idenya adalah menghitung akar yang merupakan titik potong antara sumbu x dengan garis singgung pada kurva di titik $(x_{n-1}, f(x_{n-1}))$. Kemiringan kurva di titik tersebut adalah $f'(x_{n-1})$, sehingga garis singgung mempunyai persamaan

$$y - f(x_{n-1}) = f'(x_{n-1})(x - x_{n-1}).$$

Karena itu diperoleh akar hampiran dengan mengambil $y = 0$, yaitu

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}. \tag{3.2}$$

Algorithm 3 Algoritma Metode Newton-Raphson

Masukan:

fungsi kontinu: $f(x)$, $f'(x)$
tebakan awal: x_0
maksimum iterasi: $N \in \mathbb{N}$
toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-5}$

Penghitungan Inti: Ketika $n \leq N$ dan $\epsilon_h \geq \epsilon$,

Hitung: $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$.
Hitung: $\epsilon_h = \left|\frac{x_n - x_{n-1}}{x_n}\right| \times 100\%$

Hasil akhir: akar x_n sedemikian sehingga $f(x_n) \approx 0$.

Contoh 3.6 *Gunakan metode Newton-Raphson untuk menyelesaikan* $x^2 - 3 = 0$ *dengan tebakan awal* $x_0 = 1.5$.

Penyelesaian. Karena $f(x) = x^2 - 3$, $f'(x) = 2x$ dan dibentuk rumus pencarian akar:

$$x_n = x_{n-1} - \frac{x_{n-1}^2 - 3}{2x_{n-1}}, n = 1, 2, ...$$

Iterasi 1:

$$\begin{aligned} x_1 &= x_0 - \frac{x_0^2 - 3}{2x_0} = 1.5 - \frac{1.5^2 - 3}{2 \cdot 1.5} = 1.75, \\ \epsilon_1 &= \left|\frac{1.75 - 1.5}{1.75}\right| \cdot 100\% = 14.2857142\%. \end{aligned}$$

Iterasi 2:

$$\begin{aligned} x_2 &= x_1 - \frac{x_1^2 - 3}{2x_1} = 1.75 - \frac{1.75^2 - 3}{2 \cdot 1.75} = 1.73214, \\ \epsilon_2 &= \left|\frac{1.73214 - 1.75}{1.73214}\right| \cdot 100\% = 1.0309278\%. \end{aligned}$$

Iterasi 3:

$$\begin{aligned} x_3 &= x_2 - \frac{x_2^2 - 3}{2x_2} = 1.73214 - \frac{1.73214^2 - 3}{2 \cdot 1.73214} = 1.73205, \\ \epsilon_3 &= \left|\frac{1.73205 - 1.73214}{1.73205}\right| \cdot 100\% = 0.0053143\%. \end{aligned}$$

Iterasi 4:

$$\begin{aligned} x_4 &= x_3 - \frac{x_3^2 - 3}{2x_3} = 1.73205 - \frac{1.73205^2 - 3}{2 \cdot 1.73205} = 1.73205, \\ \epsilon_4 &= \left| \frac{1.73205 - 1.73205}{1.73205} \right| \cdot 100\% = 0\%. \end{aligned}$$

Jadi pada iterasi ke-4 diperoleh akar hampiran $x = 1.73205$. ▼

Metode Newton-Raphson merupakan suatu contoh dari metode iterasi titik tetap, $x_n = g(x_{n-1})$, dimana fungsi iterasinya adalah

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Dari sini diperoleh bahwa

$$g'(x_s) = 1 - \frac{[f'(x_s)]^2 - f(x_s) \cdot f''(x_s)}{[f'(x_s)]^2} = 0$$

dengan mengingat bahwa $f(x_s) = 0$ dan $f'(x_s) \neq 0$. Ini mengakibatkan bahwa metode Newton adalah konvergen lebih cepat daripada secara linear; pada kenyataannya dipunyai

$$|x_s - x_n| \leq C |x_s - x_{n-1}|^2$$

yaitu konvergen kuadratik.

Dalam MatLab, Algoritma 3 untuk metode Newton-Raphson diimplementasikan dalam fungsi `NewRap()` berikut ini.

```
function [x,galat] = NewRap(f,f1,x0,N,tol)
% NewtonRaphson  Menyelesaikan persamaan f(x) = 0 menggunakan metode Newton-Raphson.
%
% Input:  f = fungsi dari x, gunakan fungsi inline('ekspresi','x')
%         f1 = turunan pertama dari f(x), cari dengan fungsi \QTR{bf}{diff}
%         x0 = tebakan awal
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: x = akar hampiran yang memenuhi kriteria
%         galat = persen galat relatif
% ---PENGHITUNGAN INTI:
if nargin < 4, tol = 1e-3; end
if nargin < 3, N = 100; end
n = 1; galat = 1;
while ( n <= N & galat > tol )
    x = x0-f(x0)/f1(x0); % persamaan (3.2)
    galat = abs((x - x0)/x)*100;
    x0 = x;
    n = n+1;
end
```

3.6 Metode Garis Potong

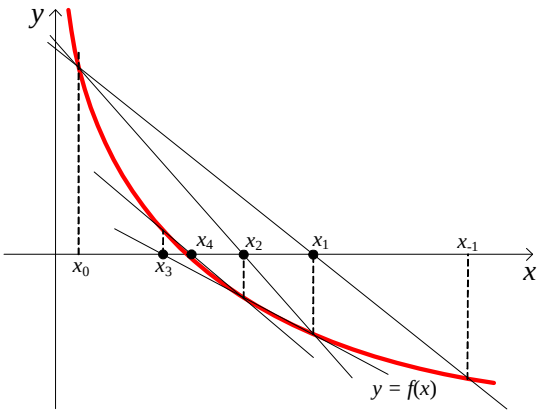
Masalah yang ada dalam metode Newton adalah terkadang sulit untuk mendapatkan turunan pertama $f'(x)$. Alternatifnya adalah turunan $f'(x_{n-1})$, kemiringan garis di $(x_{n-1}, f(x_{n-1}))$, dihampiri oleh kemiringan garis potong yang melalui $(x_{n-2}, f(x_{n-2}))$ dan $(x_{n-1}, f(x_{n-1}))$:

$$f'(x_{n-1}) \approx \frac{f(x_{n-2}) - f(x_{n-1})}{x_{n-2} - x_{n-1}}.$$

Jadi iterasi x_n yaitu

$$x_n = x_{n-1} - \frac{f(x_{n-1})(x_{n-2} - x_{n-1})}{f(x_{n-2}) - f(x_{n-1})}. \tag{3.3}$$

Penjelasan grafis mengenai metode ini adalah seperti dalam Gambar 3.5.



Gambar 3.5: Ilustrasi grafis untuk akar hampiran dalam metode Secant.

Dicatat:

1. Persamaan di atas memerlukan dua tebakan awal x tetapi tidak memperhatikan perubahan tanda dari $f(x)$.

2. Kemudian dapat ditunjukkan bahwa

   $$|x_s - x_n| \leq C\,|x_s - x_{n-1}|^{1.6},$$

   sehingga metode garis potong lebih cepat daripada metode iterasi titik tetap, tetapi lebih lambat daripada metode Newton-Raphson.

Proses untuk metode garis potong diberikan seperti dalam Algoritma 4.

Contoh 3.7 *Gunakan metode garis potong untuk menyelesaikan* $x^2 - 3 = 0$ *dengan tebakan awal* $x_{-1} = 1$ *dan* $x_0 = 2$.

Algorithm 4 Algoritma Garis Potong

Masukan:

fungsi kontinu: $f(x)$
dua tebakan awal: x_{-1}, x_0
maksimum iterasi: $N \in \mathbb{N}$
toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-5}$

Penghitungan Inti: Ketika $n \leq N$ dan $\epsilon_h \geq \epsilon$,

Hitung: $x_n = x_{n-1} - \dfrac{f(x_{n-1})(x_{n-2} - x_{n-1})}{f(x_{n-2}) - f(x_{n-1})}$.

Hitung: $\epsilon_h = \left|\dfrac{x_n - x_{n-1}}{x_n}\right| \times 100\%$

Hasil akhir: akar x_n sedemikian sehingga $f(x_n) \approx 0$.

Penyelesaian. Dituliskan rumus untuk metode *garis potong*:

$$\begin{aligned} x_n &= x_{n-1} - \frac{\left(x_{n-1}^2 - 3\right)(x_{n-2} - x_{n-1})}{\left(x_{n-2}^2 - 3\right) - \left(x_{n-1}^2 - 3\right)} \\ &= x_{n-1} - \frac{\left(x_{n-1}^2 - 3\right)(x_{n-2} - x_{n-1})}{x_{n-2}^2 - x_{n-1}^2}, n = 1, 2, ... \end{aligned}$$

Iterasi 1:

$$\begin{aligned} x_1 &= x_0 - \frac{\left(x_0^2 - 3\right)(x_{-1} - x_0)}{x_{-1}^2 - x_0^2} = 2 - \frac{\left(2^2 - 3\right)(1 - 2)}{1^2 - 2^2} = 1.66667, \\ \epsilon_1 &= \left|\frac{1.66667 - 2}{1.66667}\right| \cdot 100\% = 20\%. \end{aligned}$$

Iterasi 2:

$$\begin{aligned} x_2 &= x_1 - \frac{\left(x_1^2 - 3\right)(x_0 - x_1)}{x_0^2 - x_1^2} = 1.66667 - \frac{\left(1.66667^2 - 3\right)(2 - 1.66667)}{2^2 - 1.66667^2} \\ &= 1.72727, \\ \epsilon_2 &= \left|\frac{1.72727 - 1.66667}{1.72727}\right| \cdot 100\% = 3.50877\%. \end{aligned}$$

Iterasi 3:

$$\begin{aligned} x_3 &= x_2 - \frac{\left(x_2^2 - 3\right)(x_1 - x_2)}{x_1^2 - x_2^2} = 1.72727 - \frac{\left(1.72727^2 - 3\right)(1.66667 - 1.72727)}{1.66667^2 - 1.72727^2} \\ &= 1.73214, \\ \epsilon_3 &= \left|\frac{1.73214 - 1.72727}{1.73214}\right| \cdot 100\% = 0.28116\%. \end{aligned}$$

Iterasi 4:

$$\begin{aligned} x_4 &= x_3 - \frac{\left(x_3^2 - 3\right)(x_2 - x_3)}{x_2^2 - x_3^2} = 1.73214 - \frac{\left(1.73214^2 - 3\right)(1.72727 - 1.73214)}{1.72727^2 - 1.73214^2} \\ &= 1.732051, \\ \epsilon_4 &= \left|\frac{1.732051 - 1.73214}{1.732051}\right| \cdot 100\% = 0.00532\%. \end{aligned}$$

Jadi pada iterasi ke-4 diperoleh akar hampiran $x = 1.732051$. ▼

Dalam MatLab, Algoritma 4 untuk metode garis potong diimplementasikan dalam fungsi `Secant()` berikut ini.

```
function [x,galat] = Secant(f,x_1,x0,N,tol)
% Secant  Menyelesaikan persamaan f(x) = 0 menggunakan metode
%         garis potong (Secant).
%
% Input:  f = fungsi dari x, gunakan fungsi inline('ekspresi','x')
%         x_1 = tebakan awal pertama
%         x0 = tebakan awal kedua
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: x = akar hampiran yang memenuhi kriteria
%         galat = persen galat relatif
% ---PENGHITUNGAN INTI:
if nargin < 4, tol = 1e-3; end
if nargin < 3, N = 100; end
n = 1; galat = 1;
while ( n <= N & galat > tol )
    x = x0-f(x0)*(x_1-x0)/(f(x_1)-f(x0)); % persamaan (3.3)
    galat = abs((x - x0)/x)*100;
    x_1 = x0;
    x0 = x;
    n = n+1;
end
```

Bab 4

Metode Iterasi

Tujuan Pembelajaran:
- Mengetahui metode-metode penyelesaian sistem persamaan linear secara iterasi.
- Mengaplikasikan iterasi Jacobi, Gauss-Seidel, dan SOR.

Untuk menyelesaikan suatu sistem persamaan linier $A\mathbf{x} = \mathbf{b}$, dipunyai pilihan antara metode langsung atau metode iterasi. Contoh dari metode langsung yaitu metode invers, eliminasi Gauss, dan dekomposisi LU. Metode iterasi mempunyai kelebihan dalam hal kemurahan memori dan waktu *CPU*. Metode ini dimulai dari penentuan nilai awal vektor x_0 sebagai suatu penyelesaian awal untuk $\mathbf{x}$. Terdapat beberapa metode iterasi seperti iterasi Jacobi, iterasi Gauss-Seidel, dan iterasi SOR.

Suatu klas besar dari metode iterasi dapat didefinisikan sebagai berikut. Sistem $A\mathbf{x} = \mathbf{b}$ dituliskan menjadi $Q\mathbf{x} = (Q - A)\,\mathbf{x} + \mathbf{b}$ untuk suatu matriks Q yang berorde sama dengan A. Selanjutnya didefinisikan suatu skema iterasi ke-k:

$$Q\mathbf{x}^{(k)} = (Q - A)\,\mathbf{x}^{(k-1)} + \mathbf{b}, \qquad k = 1, 2, ..., \tag{4.1}$$

untuk $\mathbf{x}^{(0)}$ adalah suatu tebakan awal. Diasumsikan bahwa Q adalah *non-singular*, sehingga skema iterasi menghasilkan suatu barisan tunggal vektor-vektor $\left\{\mathbf{x}^{(k)}\right\}$.

4.1 Iterasi Jacobi

Pertama kali dicatat bahwa matriks A dapat dituliskan sebagai $A = L + D + U$, dengan L adalah matriks segitiga bawah, D adalah matriks diagonal, dan U adalah matriks segitiga atas. Iterasi Jacobi memilih $Q = D$. Karena itu

$$\begin{aligned} \mathbf{x}^{(k)} &= D^{-1}\,(D - A)\,\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b} \\ &= D^{-1}\left\{\mathbf{b} - (A - D)\,\mathbf{x}^{(k-1)}\right\}, \qquad k = 1, 2, ..., \end{aligned}$$

dengan asumsi bahwa masukan-masukan diagonal dari A tidak sama dengan nol (jika tidak maka dilakukan penukaran baris-baris dan kolom-kolom untuk mendapatkan suatu sistem yang ekivalen). Untuk langkah ke-k, komponen-komponen $x_i^{(k)}$ dinyatakan oleh

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1,\ j\neq i}^{n} a_{ij}x_j^{(k-1)}\right), \qquad i = 1, 2, ..., n \tag{4.2}$$

Contoh 4.1 *Diketahui sistem persamaan linear:*

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned}$$

Sistem persamaan tersebut diubah susunannya menjadi

$$\begin{aligned} x_1 &= \frac{1}{10}\left(6 + x_2 - 2x_3\right), \\ x_2 &= \frac{1}{11}\left(25 + x_1 + x_3 - 3x_4\right), \\ x_3 &= \frac{1}{10}\left(-11 - 2x_1 + x_2 + x_4\right), \\ x_4 &= \frac{1}{8}\left(15 - 3x_2 + x_3\right). \end{aligned}$$

Kita bisa menyatakan bahwa nilai x_1, x_2, x_3, dan x_4 yang berada di ruas kiri sebagai $x^{(baru)}$. Sementara itu, nilai x_1, x_2, x_3, dan x_4 yang berada di ruas kanan sebagai $x^{(lama)}$. Secara umum, sistem persamaan tersebut dapat dituliskan menjadi seperti

$$\begin{aligned} x_1^{(k)} &= \frac{1}{10}\left(x_2^{(k-1)} - 2x_3^{(k-1)} + 6\right), \\ x_2^{(k)} &= \frac{1}{11}\left(x_1^{(k-1)} + x_3^{(k-1)} - 3x_4^{(k-1)} + 25\right), \\ x_3^{(k)} &= \frac{1}{10}\left(-2x_1^{(k-1)} + x_2^{(k-1)} + x_4^{(k-1)} - 11\right), \\ x_4^{(k)} &= \frac{1}{8}\left(-3x_2^{(k-1)} + x_3^{(k-1)} + 15\right). \end{aligned}$$

Dimisalkan bahwa nilai-nilai awal $\mathbf{x}^{(0)} = \mathbf{0}$. Pada langkah $k = 1$, diperoleh nilai-nilai untuk $\mathbf{x}^{(1)}$:

$$x_1^{(1)} = \frac{6}{10}, x_2^{(1)} = \frac{25}{11}, x_3^{(1)} = -\frac{11}{10}, x_4^{(1)} = \frac{15}{8}.$$

Setelah nilai-nilai $\mathbf{x}^{(1)}$ diperoleh, penghitungan diulangi kembali dengan nilai $k = 2$ untuk memperoleh nilai-nilai $\mathbf{x}^{(2)}$:

$$x_1^{(2)} = 1.0473, x_2^{(2)} = 1.7159, x_3^{(2)} = -0.8052, x_4^{(2)} = 0.8852.$$

Proses diulang lagi untuk nilai-nilai k berikutnya. Berikut ini diberikan suatu hasil sampai langkah ke-4.

k	0	1	2	3	4
$x_1^{(k)}$	0	0.6000	1.0473	0.9326	1.0152
$x_2^{(k)}$	0	2.2727	1.7159	2.0530	1.9537
$x_3^{(k)}$	0	−1.1000	−0.8052	−1.0493	−0.9681
$x_4^{(k)}$	0	1.8852	0.8852	1.1309	0.9738

Untuk kriteria penghentian iterasi, kita bisa menggunakan suatu norma dari vektor:

$$l_\infty = \|\mathbf{x}\|_\infty = \underset{1\le i\le n}{\text{maks}} |x_i| .$$

Sebagai contoh, vektor $\mathbf{x} = (3, -2, 8, 5)$ memiliki norma

$$l_\infty = \|\mathbf{x}\|_\infty = \underset{1\le i\le n}{\text{maks}} \{|3| , |-2| , |8| , |5|\} = 8.$$

Sekarang kita ambil kriteria penghentian dalam metode iterasi:

$$l_\infty = \left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty = \underset{1\le i\le n}{\text{maks}} \left|x_i^{(k)} - x_i^{(k-1)}\right| .$$

Berdasarkan kriteria tersebut, metode iterasi Jacobi dapat dinyatakan seperti dalam Algoritma 5, sedangkan implementasi dalam MatLab diberikan oleh fungsi `jacobi()`.

Algorithm 5 Algoritma Iterasi Jacobi

Masukan:

matriks koefisien sistem: $A \in \mathbb{R}^{n\times n}$
vektor konstanta sistem: $b \in \mathbb{R}^n$
vektor penyelesaian awal: $\mathbf{x}^{(0)} \in \mathbb{R}^n$
maksimum iterasi: $N \in \mathbb{N}$
toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-6}$

Penghitungan Inti:

Dibentuk matriks: $P = D^{-1}(D - A)$ dan $Q = D^{-1}b$.
Ketika $k \le N$ dan $norm \ge \varepsilon$,
Hitung: $\mathbf{x}^n = Q - P\mathbf{x}^{(n-1)}$
$norm = \text{maks}_{1\le i\le n} |x_i|$

Hasil akhir: vektor $\mathbf{x}$ sedemikian sehingga $A\mathbf{x} \approx b$.

```
function X = jacobi(A,b,X0,N,tol)
% jacobi  Menyelesaikan SPL AX = b menggunakan iterasi Jacobi.
%
% Input:  A = matriks koefisien dari sistem
%         b = vektor konstan dari sistem
%         X0 = penyelesaian awal
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: X = penyelesaian sistem

if nargin < 5, tol = 1e-6; end
if nargin < 4, N = 1000; end
if nargin < 3, X0 = zeros(size(b)); end
n = size(A,1);
X = X0;
P = zeros(n,n);
for i = 1:n
    for j = 1:n
        if j ~= i
```

```
            P(i,j) = A(i,j)/A(i,i); % elemen dari matriks inv(D)*(D-A)
        end
    end
    Q(i) = b(i)/A(i,i); % elemen dari matriks inv(D)*b
end
while k <= N & norma > tol
    X = Q' - P*X; % persamaan (4.2)
    norma = max(abs(X-X0));
    X0 = X;
end
```

4.2 Iterasi Gauss-Seidel

Dari persamaan (4.2) dicatat bahwa komponen-komponen dari $x^{(k)}$ diketahui, tetapi tidak digunakan, ketika penghitungan komponen-komponen sisanya. Metode Gauss-Seidel merupakan suatu modifikasi dari metode Jacobi, yaitu semua komponen-komponen terakhir yang dihitung dipergunakan. Prosedur dalam metode Gauss-Seidel diperoleh dengan memilih $Q = D + L$. Karena itu, dari (4.1) didapatkan bentuk matriks

$$\begin{aligned}(D+L)\,\mathbf{x}^{(k)} &= (D+L-A)\,\mathbf{x}^{(k-1)} + \mathbf{b} \\ D\mathbf{x}^{(k)} &= -L\mathbf{x}^{(k)} - U\mathbf{x}^{(k-1)} + \mathbf{b} \\ \mathbf{x}^{(k)} &= D^{-1}\left(\mathbf{b} - L\mathbf{x}^{(k)} - U\mathbf{x}^{(k-1)}\right), \qquad k = 1, 2, ...\end{aligned}$$

Jadi, skemanya adalah

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right), \qquad i = 1, 2, ..., n \tag{4.3}$$

dengan asumsi bahwa untuk langkah k, komponen-komponen $x_j^{(k)}$, $1 \le j \le i-1$, sudah diketahui.

Contoh 4.2 *Diperhatikan kembali sistem persamaan linear pada contoh sebelumnya. Sistem persamaan diubah susunannya menjadi*

$$\begin{aligned}x_1^{(k)} &= -\frac{1}{10}\left(x_2^{(k-1)} - 2x_3^{(k-1)} + 6\right), \\ x_2^{(k)} &= \frac{1}{11}\left(x_1^{(k)} + x_3^{(k-1)} - 3x_4^{(k-1)} + 25\right), \\ x_3^{(k)} &= \frac{1}{10}\left(-2x_1^{(k)} + x_2^{(k)} + x_4^{(k-1)} - 11\right), \\ x_4^{(k)} &= \frac{1}{8}\left(-3x_2^{(k)} + x_3^{(k)} + 15\right).\end{aligned}$$

Pada langkah $k = 1$, diperoleh nilai-nilai untuk $\mathbf{x}^{(1)}$:

$$x_1^{(1)} = 0.6000, x_2^{(1)} = 2.3272, x_3^{(1)} = -0.9873, x_4^{(1)} = 0.8789.$$

Setelah nilai-nilai $\mathbf{x}^{(1)}$ diperoleh, penghitungan diulangi kembali dengan nilai $k = 2$

untuk memperoleh nilai-nilai $\mathbf{x}^{(2)}$*:*

$$x_1^{(2)} = 1.0300, x_2^{(2)} = 2.0369, x_3^{(2)} = -1.0145, x_4^{(2)} = 0.9843.$$

Proses diulang lagi untuk nilai-nilai k *berikutnya. Berikut ini diberikan suatu hasil sampai langkah ke-*4*.*

k	0	1	2	3	4
$x_1^{(k)}$	0	0.6000	1.0300	1.0066	1.0009
$x_2^{(k)}$	0	2.3272	2.0369	2.0036	2.0003
$x_3^{(k)}$	0	$-0,9873$	-1.0145	-1.0025	-1.0003
$x_4^{(k)}$	0	0.8789	0.9843	0.9984	0.9998

Untuk keperluan komputasi, iterasi (4.3) dinyatakan secara khusus untuk $i = 1$ dan $i = n$ berturut-turut:

$$x_1^{(k)} = \left(b_1 - \sum_{j=2}^{n} a_{1j} x_j^{(k-1)} \right), \; x_n^{(k)} = \frac{1}{a_{nn}} \left(b_n - \sum_{j=1}^{n-1} a_{nj} x_j^{(k)} \right). \tag{4.4}$$

Sekarang kita bisa menyatakan skema untuk iterasi Gauss-Seidel seperti dalam Algoritma 6 dan implementasi dalam MatLab diberikan oleh fungsi `gauseid()`.

Algorithm 6 Algoritma Iterasi Gauss-Seidel

Masukan:
matriks koefisien sistem: $A \in \mathbb{R}^{n \times n}$
vektor konstanta sistem: $b \in \mathbb{R}^n$
vektor penyelesaian awal: $\mathbf{x}^{(0)} \in \mathbb{R}^n$
maksimum iterasi: $N \in \mathbb{N}$
toleransi keakuratan: ε, misalnya $\varepsilon = 10^{-6}$

Penghitungan Inti: Ketika $1 \leq k \leq N$ dan $norm \geq \varepsilon$,
Hitung: $x_1^{(k)}$ menggunakan (4.4)
untuk $i = 2 : 1 : n - 1$
Hitung: $x_i^{(k)}$ menggunakan (4.3)
$x_n^{(k)}$ menggunakan (4.4)
Hitung: $norm = \text{maks}_{1 \leq i \leq n} |x_i|$

Hasil akhir: vektor $\mathbf{x}$ sedemikian sehingga $A\mathbf{x} \approx b$.

```
function X = gauseid(A,b,X0,N,tol)
% gauseid  Menyelesaikan SPL AX = b menggunakan iterasi Gauss-Seidel.
%
% Input:  A = matriks koefisien dari sistem
%         b = vektor kolom untuk nilai konstanta dari sistem
%         X0 = penyelesaian awal
%         N = maksimum iterasi
%         tol = toleransi keakuratan
%
% Output: X = penyelesaian sistem
```

```
if nargin < 5, tol = 1e-6; end
if nargin < 4, N = 1000; end
if nargin < 3, X0 = zeros(size(b)); end
n = size(A,1);
X = X0;
while k <= N & norma > tol
    X(1,:) = (b(1)-A(1,2:n)*X(2:n,:))/A(1,1); % persamaan (4.4)
    for i = 2:n-1
        tmp = b(i,:)-A(i,1:i-1)*X(1:i-1,:)-A(i,i+1:n)*X(i+1:n,:);
        X(i,:) = tmp/A(i,i); % persamaan (4.3)
    end
    X(n,:) = (b(n,:)-A(n,1:n-1)*X(1:n-1,:))/A(n,n); % persamaan (4.4)
    norma = max(abs(X-X0));
    X0 = X;
end
```

4.3 Iterasi SOR

Berikutnya diperhatikan suatu metode untuk mempercepat konvergensi dari metode iterasi. Dipilih

$$Q = \frac{1}{\omega}D + L$$

dengan ω adalah suatu faktor skala, maka iterasi (4.1) menjadi

$$\begin{aligned}
\left(\frac{1}{\omega}D + L\right)\mathbf{x}^{(k)} &= \left(\frac{1}{\omega}D + L - A\right)\mathbf{x}^{(k-1)} + \mathbf{b} \\
\left(\frac{1}{\omega}D + L\right)\mathbf{x}^{(k)} &= \left\{\left(\frac{1}{\omega} - 1\right)D - U\right\}\mathbf{x}^{(k-1)} + \mathbf{b} \\
\frac{1}{\omega}D\mathbf{x}^{(k)} &= -L\mathbf{x}^{(k)} + \left\{\left(\frac{1}{\omega} - 1\right)D - U\right\}\mathbf{x}^{(k-1)} + b \\
\mathbf{x}^{(k)} &= -\omega D^{-1}L\mathbf{x}^{(k)} + \left(1 - \omega - \omega D^{-1}U\right)\mathbf{x}^{(k-1)} + \omega D^{-1}b \\
\mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} - \omega D^{-1}\left(L\mathbf{x}^{(k)} + D\mathbf{x}^{(k-1)} + U\mathbf{x}^{(k-1)} - b\right)
\end{aligned}$$

untuk $k = 1, 2, ...$ Secara jelas, ini diproses dengan cara:

$$\begin{aligned}
x_i^{(k)} &= x_i^{(k-1)} - \frac{\omega}{a_{ii}}\left(\sum_{i<j} a_{ji}x_i^{(k)} + a_{ii}x_i^{(k-1)} + \sum_{i>j} a_{ji}x_i^{(k-1)} - b_i\right) \\
&= (1-\omega)\,x_i^{(k-1)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right) \qquad (4.5)
\end{aligned}$$

untuk $i = 1, 2, ..., n$, dan diasumsikan bahwa untuk langkah ke-k, komponen-komponen $x_j^{(k)}$, $1 \le j \le i-1$, sudah diketahui.

Untuk $\omega = 1$, iterasi (4.5) memberikan metode Gauss-Seidel. Untuk $0 < \omega < 1$, prosedurnya dinamakan metode *under-relaxation* dan dapat digunakan untuk memperoleh konvergensi dari beberapa sistem yang tidak konvergen oleh metode Gauss-Seidel.

Untuk $\omega > 1$, prosedurnya dinamakan metode *overrelaxation*, yang digunakan untuk mempercepat konvergensi bagi sistem yang konvergen oleh teknik Gauss-Seidel.

Metode-metode tersebut disingkat SOR untuk *Successive Overrelaxation* dan digunakan untuk penyelesaian sistem linier yang muncul dalam penyelesaian numeris dari persamaan diferensial parsial tertentu.

Contoh 4.3 *Diketahui sistem persamaan linear:*

$$\begin{aligned} 4x_1 + 3x_2 &= 24, \\ 3x_1 + 4x_2 - x_3 &= 30, \\ -x_2 + 4x_3 &= -24. \end{aligned}$$

Persamaan untuk metode relaksasi dengan $\omega = 1.25$*:*

$$\begin{aligned} x_1^{(k)} &= (1-1.25)\, x_1^{(k-1)} + \frac{1.25}{4}\left(24 - 3x_2^{(k-1)}\right) \\ &= -0.25x_1^{(k-1)} - 0.9375x_2^{(k-1)} + 7.5, \\ x_2^{(k)} &= (1-1.25)\, x_2^{(k-1)} + \frac{1.25}{4}\left(30 - 3x_1^{(k)} + x_3^{(k-1)}\right) \\ &= -0.9375x_1^{(k)} - 0.25x_2^{(k-1)} + 0.3125x_3^{(k-1)} + 9.375, \\ x_3^{(k)} &= (1-1.25)\, x_3^{(k-1)} + \frac{1.25}{4}\left(-24 + x_2^{(k)}\right) \\ &= 0.3125x_2^{(k)} - 0.25x_3^{(k-1)} - 7.5. \end{aligned}$$

*Tabel berikut ini menampilkan hasil penghitungan sampai langkah ke-*4 *menggunakan penyelesaian awal* $\mathbf{x}^{(0)} = \mathbf{0}$*.*

k	0	1	2	3	4
$x_1^{(k)}$	1	7.5000	3.4277	3.3987	3.0442
$x_2^{(k)}$	1	2.3438	3.4607	3.8465	3.9606
$x_3^{(k)}$	1	−6.7676	−4.7266	−5.1163	−4.9832

Skema iterasi (4.5) diimplementasikan dengan fungsi MatLab `SOR()` berikut ini.

```
function X = SOR(A,b,X0,N,tol,w)
% SOR  Menyelesaikan SPL Ax = b menggunakan iterasi SOR
%
% Input:  A = matriks koefisien dari sistem
%         b = vektor kolom untuk nilai konstanta dari sistem
%         X0 = penyelesaian awal
%         w = faktor skala
%         tol  = toleransi keakuratan
%
% Output: X = penyelesaian sistem
if nargin < 6, w = 1.25; end
if nargin < 5, tol = 1e-6; end
if nargin < 4, N = 1000; end
if nargin < 3, X0 = zeros(size(b)); end
n = size(A,1);
X = X0;
for k = 1:N
    X(1) = (1-w)*X(1)+w*(b(1)-A(1,2:n)*X(2:n,:))/A(1,1);
```

```
    for i = 2:n-1
        tmp = b(i,:)-A(i,1:i-1)*X(1:i-1,:)-A(i,i+1:n)*X(i+1:n,:);
        X(i) = (1-w)*X(i)+w*tmp/A(i,i);
    end
    X(n) = (1-w)*X(n)+w*(b(n,:)-A(n,1:n-1)*X(1:n-1,:))/A(n,n);
    galat = max(abs(X-X0));
    if galat < tol, break; end
    X0 = X;
end
```

Bab 5

Interpolasi Polinomial

Tujuan Pembelajaran:
- Mengetahui metode penentuan suatu polinomial yang melewati semua titik data yang diberikan.
- Mengaplikasikan interpolasi polinomial untuk menaksir nilai antara titik-titik data.

Interpolasi menghubungkan titik-titik data diskret dalam suatu cara yang masuk akal sehingga dapat diperoleh taksiran layak dari titik-titik data di antara titik-titik yang diketahui. Dicatat bahwa kurva interpolasi melalui semua titik data.

5.1 Interpolasi Linear

Interpolasi linear merupakan interpolasi paling sederhana dengan mengasumsikan bahwa hubungan titik-titik antara dua titik data adalah linear. Karena itu digunakan pendekatan fungsi linear antara dua titik data, misalnya (x_0, y_0) dan (x_1, y_1). Persamaan yang menghubungkan kedua titik tersebut yaitu

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0).$$

Contoh 5.1 *Hitung taksiran y untuk $x = 2$ dengan menggunakan interpolasi linear untuk data:* $(1, 0)$ *dan* $(4, 1.386294)$.

Penyelesaian. Taksiran y untuk $x = 2$ yaitu

$$\begin{aligned} y &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(2 - x_0) = 0 + \frac{1.386294 - 0}{4 - 1}(2 - 1) \\ &= 0.462098. \end{aligned}$$

5.2 Interpolasi Kuadratik

Interpolasi kuadratik menentukan titik-titik antara tiga titik data dengan menggunakan pendekatan fungsi kuadrat. Dibentuk persamaan kuadrat yang melalui tiga titik data, misalnya (x_0, y_0), (x_1, y_1), dan (x_2, y_2), yaitu

$$y = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1).$$

Kita akan menentukan b_0, b_1, b_2 sedemikian sehingga persamaan di atas melalui ketiga titik data yang diberikan.

$$\begin{aligned}
x &= x_0, y = y_0 \implies a_0 = y_0,\\
x &= x_1, y = y_1 \implies y_0 + b_1(x_1 - x_0) = y_1 \implies a_1 = \frac{y_1 - y_0}{x_1 - x_0},\\
x &= x_2, y = y_2 \implies y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = y_2\\
&\implies a_2 = \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_1 - x_0)(x_2 - x_1)}\\
&\implies a_2 = \frac{y_2 - y_1}{(x_2 - x_0)(x_2 - x_1)} + \frac{y_1 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_1 - x_0)(x_2 - x_1)}\\
&\implies a_2 = \frac{y_2 - y_1}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_2 - x_0)(x_1 - x_0)}\\
&\implies a_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}.
\end{aligned}$$

Contoh 5.2 *Hitung taksiran y untuk $x = 2$ dengan menggunakan interpolasi kuadratik untuk data:* $(1, 0)$, $(4, 1.386294)$, $(6, 1.791759)$.

Penyelesaian. Koefisien-koefisien dari persamaan interpolasi kuadratik:

$$\begin{aligned}
a_0 &= 1,\\
a_1 &= \frac{1.386294 - 0}{4 - 1} = 0.4620981,\\
a_2 &= \frac{\frac{1.791759 - 1.386294}{6 - 4} - \frac{1.386294 - 0}{4 - 1}}{6 - 1} = -0.0518731,
\end{aligned}$$

sehingga interpolasi kuadratik untuk data yang diberikan yaitu

$$y = 1 + 0.4620981(x - 1) - 0.0518731_2(x - 1)(x - 4).$$

Karena itu, nilai y untuk $x = 2$ yaitu

$$y = 1 + 0.4620981(2 - 1) - 0.0518731_2(2 - 1)(2 - 4) = 0.5658444.$$

5.3 Interpolasi Newton

Secara umum, $n+1$ titik data, misalnya (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) dapat dicocokkan dengan suatu polinomial berderajat n yang mempunyai bentuk

$$\begin{aligned}
y &= f_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots\\
&\quad + a_n(x - x_0)(x - x_1) \cdot \cdots \cdot (x - x_{n-1}).
\end{aligned} \tag{5.1}$$

Persamaan-persamaan yang digunakan untuk menghitung koefisien-koefisien yaitu

$$\begin{aligned}
a_0 &= y_0, \\
a_1 &= \frac{y_1 - y_0}{x_1 - x_0} = y\left[x_1, x_0\right], \\
a_2 &= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} = \frac{y\left[x_2, x_1\right] - y\left[x_1, x_0\right]}{x_2 - x_0} = y\left[x_2, x_1, x_0\right], \\
&\vdots \\
a_3 &= \frac{y\left[x_n, x_{n-1}, ..., x_2, x_1\right] - y\left[x_{n-1}, x_{n-2}, ..., x_1, x_0\right]}{x_n - x_0} = y\left[x_n, x_{n-1}, ..., x_1, x_0\right].
\end{aligned}$$

Nilai fungsi berkurung siku dinamakan beda terbagi hingga dan didefinisikan sebagai

$$\begin{aligned}
y\left[x_i, x_{i-1}\right] &= \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \text{ (beda terbagi hingga pertama)}, \\
y\left[x_{i-2}, x_{i-1}, x_i\right] &= \frac{y\left[x_{i-2}, x_{i-1}\right] - y\left[x_{i-1}, x_i\right]}{x_{i-2} - x_i} \text{ (beda terbagi hingga kedua)}, \\
y\left[x_n, x_{n-1}, ..., x_1, x_0\right] &= \frac{y\left[x_n, ..., x_1\right] - y\left[x_{n-1}, ..., x_0\right]}{x_n - x_0} \text{(beda terbagi hingga ke-}(n)\text{)}
\end{aligned} \tag{5.2}$$

Persamaan-persamaan di atas adalah rekursif, yaitu beda orde lebih tinggi dihitung dengan mengambil beda dari orde lebih rendah.

Sebagai contoh, penghitungan koefisien-koefisien dalam polinomial berderajat tiga dapat diperoleh secara berturut-turut mulai dari baris kedua dalam Tabel 5.1.

Tabel 5.1: Tabel beda terbagi hingga untuk orde 3.

n	x_n	y_n	pertama	kedua	ketiga
0	x_0	$b_0 = y_0$	$b_1 = \frac{y_1 - y_0}{x_1 - x_0}$	$b_2 = \frac{b_{11} - b_1}{x_2 - x_0}$	$b_3 = \frac{b_{21} - b_2}{x_3 - x_0}$
1	x_1	y_1	$b_{11} = \frac{y_2 - y_1}{x_2 - x_1}$	$b_{21} = \frac{b_{12} - b_{11}}{x_3 - x_1}$	–
2	x_2	y_2	$b_{12} = \frac{y_3 - y_2}{x_3 - x_2}$	–	–
3	x_3	y_3	–	–	–

Contoh 5.3 *Hitung taksiran y untuk $x = 2$ dengan menggunakan interpolasi polinomial Newton untuk empat titik data:*

$$(1, 0), (4, 1.386294), (5, 1.609438), \text{ dan } (6, 1.791759).$$

Penyelesaian. Disusun tabel beda terbagi hingga:

n	x_n	y_n	pertama	kedua	ketiga
0	1	$a_0 = 0$	$a_1 = 0.4621$	$a_2 = -0.0598$	$a_3 = 0.0079$
1	4	1.386294	$a_{11} = 0.2231$	$a_{21} = -0.0203$	–
2	5	1.609438	$a_{12} = 0.1824$	–	–
3	6	1.791759	–	–	–

Diperoleh polinomial Newton orde ketiga:

$$y = 0.4621\,(x-1) - 0.0598\,(x-1)\,(x-4) + 0.0079\,(x-1)\,(x-4)\,(x-5)\,. \qquad (5.3)$$

Karena itu, nilai y untuk $x = 2$ yaitu

$$\begin{aligned} y &= 0 + 0.4621\,(2-1) - 0.0598\,(2-1)\,(2-4) + 0.0079\,(2-1)\,(2-4)\,(2-5) \\ &= 0.6289. \end{aligned}$$

Dicatat bahwa efisiensi komputasi yang lebih baik untuk menuliskan polinomial Newton (5.1) adalah dalam bentuk perkalian bersarang seperti

$$y = ((\cdots(a_n\,(x - x_{n-1}) + a_{n-1})\,(x - x_{n-2}) + \cdots) + a_1)\,(x - x_0) + a_0, \qquad (5.4)$$

untuk mendapatkan koefisien-koefisien dari suku-suku x^n, x^{n-1}, ..., x^2, x, dan x^0. Dalam MatLab ini dapat dilakukan dengan

```
X; % X = [x_0 x_1 x_2 ... x_n]
a; % a = [a_0 a_1 a_2 ... a_n]
K = koef(n+1); % koefisien dari x^n
for i = n:-1:1 % koefisien dari x^n, ..., x^0
    K = [K a(i)] - [0 K*X(i)]; % a(i)*(x - x_(i - 1))+a_(i - 1)
end
```

Fungsi MatLab `poliNewton()` menyusun Tabel Beda Terbagi Hingga seperti Tabel 5.1, mengkonstruksi polinomial Newton dan selanjutnya menghitung nilai y untuk suatu nilai x yang diberikan. Fungsi ini juga menampilkan plot titik-titik data dan kurva polinomial Newton. Di sini dicatat bahwa koefisien-koefisien dari polinomial dinyatakan sebagai vektor yang disusun dalam derajat yang menurun.

```
function [T,K,yi]=poliNewton(X,Y,xi)
% poliNewton  Menyusun Tabel Beda Terbagi Hingga untuk interpolasi polinomial Newton
%             dan menghitung nilai fungsi interpolasi untuk suatu nilai x = xi.
%
% Input:  X = data x
%         Y = data y yang berkorespondensi dengan x
%         xi = suatu nilai di antara titik-titik data x
%
% Output: T = Tabel Beda Terbagi Hingga mulai pertama
%         K = vektor yang memuat koefisien dari x^n, x^(n-1), ..., x, x^0
%         yi = f(xi)
%
% ---PENGHITUNGAN INTI:
N = length(X)-1; % banyaknya pasangan data
% konstruksi tabel beda terbagi
hasil = zeros(N+1,N+3);
hasil(:,1) = [0:N]'; % iterasi
hasil(:,2) = X'; % nilai x
hasil(:,3) = Y'; % nilai y
```

```
for j=4:N+3
    for i=1:N+3-(j-1) % beda terbagi orde 1,2, ..., n-1
        hasil(i,j) = (hasil(i+1,j-1)-hasil(i,j-1))/(hasil(j+i-3,2)-hasil(i,2));
    end
end
Tabel = hasil;

% ---OUTPUT:
% tabel beda terbagi hingga:
T = Tabel(:,4:end);
a = hasil(1,3:end); % a = [a_0 a_1 a_2 ... a_N]

% koef x^n, ..., x^0:
K = a(N+1); % koefisien dari x^N
for i = N:-1:1
   K = [K a(i)] - [0 K*X(i)]; %a(i)*(x - x_(i - 1))+a_(i - 1)
end

% nilai y untuk suatu nilai xi:
yi = sum(K'.*xi.^([N:-1:0]'));

% plot titik-titik data dan kurva polinomial
xsim = X(1):0.1:X(end);
for k=1:length(xsim)
    ysim(k) = sum(K'.*xsim(k).^([N:-1:0]'));
end
plot(X,Y,'bo',xsim,ysim,'r-')
```

Sebagai contoh, dengan menjalankan fungsi MatLab `poliNewton` untuk data dalam 5.3 akan diperoleh fungsi polinomial Newton orde tiga yaitu

$$y = 0.0079x^3 - 0.1386x^2 + 0.9894x - 0.8587$$

yang adalah sama dengan hasil (5.3). Kurva dari fungsi tersebut diberikan dalam Gambar 5.1.

5.4 Interpolasi Lagrange

Interpolasi polinomial Lagrange hanyalah perumusan ulang dari polinomial Newton yang menghindari penghitungan beda terbagi hingga. Berikut ini adalah penurunan bentuk Lagrange secara langsung dari interpolasi polinomial Newton.

Untuk kasus orde pertama dipunyai

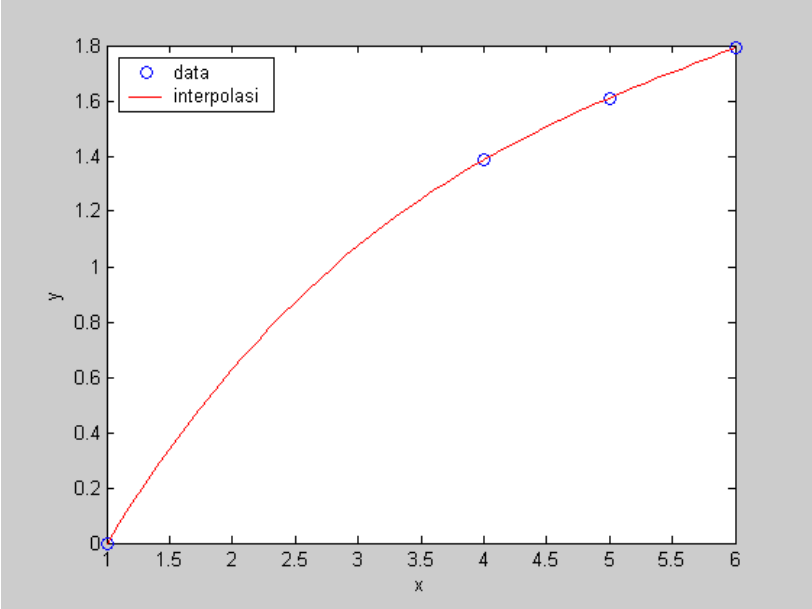
$$f_1(x) = y_0 + (x - x_0)\, y[x_1, x_0]. \tag{5.5}$$

Beda terbagi hingga pertama

$$y[x_1, x_0] = \frac{y_1 - y_0}{x_1 - x_0}$$

dapat dirumuskan ulang sebagai

$$y[x_1, x_0] = \frac{y_1}{x_1 - x_0} + \frac{y_0}{x_0 - x_1}. \tag{5.6}$$



Gambar 5.1: Kurva polinomial Newton untuk Contoh 5.3.

Karena itu, dengan mensubstitusikan (5.6) ke (5.5) akan dihasilkan

$$f_1(x) = y_0 + \frac{x - x_0}{x_1 - x_0} y_1 + \frac{x - x_0}{x_0 - x_1} y_0.$$

Selanjutnya dengan pengelompokkan suku-suku yang serupa dan penyederhanaan akan diperoleh bentuk Lagrange:

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1.$$

Dengan cara yang serupa, yaitu beda-beda terbagi hingga dirumuskan ulang, akan dihasilkan polinomial Lagrange derajat n untuk $n + 1$ titik data, misalnya (x_0, y_0), (x_1, y_1), ..., (x_n, y_n):

$$y = f_n(x) = \sum_{i=0}^{n} L_i(x)\, y_i \tag{5.7}$$

dengan

$$L_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}. \tag{5.8}$$

Dalam hal ini, $L_i(x)$ dinamakan sebagai polinomial koefisien Lagrange dan mempunyai sifat:

$$L_i(x_k) = \begin{cases} 0 & , i \neq k \\ 1 & , i = k \end{cases}.$$

Sebagai contoh, dari rumus di atas, versi orde kedua untuk polinomial Lagrange mem-

punyai bentuk yaitu

$$\begin{aligned} f_2(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 \\ &\quad + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2. \end{aligned}$$

Contoh 5.4 *Diperhatikan kembali data dalam Contoh 5.3. Polinomial-polinomial koefisien Lagrange:*

$$\begin{aligned} L_0(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-4)(x-5)(x-6)}{(1-4)(1-5)(1-6)} \\ &= -\frac{1}{60}(x-4)(x-5)(x-6) = -\frac{1}{60}(), \\ L_1(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-1)(x-5)(x-6)}{(4-1)(4-5)(4-6)} \\ &= \frac{1}{6}(x-1)(x-5)(x-6), \\ L_2(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-1)(x-4)(x-6)}{(5-1)(5-4)(5-6)} \\ &= -\frac{1}{4}(x-1)(x-4)(x-6), \\ L_3(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-1)(x-4)(x-5)}{(6-1)(6-4)(6-5)} \\ &= \frac{1}{10}(x-1)(x-4)(x-5). \end{aligned}$$

Jadi, polinomial Lagrange yang sesuai dengan 4 *titik data yang diberikan:*

$$\begin{aligned} y &= -\frac{1}{60}(x-4)(x-5)(x-6)\cdot 0 + \frac{1}{6}(x-1)(x-5)(x-6)\cdot 1.386294 \\ &\quad -\frac{1}{4}(x-1)(x-4)(x-6)\cdot 1.609438 + \frac{1}{10}(x-1)(x-4)(x-5)\cdot 1.791759 \\ &= 0.231049(x-1)(x-5)(x-6) - 0.4023595(x-1)(x-4)(x-6) \\ &\quad +0.1791759(x-1)(x-4)(x-5). \end{aligned}$$

Koefisien-koefisien dari suku-suku x^n, x^{n-1}, ..., x, x^0 dalam polinomial Lagrange orde n dapat dibentuk melalui perkalian dua polinomial:

$$(b_n x^n + \cdots + b_1 x + b_0)(c_n x^n + \cdots + c_1 x + c_0) = d_{2n}x^{2n} + \cdots d_1 x + d_0$$

dengan

$$d_k = \sum_{m=\text{maks}(0,k-n)}^{\min(k,n)} b_{k-m}c_m, \quad \text{untuk } k = 2n, 2n-1, ..., 1, 0.$$

Operasi ini dapat dilakukan dengan menggunakan perintah `conv()` yang sudah diberikan oleh MatLab seperti diilustrasikan berikut ini.

```
>> b = [1 -1]; c = [1 1 1];
>> d = conv(a,b)
     d = 1 0 0 -1 % artinya bahwa (x-1)*(x^2+x+1)=x^3+0*x^2+0*x-1
```

Fungsi MatLab `poliLagrange()` mencari koefisien-koefisien dari polinomial Lagrange (5.7) bersama-sama dengan setiap polinomial koefisien Lagrange (5.8). Fungsi ini juga menampilkan plot titik-titik data dan kurva polinomial Lagrange. Di sini dicatat bahwa koefisien-koefisien dari polinomial dinyatakan sebagai vektor yang disusun dalam derajat yang menurun.

```
function [K,L,yi]=poliLagrange(X,Y,xi)
% poliLagrange  Mencari koefisien dari polinomial Lagrange dan menghitung
%               nilai fungsi interpolasi untuk suatu nilai x = xi.
%
% Input:  X = data x
%         Y = data y yang berkorespondensi dengan x
%         xi = suatu nilai di antara titik-titik data x
%
% Output: K = vektor yang memuat koefisien dari x^n, x^(n-1), ..., x, x^0
%         L = vektor yang memuat polinomial koefisien Lagrange
%         yi = f(xi)
%
% ---PENGHITUNGAN INTI:
N = length(X)-1; % banyaknya pasangan data
L = zeros(N+1,N+1);

% membentuk polinomial koefisien Lagrange
for i=1:N+1
   P = 1;
   for j=1:N+1
      if i~=j
         P = conv(P,[1 -X(j)])/(X(i)-X(j));
      end
   end
   L(i,:) = P;
end

% menentukan koefisien dari polinomial Lagrange
K = Y*L;

% menghitung nilai y = f(xi):
yi = sum(K'.*xi.^([N:-1:0]'));

% plot titik-titik data dan kurva polinomial
xsim = X(1):0.1:X(end);
for k=1:length(xsim)
    ysim(k) = sum(K'.*xsim(k).^([N:-1:0]'));
end
plot(X,Y,'bo',xsim,ysim,'r-')
```

Bab 6

Interpolasi Spline

Tujuan Pembelajaran:
- Mengetahui interpolasi spline untuk mendapatkan kurva mulus di antara dua titik data.
- Mengaplikasikan interpolasi spline untuk sekumpulan data.

Pada bab sebelumnya telah dibahas mengenai interpolasi titik-titik data (x_0, y_0) sampai (x_n, y_n) menggunakan suatu polinomial berderajat n. Namun terdapat kasus dimana fungsi-fungsi ini memberikan hasil yang salah. Pendekatan alternatifnya adalah menerapkan polinomial-polinomial berderajat lebih rendah pada sebagian titik data. Polinomial penghubung tersebut dinamakan fungsi-fungsi spline.

Definisi 6.1 *Suatu fungsi $f(x)$ dinamakan suatu spline berderajat k jika*

1. *Domain dari S adalah suatu interval $[a, b]$.*
2. *$S, S', ..., S^{(k-1)}$ kontinu pada $[a, b]$.*
3. *Terdapat titik-titik x_i sedemikian sehingga $a = x_0 < x_1 < ... < x_n = b$ dan juga S adalah suatu polinomial berderajat k pada setiap $[x_i, x_{i+1}]$.*

Dengan kata lain, spline adalah potongan-potongan fungsi polinomial dengan turunan-turunan memenuhi kendala-kendala kekontinuan tertentu. Ketika $k = 1$, spline dinamakan spline linear. Ketika $k = 2$, spline dinamakan spline kuadratik. Ketika $k = 3$, spline dinamakan spline kubik.

6.1 Spline Linear

Kita mencoba mencari suatu fungsi spline linear $S(x)$ sedemikian sehingga $S(x_i) = y_i$ untuk $0 \leq i \leq n$. Diambil

$$S(x) = \begin{cases} S_0(x) & , x_0 \leq x \leq x_1 \\ S_1(x) & , x_1 \leq x \leq x_2 \\ \vdots & \vdots \\ S_{n-1}(x) & , x_{n-1} \leq x \leq x_n \end{cases}$$

dimana setiap $S_i(x)$ adalah linear.

Diperhatikan fungsi linear $S_i(x)$. Garis ini melalui titik (x_i, y_i) dan (x_{i+1}, y_{i+1}), sehingga kemiringan dari $S_i(x)$ yaitu

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}.$$

Kita dapat juga mengatakan bahwa garis tersebut melalui titik (x_i, y_i) dan $(x, S(x))$ untuk sembarang $x \in [x_i, x_{i+1}]$, sehingga

$$m_i = \frac{S_i(x) - y_i}{x - x_i},$$

yang memberikan

$$\begin{aligned} S_i(x) &= y_i + m_i(x - x_i) \\ &= y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \end{aligned} \tag{6.1}$$

Contoh 6.2 *Konstruksi spline kuadratik untuk data berikut:*

x	0.0	0.1	0.4	0.5	0.75	1.0
y	1.3	4.5	2.0	2.1	5.0	3.0

Penyelesaian.

$$\begin{aligned} [0.0, 0.1] &: S_0(x) = 1.3 + \frac{4.5 - 1.3}{0.1 - 0}(x - 0) = 1.3 + 32x, \\ [0.1, 0.4] &: S_1(x) = 4.5 + \frac{2.0 - 4.5}{0.4 - 0.1}(x - 0.1) = \frac{16}{3} - \frac{25}{3}x, \\ [0.4, 0.5] &: S_2(x) = 2.0 + \frac{2.1 - 2.0}{0.5 - 0.4}(x - 0.4) = 1.6 + x, \\ [0.5, 0.75] &: S_3(x) = 2.1 + \frac{5.0 - 2.1}{0.75 - 0.5}(x - 0.5) = -3.7 + 11.6x, \\ [0.75, 1.0] &: S_4(x) = 5.0 + \frac{3.0 - 5.0}{1.0 - 0.75}(x - 0.75) = 11 - 8x. \end{aligned}$$

Jadi spline adalah potongan linear, yaitu linear di antara setiap titik data.

Persamaan (6.1) dapat dituliskan kembali sebagai

$$S_i(x) = a_i x + b_i, \quad i = 0, 1, ..., n - 1 \tag{6.2}$$

dengan

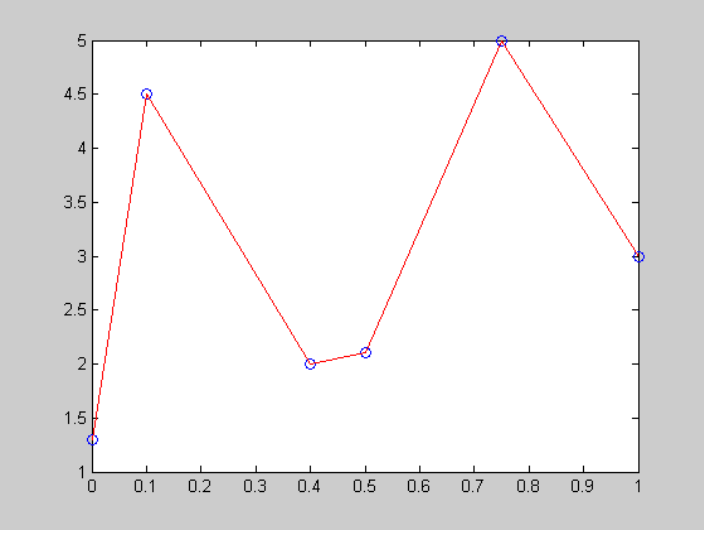
$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad \text{dan} \quad b_i = y_i - a_i x_i.$$

Fungsi MatLab `spline1()` mengkonstruksi spline linear dengan menghitung koefisien-koefisien spline linear (6.2) untuk koordinat-koordinat (x, y) dari titik-titik data. Fungsi ini juga menghitung taksiran nilai y untuk suatu nilai x di antara titik-titik data. Selain itu, kurva spline linear akan ditampilkan oleh fungsi tersebut.

```
function [S,yi] = spline1(X,Y,xi)
% spline2  Mengkonstruksi fungsi spline linear untuk titk-titik data (x,y).
%
% Input:  X = [x0,x1,...,xn]
%         Y = [y0,y1,...,yn]
%         xi = suatu nilai di antara x0 dan xn
%
% Output: S = [ai,bi], koefisien-koefisien spline linear y = ai*x + bi
%         yi = taksiran nilai pada x = xi
%
% ---PENGHITUNGAN INTI:
if nargin < 3, xi = X(1); end
n = length(X);
S = [];

% koefisien-koefisien spline linear
for i=1:n-1
    a = (Y(i+1)-Y(i))/(X(i+1)-X(i));
    b = Y(i)-a*X(i);
    if xi>X(i) & xi<X(i+1)
        yi = a*xi+bi; % nilai y untuk suatu x = xi
    end
    S = [S; a b]; % derajat turun
end

% plot kurva-kurva spline
plot(X,Y,'bo')
for i=1:n-1
    hold on
    y(i) = S(i,1)*X(i)+S(i,2);
    y(i+1) = S(i,1)*X(i+1)+S(i,2);
    plot(X(i:i+1),y(i:i+1),'r-')
end
hold off
```



Gambar 6.1: Spline kuadratik untuk Contoh 6.2.

Dari contoh di atas telah ditunjukkan bahwa kekurangan utama spline-spline linear adalah ketidakmulusannya. Artinya, pada titik-titik data di mana dua spline bertemu, kemiringannya berubah secara mendadak. Secara formal ini berarti bahwa turunan pertama dari fungsi tidak kontinu pada titik-titik tersebut. Kelemahan ini diatasi oleh penggunaan polinomial spline orde yang lebih tinggi.

6.2 Spline Kuadratik

Tidak seperti spline linear, spline kuadratik tidak didefinisikan sepenuhnya oleh nilai-nilai di x_i. Berikut ini kita perhatikan alasannya. Spline kuadratik didefinisikan oleh

$$S_i(x) = a_i x^2 + b_i x + c_i.$$

Jadi terdapat $3n$ parameter untuk mendefinisikan $S(x)$.

Diperhatikan titik-titik data:

x_0	x_1	x_2	$\cdots$	x_n
y_0	y_1	y_2	$\cdots$	y_n

Syarat-syarat untuk menentukan $3n$ parameter dijelaskan seperti berikut ini.

1. Setiap subinterval $[x_i, x_{i+1}]$, untuk $i = 0, 1, 2, ..., n-1$, memberikan dua persamaan berkaitan dengan $S_i(x)$, yaitu

   $$S_i(x_i) = y_i \quad \text{dan} \quad S_i(x_{i+1}) = y_{i+1}.$$

   Jadi dari sini dipunyai $2n$ persamaan.

2. Syarat pada kontinuitas dari $S'(x)$ memberikan suatu persamaan tunggal untuk setiap titik dalam x_i, $i = 1, 2, ..., n-1$, yaitu

   $$S'_{i-1}(x_i) = S'_i(x_i).$$

   Jadi dari sini dipunyai $n-1$ persamaan. Sekarang totalnya terdapat $3n-1$ persamaan, tetapi karena terdapat $3n$ parameter yang tidak diketahui maka sistem mempunyai kekurangan ketentuan.

3. Pilihan-pilihan yang mungkin untuk melengkapi kekurangan ketentuan yaitu

   $$S'(x_0) = 0 \quad \text{atau} \quad S''(x_0) = 0.$$

Sekarang dimisalkan $z_i = S'_i(x_i)$. Karena $S_i(x_i) = y_i$, $S'_i(x_i) = z_i$, dan $S'_i(x_{i+1}) = z_{i+1}$, maka kita dapat mendefinisikan

$$S_i(x) = \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x - x_i)^2 + z_i(x - x_i) + y_i. \tag{6.3}$$

Selanjutnya, dengan pengambilan $x = x_{i+1}$ diperoleh

$$\begin{aligned} y_{i+1} &= S_i(x_{i+1}) = \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x_{i+1} - x_i)^2 + z_i(x_{i+1} - x_i) + y_i \\ y_{i+1} - y_i &= \frac{z_{i+1} - z_i}{2}(x_{i+1} - x_i) + z_i(x_{i+1} - x_i) \\ y_{i+1} - y_i &= \frac{z_{i+1} + z_i}{2}(x_{i+1} - x_i). \end{aligned}$$

Jadi, kita dapat menentukan z_{i+1} dari z_i:

$$z_{i+1} = 2\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - z_i. \tag{6.4}$$

Contoh 6.3 *Konstruksi spline kuadratik untuk data berikut:*

x	0.0	0.1	0.4	0.5
y	1.3	4.5	2.0	2.1

dengan penetapan $z_0 = 0$.

Penyelesaian. Pertama kali dihitung nilai-nilai z_i:

$$\begin{aligned} z_1 &= 2\frac{y_1 - y_0}{x_1 - x_0} - z_0 = 2\frac{4.5 - 1.3}{0.1 - 0.0} - 0 = 64, \\ z_2 &= 2\frac{y_2 - y_1}{x_2 - x_1} - z_1 = 2\frac{2.0 - 4.5}{0.4 - 0.1} - 64 = -\frac{242}{3}, \\ z_3 &= 2\frac{y_3 - y_2}{x_3 - x_2} - z_2 = 2\frac{2.1 - 2.0}{0.5 - 0.4} + \frac{242}{3} = \frac{248}{3}, \end{aligned}$$

Jadi, fungsi spline kuadratik $S(x)$:

$$\begin{aligned} S_0(x) &= \frac{z_1 - z_0}{2(x_1 - x_0)}(x - x_0)^2 + z_0(x - x_0) + y_0 \\ &= 320x^2 + 1.3 = 320x^2 + 1.3, \quad \text{untuk } 0.0 \le x \le 0.1 \\ S_1(x) &= \frac{z_2 - z_1}{2(x_2 - x_1)}(x - x_1)^2 + z_1(x - x_1) + y_1 \\ &= -\frac{2170}{9}(x - 0.1)^2 + 64(x - 0.1) + 4.5 \\ &= -\frac{2170}{9}x^2 + \frac{1010}{9}x - \frac{194}{45}, \quad \text{untuk } 0.1 \le x \le 0.4 \\ S_2(x) &= \frac{z_3 - z_2}{2(x_3 - x_2)}(x - x_2)^2 + z_2(x - x_2) + y_2 \\ &= \frac{2450}{3}(x - 0.4)^2 - \frac{242}{3}(x - 0.4) + 2 \\ &= \frac{2450}{3}x^2 - \frac{2202}{3}x + \frac{4948}{30}, \quad \text{untuk } 0.4 \le x \le 0.5 \end{aligned}$$

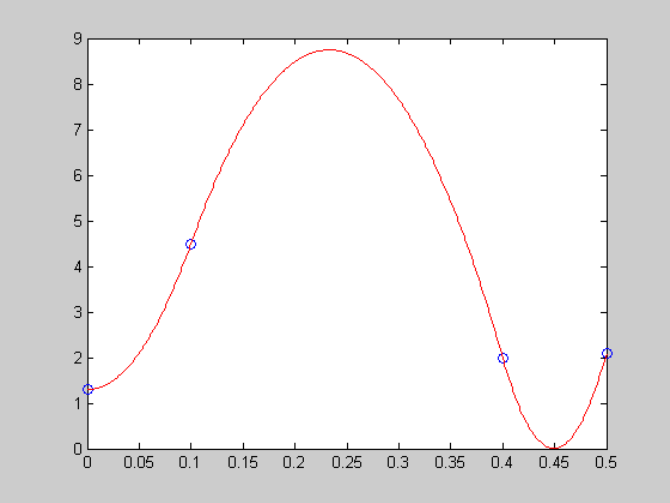
Persamaan (6.3) dapat dituliskan kembali sebagai

$$S_i(x) = a_i x^2 + b_i x + c_i, \quad i = 0, 1, ..., n-1 \tag{6.5}$$

dengan

$$a_i = \frac{z_{i+1} - z_i}{2\,(x_{i+1} - x_i)},\ b_i = z_i - 2a_i x_i,\ c_i = a_i x_i^2 - z_i x_i + y_i.$$

Fungsi MatLab `spline2()` mengkonstruksi (6.5) untuk koordinat-koordinat titik data (x, y) dengan suatu syarat batas $S'(x_0) = 0$. Fungsi ini juga menghitung taksiran nilai y untuk suatu nilai x di antara titik-titik data. Selain itu, kurva dari spline kuadratik untuk titik-titik data yang diberikan juga akan ditampilkan.



Gambar 6.2: Spline kuadratik untuk Contoh 6.3.

```
function [S,yi] = spline2(X,Y,xi)
% spline2  Mengkonstruksi fungsi spline kuadrat untuk titik-titik data (x,y).
%
% Input:  X = [x0,x1,...,xn]
%         Y = [y0,y1,...,yn]
%         xi = suatu nilai di antara x0 dan xn
%
% Output: S = [ai,bi,ci], koefisien-koefisien spline kuadrat y = ai*x^2+bi*x+ci
%         yi = taksiran nilai pada x = xi
%
% ---PENGHITUNGAN INTI:
if nargin < 3, xi = X(1); end
z0 = 0;
n = length(X);
S = [];
z_lama = z0;

% koefisien-koefisien spline linear:
for i=1:n-1
    z_baru = 2*(Y(i+1)-Y(i))/(X(i+1)-X(i))-z_lama;
    a = 0.5*(z_baru-z_lama)/(X(i+1)-X(i));
    b = z_baru-2*a*X(i+1);
    c = a*X(i)^2-z_lama*X(i)+Y(i);
    S = [S; a b c];
    z_lama = z_baru;
```

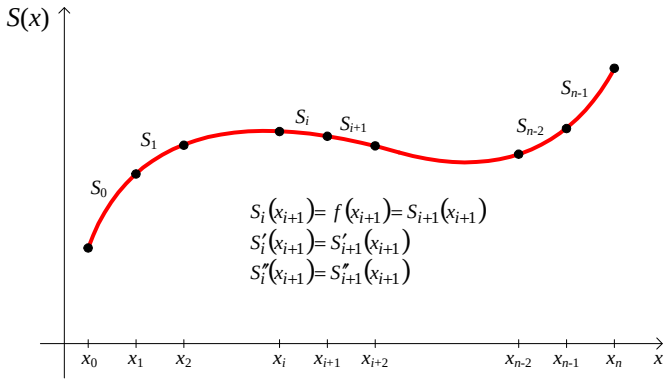
```
end

% nilai potongan fungsi f(xi):
for i=1:n-1
    if xi>X(i) & xi<X(i+1)
        yi = S(i,1)*xi^2+S(i,2)*xi+S(i,3);
    end
end

% plot kurva-kurva spline:
for i=1:n-1
    dX(i)=(X(i+1)-X(i));
end
dX = min(dX);
dx = dX/100; % lebar domain x

plot(X,Y,'bo')
for i=1:n-1
    hold on
    xx = X(i):dx:X(i+1);
    yy = S(i,1)*xx.^2+S(i,2)*xx+S(i,3);
    plot(xx,yy,'r-')
end
hold off
```

6.3 Spline Kubik



Gambar 6.3: Pendekatan dengan polinomial spline kubik.

Diketahui suatu fungsi $f(x)$ yang dibatasi oleh interval a dan b, dan memiliki sejumlah titik data $a = x_0 < x_1 < x_2 < ... < x_n = b$. Interpolasi spline kubik $S(x)$ adalah suatu potongan fungsi polinomial berderajat tiga (kubik) yang menghubungkan dua titik yang bersebelahan, lihat Gambar 6.3, dengan ketentuan, untuk $i = 0, 1, ..., n-1$:

(S0) Potongan fungsi pada subinterval $[x_i, x_{i+1}]$, $i = 0, 1, ..., n-1$:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$

(S1) Pada setiap titik data $x = x_i$, $i = 0, 1, ..., n$:

$$S(x_i) = f(x_i).$$

(S2) Nilai-nilai fungsi harus sama pada titik-titik dalam:

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}), \quad i = 0, 1, ..., n-2.$$

(S3) Turunan-turunan pertama pada titik dalam harus sama:

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad i = 0, 1, ..., n-2.$$

(S4) Turunan-turunan kedua pada titik dalam harus sama:

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \quad i = 0, 1, ..., n-2.$$

(S5) Salah satu syarat batas di antara dua syarat batas x_0 dan x_n berikut ini harus dipenuhi:

- $S''(x_0) = S''(x_n) = 0$ (disebut batas alamiah/ *natural boundary*)
- $S'(x_0) = f'(x_0)$ dan $S'(x_n) = f'(x_n)$ (disebut batas apitan/ *clamped boundary*)

Berikut ini akan dijelaskan bagaimana mencari polinomial spline kubik S.

Langkah 1 (Kendala S0): Polinomial spline kubik S untuk suatu fungsi f didefinisikan oleh

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{6.6}$$

dengan $i = 0, 1, ..., n-1$.

Langkah 2 (Kendala S1): Ketika $x = x_i$ maka dipunyai

$$\begin{aligned} S_i(x_i) &= a_i(x_i - x_i)^3 + b_i(x_i - x_i)^2 + c_i(x_i - x_i) + d_i \\ &= d_i = f(x_i). \end{aligned} \tag{6.7}$$

Ini berarti bahwa d_i selalu menjadi pasangan titik data dari x_i . Dengan pola ini maka pasangan titik data x_{i+1} adalah a_{i+1}, konsekuensinya $S(x_{i+1}) = d_{i+1}$.

Langkah 3 (Kendala S2): Ketika $x = x_{i+1}$ disubstitusikan ke persamaan $(S3)$ maka diperoleh

$$\begin{aligned} S_i(x_{i+1}) &= S_{i+1}(x_{i+1}) \\ a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + d_i &= d_{i+1} \end{aligned}$$

dengan $i = 0, 1, ..., n-2$. Sekarang dimisalkan $h_i = x_{i+1} - x_i$, sehingga persamaan di atas bisa dituliskan kembali menjadi

$$d_{i+1} = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i. \tag{6.8}$$

Langkah 4 (Kendala S3): Turunan pertama dari (6.6) yaitu

$$S_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i.$$

Ketika $x = x_i$ dipunyai

$$S_i'(x_i) = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i,$$

dan ketika $x = x_{i+1}$ dipunyai

$$\begin{aligned} S_i'(x_{i+1}) &= S_{i+1}'(x_{i+1}) \\ 3a_i(x_{i+1} - x_i)^2 + 2b_i(x_{i+1} - x_i) + c_i &= c_{i+1} \\ 3a_i h_i^2 + 2b_i h_i + c_i &= c_{i+1}. \end{aligned} \tag{6.9}$$

Langkah 5 (Kendala S4): Turunan kedua dari (6.6) yaitu

$$S_i''(x) = 6a_i(x - x_i) + 2b_i. \tag{6.10}$$

Dengan ketentuan tambahan $\dfrac{S''(x)}{2}$, persamaan di atas dimodifikasi menjadi

$$S_i''(x) = 3a_i(x - x_i) + b_i.$$

Ketika $x = x_i$ dipunyai

$$S_i''(x_i) = 3a_i(x_i - x_i) + b_i = b_i,$$

dan ketika $x = x_{i+1}$ dipunyai

$$\begin{aligned} S_i''(x_{i+1}) &= S_{i+1}''(x_{i+1}) \\ 3a_i(x_{i+1} - x_i) + b_i &= b_{i+1} \\ 3a_i h_i + b_i &= b_{i+1}. \end{aligned}$$

Dari sini bisa dinyatakan

$$a_i = \frac{1}{3h_i}(b_{i+1} - b_i). \tag{6.11}$$

Karena itu, persamaan (6.8) dapat dituliskan kembali menjadi

$$\begin{aligned} d_{i+1} &= \frac{1}{3h_i}(b_{i+1} - b_i)h_i^3 + b_i h_i^2 + c_i h_i + d_i \\ &= \frac{h_i^2}{3}(2b_i + b_{i+1}) + c_i h_i + d_i, \end{aligned} \tag{6.12}$$

sedangkan persamaan (6.9) menjadi

$$\begin{aligned} c_{i+1} &= h_i(b_{i+1} - b_i) + 2b_i h_i + c_i \\ &= h_i(b_i + b_{i+1}) + c_i \end{aligned}$$

atau dinyatakan menjadi

$$c_i = h_{i-1}(b_{i-1} + b_i) + c_{i-1}. \tag{6.13}$$

Dari persamaan (6.12) diperoleh

$$c_i = \frac{1}{h_i}\left(d_{i+1} - d_i\right) - \frac{h_i}{3}\left(2b_i + b_{i+1}\right) \tag{6.14}$$

dan untuk c_{i-1} dinyatakan

$$c_{i-1} = \frac{1}{h_{i-1}}\left(d_i - d_{i-1}\right) - \frac{h_{i-1}}{3}\left(2b_{i-1} + b_i\right). \tag{6.15}$$

Disubstitusikan c_i dan c_{i-1} ke persamaan (6.13) untuk memperoleh

$$h_{i-1}b_{i-1} + 2\left(h_{i-1} + h_i\right)b_i + h_i b_{i+1} = \frac{3}{h_i}\left(d_{i+1} - d_i\right) - \frac{3}{h_{i-1}}\left(d_i - d_{i-1}\right) \tag{6.16}$$

dengan $i = 1, 2, ..., n-1$. Dalam sistem persamaan ini, nilai $\{h_i\}_{i=0}^{n-1}$ dan nilai $\{d_i\}_{i=0}^{n}$ sudah diketahui, sedangkan nilai $\{b_i\}_{i=0}^{n}$ belum diketahui dan memang nilai inilah yang akan dihitung dari persamaan tersebut.

Langkah 6 (Kendala S5): (1) Batas alamiah. Ketika $S''(x_0) = S''(x_n) = 0$, dari persamaan (6.10) diperoleh

$$\begin{aligned} S''(x_0) &= 6a_0(x_0 - x_0) + 2b_0 = 0 \implies b_0 = 0, \\ S''(x_n) &= 6a_n(x_n - x_n) + 2b_n = 0 \implies b_n = 0. \end{aligned}$$

Sistem persamaan (6.16) dapat dituliskan sebagai perkalian matriks $\mathbf{Az} = \mathbf{B}$ dimana

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{z} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(d_2 - d_1) - \frac{3}{h_0}(d_1 - d_0) \\ \vdots \\ \frac{3}{h_{n-1}}(d_n - d_{n-1}) - \frac{3}{h_{n-2}}(d_{n-1} - d_{n-2}) \\ 0 \end{bmatrix}.$$

(2) Batas apitan. Dari persamaan (6.14), dengan $i = 0$, dimana $f'(x_0) = S'(x_0) = c_0$ diperoleh

$$f'(x_0) = \frac{1}{h_0}\left(d_1 - d_0\right) - \frac{h_0}{3}\left(2b_0 + b_1\right),$$

dan konsekuensinya

$$2h_0b_0 + h_0b_1 = \frac{3}{h_0}\left(d_1 - d_0\right) - 3c_0. \tag{6.17}$$

Sementara itu, pada $x = x_n$ dan dengan menggunakan persamaan (6.13)

diperoleh

$$f'(x_n) = c_n = c_{n-1} + h_{n-1}(b_{n-1} + b_n)$$

dengan c_{n-1} bisa diperoleh dari persamaan (6.15), dengan $i = n-1$,

$$c_{n-1} = \frac{1}{h_{n-1}}(d_n - d_{n-1}) - \frac{h_{n-1}}{3}(2b_{n-1} + b_n).$$

Jadi

$$\begin{aligned} f'(x_n) &= \frac{1}{h_{n-1}}(d_n - d_{n-1}) - \frac{h_{n-1}}{3}(2b_{n-1} + b_n) + h_{n-1}(b_{n-1} + b_n) \\ &= \frac{1}{h_{n-1}}(d_n - d_{n-1}) + \frac{h_{n-1}}{3}(b_{n-1} + 2b_n) \end{aligned}$$

dan akhirnya diperoleh

$$h_{n-1}b_{n-1} + 2h_{n-1}b_n = 3c_n - \frac{3}{h_{n-1}}(d_n - d_{n-1}) \tag{6.18}$$

Persamaan (6.17) dan (6.18) ditambah persamaan (6.16) membentuk perkalian matriks $\mathbf{Az} = \mathbf{B}$ dimana

$$\mathbf{A} = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix},$$

$$\mathbf{z} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}, \quad \mathbf{B} = 3\begin{bmatrix} \frac{1}{h_0}(d_1 - d_0) - c_0 \\ \frac{1}{h_1}(d_2 - d_1) - \frac{1}{h_0}(d_1 - d_0) \\ \vdots \\ \frac{1}{h_{n-1}}(d_n - d_{n-1}) - \frac{1}{h_{n-2}}(d_{n-1} - d_{n-2}) \\ c_n - \frac{1}{h_{n-1}}(d_n - d_{n-1}) \end{bmatrix}.$$

Langkah 7: Setelah sistem persamaan diselesaikan untuk b_0, b_1, ..., b_n, kita mensubstitusikan nilai-nilai tersebut ke kendala **(S1)**, (6.14), (6.11) untuk mendapatkan koefisien-koefisien lain dari spline kubik:

$$d_i \overset{\textbf{(S1)}}{=} y_i, \; c_i \overset{\textbf{(6.14)}}{=} \frac{d_{i+1} - d_i}{h_i} - \frac{h_i}{3}(2b_i + b_{i+1}), \; a_i \overset{\textbf{(6.14)}}{=} \frac{1}{3h_i}(b_{i+1} - b_i). \tag{6.19}$$

Contoh 6.4 *Konstruksikan spline kubik untuk* 4 *titik data berikut:*

x	0	1	2	3
y	0	1	4	5

terhadap syarat batas: $S'(x_0) = S'(0) = c_0 = 2$ *dan* $S'(x_n) = S'(3) = c_n = 2$.

Penyelesaian. Lebar subinterval pada sumbu x:

$$h_0 = h_1 = h_2 = h_3 = 1$$

dan beda terbagi pertama, dengan mengingat bahwa $d_i = f(x_i) = y_i$, yaitu

$$\frac{d_1 - d_0}{h_0} = 1,\ \frac{d_2 - d_1}{h_1} = 3,\ \frac{d_3 - d_2}{h_2} = 1.$$

Persamaan matriks dapat dituliskan sebagai

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = 3 \begin{bmatrix} 1-2 \\ 3-1 \\ 1-3 \\ 2-1 \end{bmatrix} = \begin{bmatrix} -3 \\ 6 \\ -6 \\ 3 \end{bmatrix},$$

yang mempunyai penyelesaian

$$b_0 = -3,\ b_1 = 3, b_2 = -3, \text{ dan } b_3 = 3.$$

Disubstitusikan penyelesaian tersebut ke persamaan (6.19) untuk memperoleh koefisien-koefisien lain dari spline kubik:

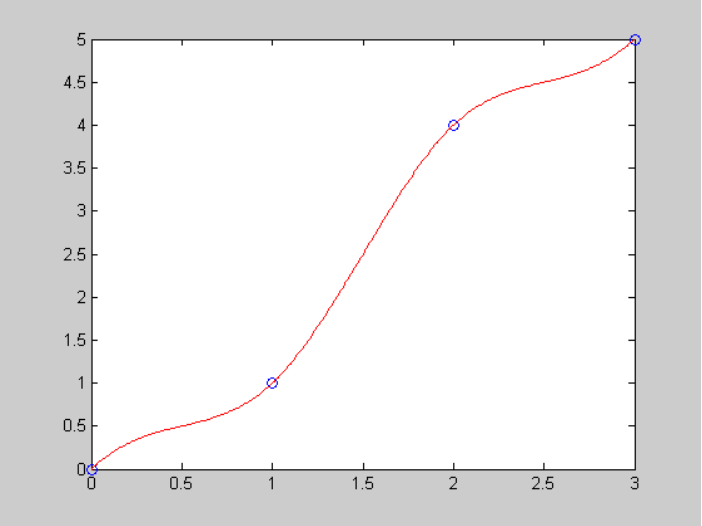
$$\begin{aligned} d_0 &= 0,\ d_1 = 1,\ d_2 = 4, \\ c_0 &= 1 - \frac{1}{3}(3 + 2(-3)) = 2, c_1 = 3 - \frac{1}{3}(-3 + 2(3)) = 2, c_2 = 1 - \frac{1}{3}(3 + 2(-3)) = 2, \\ a_0 &= \frac{3-(-3)}{3} = 2, a_1 = \frac{-3-3}{3} = -2, a_2 = \frac{3-(-3)}{3} = 2. \end{aligned}$$

Terakhir, kita dapat menuliskan persamaan spline kubik seperti

$$\begin{aligned} S_0(x) &= 2x^3 - 3x^2 + 2x, \quad \text{untuk } x \in [0,1]; \\ S_1(x) &= -2(x-1)^3 + 3(x-1)^2 + 2(x-1) + 1, \quad \text{untuk } x \in [1,2]; \\ S_2(x) &= 2(x-2)^3 - 3(x-2)^2 + 2(x-1) + 4, \quad \text{untuk } x \in [2,3]. \end{aligned}$$

Fungsi MatLab `splinekubik()` mengkonstruksi persamaan matriks $\mathbf{Ax} = \mathbf{b}$, menyelesaikannya menggunakan iterasi SOR untuk mendapatkan koefisien-koefisien spline kubik untuk koordinat-koordinat x, y dari titik-titik data yang dilengkapi syarat batas apitan.

```
function [S,yi] = splinekubik(X,Y,dy0,dyn,xi)
% splinekubik  Mengkonstruksi polinomial spline kubik untuk titik-titik data (x,y).
%
% Input:  X = [x0,x1,...,xn]
%         Y = [y0,y1,...,yn]
%         dy0 = S'(x0) = c0: turunan awal
%         dyn = S'(xn) = cn: turunan akhir
%         xi = suatu nilai di antara x0 dan xn
%
% Output: S = [ai,bi,ci,di], koefisien-koefisien spline kubik
```



Gambar 6.4: Spline kuadratik untuk Contoh 6.4.

```
%                           Si(x)= ai*(x-xi)^3+bi*(x-xi)^2+ci(x-xi)+di
%         yi = taksiran nilai pada x = xi
%
% ---PENGHITUNGAN INTI:
if nargin < 5, xi = X(1); end
if nargin < 4, dyn = 0; end
if nargin < 3, dy0 = 0; end
n = length(X);
for i = 1:n-1 % lebar subinterval
    h(i) = X(i+1) - X(i);
end

% mengkonstruksi matriks A dan B:
A = zeros(n,n); B = zeros(n,1);
A(1,1) = 2*h(1); A(1,2) = h(1); A(2,1) = h(1);
A(n,n) = 2*h(n-1); A(n-1,n) = h(n-1); A(n,n-1) = h(n-1);

for i = 2:n-1
    for j = 2:n-1
        if j==i, A(i,j) = 2*(h(i-1)+h(i));
        elseif j==i+1, A(i,j) = h(i);
        elseif j==i-1, A(i,j) = h(i);
        else A(i,j) = 0;
        end
    end
end

B(1) = (Y(2)-Y(1))/h(1) - dy0;
B(n) = dyn - (Y(n)-Y(n-1))/h(n-1);
for i = 2:n-1
    B(i) = (Y(i+1)-Y(i))/h(i) - (Y(i)-Y(i-1))/h(i-1);
end
B = 3*B;

% koefisien-koefisien spline kubik:
z = iterasiSOR(A,B);
```

```
bi = z(1:3);
di = Y(1:3)';
for i = 1: n-1
    ci(i) = (Y(i+1)-Y(i))/h(i) - h(i)/3*(2*z(i)+z(i+1));
    ai(i) = (z(i+1)-z(i))/(3*h(i));
end
S = [ai' bi ci' di]; % orde menurun

% nilai potongan fungsi f(xi):
for i=1:n-1
    if xi>X(i) & xi<X(i+1)
    yi = S(i,1)*(xi-X(i))^3+S(i,2)*(xi-X(i))^2+S(i,3)*(xi-X(i))+S(i,4);
    end
end

% plot kurva-kurva spline:
for i=1:n-1
    dX(i)=(X(i+1)-X(i));
end
dX = min(dX);
dx = dX/100; % lebar domain x

plot(X,Y,'bo')
for i=1:n-1
    hold on
    xx = X(i):dx:X(i+1);
    yy = S(i,1)*(xx-X(i)).^3+S(i,2)*(xx-X(i)).^2+S(i,3)*(xx-X(i))+S(i,4);
    plot(xx,yy,'r-')
end
hold off
```

Bab 7

Regresi Kuadrat Terkecil

Tujuan Pembelajaran:
- Mengetahui teknik pencocokan kurva yang merepresentasikan *trend* data.
- Mengaplikasikan regresi kuadrat terkecil untuk menentukan fungsi hampiran.

Jika terdapat banyak galat yang berhubungan dengan data, khususnya data eksperimental, maka interpolasi tidak sesuai dan mungkin memberikan hasil-hasil yang tidak memuaskan jika dipakai untuk meramalkan nilai-nilai antara. Untuk kasus yang demikian, suatu strategi yang sesuai adalah dengan pencocokan kurva. Pencocokan kurva adalah pencarian suatu kurva yang bisa menunjukkan kecenderungan (*trend*) dari himpunan data. Kurva ini tidak harus melalui titik-titik data. Suatu kriteria yang dipakai untuk mengukur kecukupan dari kecocokan yaitu regresi kuadrat terkecil.

7.1 Regresi Linear

Regresi linier digunakan untuk menentukan fungsi linier yang paling sesuai dengan kumpulan titik data $\{(x_k, y_k) : k = 1, ..., n\}$ yang diketahui. Pernyataan matematis untuk fungsi linear tersebut yaitu

$$y = a_0 + a_1 x + e$$

dengan e dinamakan galat atau sisa. Sisa adalah selisih antara pengamatan dengan garis:

$$e = y - a_0 + a_1 x.$$

Suatu kriteria untuk pencocokan yang terbaik adalah hampiran kuadrat terkecil yang meminimalkan jumlahan kuadrat dari sisa:

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_{i,observasi} - y_{i,data})^2 = \sum_{i=1}^{n} (y_i - a_0 + a_1 x_i)^2 .$$

Kriteria ini menghasilkan suatu garis tunggal untuk himpunan data yang diberikan. Untuk menentukan nilai-nilai a_0 dan a_1, diturunkan S_r terhadap setiap koefisien dan

selanjutnya disamakan dengan nol:

$$\begin{aligned}\frac{\partial S}{\partial a_0} &= -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i) = 0,\\ \frac{\partial S}{\partial a_1} &= -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)\, x_i = 0.\end{aligned}$$

Persamaan-persamaan di atas dapat dituliskan kembali menjadi

$$\begin{aligned}\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a_0 - \sum_{i=1}^{n} a_1 x_i &= 0,\\ \sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n} a_0 x_i - \sum_{i=1}^{n} a_1 x_i^2 &= 0,\end{aligned}$$

atau ekivalen dengan

$$\begin{aligned}n a_0 + a_1 \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} y_i,\\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} y_i x_i.\end{aligned}$$

Selanjutnya diselesaikan kedua persamaan untuk memperoleh

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \quad \text{dan} \quad a_0 = \frac{1}{n}\sum_{i=1}^{n} y_i - a_1 \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Contoh 7.1 *Berikut ini akan dicari persamaan kurva linear jika diberikan tujuh data untuk x dan y:*

x_i	1	2	3	4	5	6	7
y_i	0.5	2.5	2.0	4.0	3.5	6.0	5.5

Berdasarkan data diperoleh:

$$n = 7, \sum_{i=1}^{7} x_i y_i = 119.5, \sum_{i=1}^{7} x_i = 28, \sum_{i=1}^{7} y_i = 24, \sum_{i=1}^{7} x_i^2 = 140.$$

Karena itu

$$\begin{aligned}a_1 &= \frac{7 \cdot 119.5 - 28 \cdot 24}{7 \cdot 140 - (28)^2} = 0.8392857,\\ a_0 &= \frac{24}{7} - 0.8392857 \cdot \frac{28}{7} = 0.07142857,\end{aligned}$$

sehingga persamaan kurva linear:

$$y = 0.07142857 + 0.8392857x.$$

Fungsi MatLab `reglin()` menjalankan skema kuadrat terkecil untuk mencari koefisien dari suatu persamaan kurva linear yang cocok dengan sekumpulan titik data. Fungsi ini juga menampilkan plot titik-titik data dan kurva regresi linear.

```
function Koef = reglin(X,Y)
% reglin  Mencari koefisien dari persamaan kurva linear untuk sekumpulan
%         titik data menggunakan kriteria kuadrat terkecil.
%
% Input:  X = data x
%         Y = data y yang berkorespondensi dengan x
%
% Output: koefisien dari x^0 dan x^1 secara berturutan
%
% ---PENGHITUNGAN INTI:
n = length(X);
sigmaXY = sum(X.*Y);
sigmaX = sum(X);
sigmaY = sum(Y);
sigmaXX = sum(X.^2);
koefx1 = (n*sigmaXY-sigmaX*sigmaY)/(n*sigmaXX-sigmaX^2); % koefisien dari x^1
koefx0 = (sigmaY-koefx1*sigmaX)/n; % koefisien dari x^0

% ---OUTPUT:
Koef = [koefx0,koefx1]; % orde naik

% plot titik-titik data dan kurva linear
xreg = X(1):0.01:X(end);
yreg = Koef(1)+Koef(2)*xreg;
plot(X,Y,'bo',xreg,yreg,'r-');
```

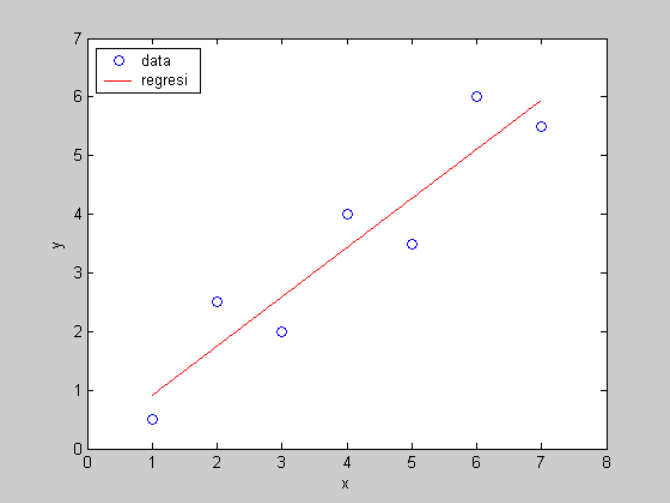
7.2 Regresi Polinomial Derajat Dua

Diberikan kumpulan titik data $\{(x_k, y_k) : k = 1, ..., n\}$. Fungsi pendekatan untuk fungsi polinomial berderajat dua:

$$y = a_0 + a_1 x + a_2 x^2.$$

Jumlahan kuadrat dari sisa yaitu

$$S_r = \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right)^2.$$



Gambar 7.1: Kurva linear dengan metode kuadrat terkecil untuk titik-titik data dalam Contoh 7.1.

Diturunkan S_r terhadap semua parameter dan selanjutnya disamakan dengan nol:

$$\begin{aligned}
\frac{\partial S_r}{\partial a_0} &= -2\sum_{i=1}^{n}\left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right) = 0, \\
\frac{\partial S_r}{\partial a_1} &= -2\sum_{i=1}^{n}\left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right) x_i = 0, \\
\frac{\partial S_r}{\partial a_1} &= -2\sum_{i=1}^{n}\left(y_i - a_0 - a_1 x_i - a_2 x_i^2\right) x_i^2 = 0
\end{aligned}$$

Persamaan-persamaan disusun kembali menjadi

$$\begin{aligned}
na_0 + a_1\sum_{i=1}^{n} x_i + a_2\sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} y_i, \\
a_0\sum_{i=1}^{n} x_i + a_1\sum_{i=1}^{n} x_i^2 + a_2\sum_{i=1}^{n} x_i^3 &= \sum_{i=1}^{n} x_i y_i, \\
a_0\sum_{i=1}^{n} x_i^2 + a_1\sum_{i=1}^{n} x_i^3 + a_2\sum_{i=1}^{n} x_i^4 &= \sum_{i=1}^{n} x_i^2 y_i,
\end{aligned}$$

atau dalam bentuk perkalian matriks:

$$\begin{bmatrix}
n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\
\sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\
\sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}
=
\begin{bmatrix}
\sum_{i=1}^{n} y_i \\
\sum_{i=1}^{n} x_i y_i \\
\sum_{i=1}^{n} x_i^2 y_i
\end{bmatrix}.$$

Contoh 7.2 *Berikut ini akan dicari persamaan kurva polinomial derajat dua untuk data:*

x_i	0	1	2	3	4	5
y_i	2.1	7.7	14	27	41	61

Berdasarkan data diperoleh:

$$\begin{aligned} n &= 6, \sum_{i=1}^{6} x_i = 15, \sum_{i=1}^{6} x_i^2 = 55, \sum_{i=1}^{6} y_i = 152.6, \sum_{i=1}^{6} x_i^3 = 225, \\ \sum_{i=1}^{6} x_i y_i &= 585.6, \sum_{i=1}^{6} x_i^4 = 979, \sum_{i=1}^{6} x_i^2 y_i = 2488.8. \end{aligned}$$

Karena itu dipunyai sistem persamaan linear dalam bentuk

$$\begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 152.6 \\ 585.6 \\ 2488.8 \end{bmatrix}$$

yang mempunyai penyelesaian

$$a_0 = 2.47857, a_1 = 2.35929, a_2 = 1.86071,$$

sehingga persamaan kurva polinomial derajat dua:

$$y = 2.47857 + 2.35929x + 1.86071x^2.$$

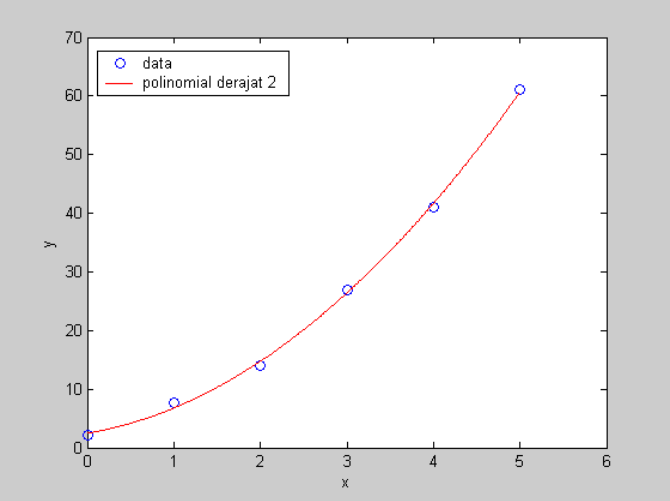
Fungsi MatLab `regkuad()` mencari koefisien dari suatu persamaan polinomial derajat dua yang cocok dengan sekumpulan titik data dengan metode kuadrat terkecil.

```
function Koef = regkuad(X,Y)
% reglin  Mencari koefisien dari persamaan polinomial derajat dua untuk
%         sekumpulan titik data menggunakan kriteria kuadrat terkecil.
%
% Input:  X = data x
%         Y = data y yang berkorespondensi dengan x
%
% Output: koefisien dari x^0, x^1, dan x^2 secara berturutan
%
% ---PENGHITUNGAN INTI:
% penyusun sistem persamaan:
A(1,1) = length(X);
A(1,2) = sum(X);
A(1,3) = sum(X.^2);
A(2,1) = A(1,2);
A(2,2) = A(1,3);
A(2,3) = sum(X.^3);
A(3,1) = A(2,2);
A(3,2) = A(2,3);
A(3,3) = sum(X.^4);
b(1) = sum(Y);
b(2) = sum(X.*Y);
b(3) = sum(X.^2.*Y);
```

```
b = b';

% penyelesaian sistem dengan SOR:
Koef = SOR(A,b)'; % orde naik

% plot titik-titik data dan kurva regresi kuadrat
xreg = X(1):0.01:X(end);
yreg = Koef(1)+Koef(2)*xreg+Koef(3)*xreg.^2;
plot(X,Y,'bo',xreg,yreg,'r-');
```



Gambar 7.2: Kurva polinomial derajat dua dengan metode kuadrat terkecil untuk titik-titik data dalam Contoh 7.2.

7.3 Linearisasi Fungsi Tak Linear

Regresi linear memberikan teknik yang ampuh untuk mencocokkan garis "terbaik" terhadap data. Namun, teknik ini tergantung pada kenyataan bahwa kaitan antara variabel tak bebas dan bebas adalah linear. Dalam analisis regresi, seharusnya langkah pertama adalah penggambaran grafik untuk memeriksa apakah pada data berlaku suatu hubungan linear.

Beberapa data yang tidak linear dapat dilinearkan dengan suatu transformasi data, seperti yang disajikan dalam Tabel 7.1.

Contoh 7.3 *Berikut ini akan dicari persamaan kurva eksponensial untuk data:*

x_i	1	2	3	4	5
y_i	0.5	1.7	3.4	5.7	8.4
$\ln(y_i)$	-0.6931	0.5306	1.2238	1.7405	2.1282

Tabel 7.1: Linearisasi dari fungsi tak linear dengan transformasi data.

Fungsi Pencocokan	Fungsi Linearisasi	Substitusi Variabel/ Perbaikan Parameter
(1) $y = \frac{a}{x} + b$	$y = aX + b$	$X = \frac{1}{x}$
(2) $y = \frac{b}{a + x}$	$Y = Ax + B$	$Y = \frac{1}{y}, A = \frac{1}{b}, B = \frac{a}{b}$
(3) $y = a \cdot b^x$	$Y = Ax + B$	$Y = \ln(y), A = \ln(b), B = \ln(a)$
(4) $y = b \cdot e^{ax}$	$Y = ax + B$	$Y = \ln(y), B = \ln(b)$
(5) $y = C - b \cdot e^{-ax}$	$Y = Ax + B$	$Y = \ln(C - y), A = -a, B = \ln(b)$
(6) $y = a \cdot x^b$	$Y = AX + B$	$Y = \ln(y), A = b, X = \ln(x), B = \ln(a)$
(7) $y = ax \cdot e^{bx}$	$Y = Ax + B$	$Y = \ln\left(\frac{y}{x}\right), A = b, B = \ln(a)$
(8) $y = \frac{C}{1 + b \cdot e^{ax}}$	$Y = ax + B$	$Y = \ln\left(\frac{C}{y} - 1\right), B = \ln(b)$
(9) $y = a \ln(x) + b$	$y = aX + b$	$X = \ln(x)$

Jadi di sini digunakan fungsi linearisasi seperti no. 4 dalam Tabel 7.1. Berdasarkan data diperoleh:

$$n = 5, \sum_{i=1}^{5} x_i z_i = 21.6425, \sum_{i=1}^{5} x_i = 15, \sum_{i=1}^{5} z_i = 4.93, \sum_{i=1}^{5} x_i^2 = 55.$$

Karena itu

$$\begin{aligned} a &= \frac{5 \cdot 21.6425 - 15 \cdot 4.93}{5 \cdot 55 - (15)^2} = 0.685, \\ B &= \frac{4.93}{5} - 0.685 \cdot \frac{15}{5} = -1.069, \end{aligned}$$

sehingga persamaan kurva linear:

$$Y = 0.685 - 1.069x$$

atau persamaan kurva eksponensial, dengan $b = e^B = e^{-1.069}$*, yaitu*

$$y = e^{0.685x - 1.069}$$

Bab 8

Diferensiasi Numerik

Tujuan Pembelajaran:
- Mencari rumus hampiran beda untuk turunan pertama dan kedua.
- Mengetahui rumus hampiran turunan yang mempunyai galat lebih baik.
- Mengaplikasikan rumus hampiran beda untuk suatu fungsi.

8.1 Turunan Pertama

Suatu teori sederhana mengenai hampiran numerik untuk turunan dapat diperoleh melalui ekspansi deret Taylor dari $f(x+h)$ di sekitar x:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots . \tag{8.1}$$

Pengurangan $f(x)$ dari kedua sisi dan membagi kedua sisi dengan ukuran langkah h menghasilkan

$$\begin{aligned} D_{f1}(x,h) &= \frac{f(x+h)-f(x)}{h} = f'(x) + \frac{h}{2!}f''(x) + \frac{h^2}{3!}f'''(x) + \cdots \\ &= f'(x) + O(h), \end{aligned} \tag{8.2}$$

dimana $O(h)$ menyatakan suku galat pemotongan yang sebanding dengan h untuk $|h| \prec 1$. Dari sini kita mendapatkan **hampiran beda maju** (*forward difference approximation*) untuk $f'(x)$:

$$D_{f1}(x,h) = \frac{f(x+h)-f(x)}{h}$$

yang mempunyai galat sebanding dengan ukuran langkah h atau ekivalen dalam orde dari h.

Sekarang, untuk menurunkan rumus hampiran yang lain untuk turunan pertama yang mempunyai galat lebih kecil, dihapus suku orde satu terhadap h dari persamaan (8.2) dengan mensubstitusikan $2h$ untuk h dalam persamaan:

$$D_{f1}(x,2h) = \frac{f(x+2h)-f(x)}{2h} = f'(x) + \frac{2h}{2!}f''(x) + \frac{4h^2}{3!}f'''(x) + \cdots$$

dan pengurangan hasil ini dari dua kali persamaan (8.2) menghasilkan

$$\begin{aligned} 2D_{f1}(x,h) - D_{f1}(x,2h) &= 2\frac{f(x+h)-f(x)}{h} - \frac{f(x+2h)-f(x)}{2h} \\ D_{f2}(x,h) &= \frac{-f(x+2h)+4f(x+h)-3f(x)}{2h} \\ &= f'(x) - \frac{2h^2}{3!}f'''(x) + \cdots \\ &= f'(x) + O\left(h^2\right) \end{aligned} \tag{8.3}$$

yang dapat dipandang sebagai suatu perbaikan atas persamaan (8.2) karena ini mempunyai galat pemotongan sebanding dengan h^2 untuk $|h| \prec 1$.

Berikutnya, dengan mensubstitusikan $-h$ untuk h dalam persamaan (8.1) akan diperoleh **hampiran beda mundur** (*backward difference approximation*) untuk $f'(x)$:

$$D_{b1}(x,h) = \frac{f(x)-f(x-h)}{h} = D_{f1}(x,-h)$$

yang galat pemotongan yang sebanding dengan h untuk $|h| \prec 1$. Untuk menghasilkan suatu versi perbaikan yang mempunyai galat pemotongan sebanding dengan h^2 untuk $|h| \prec 1$ dapat diproses seperti berikut:

$$\begin{aligned} D_{b2}(x,h) &= \frac{2D_{b1}(x,h) - D_{b1}(x,2h)}{2-1} = \frac{3f(x)-4f(x-h)+f(x-2h)}{2h} \\ &= f'(x) + O\left(h^2\right). \end{aligned} \tag{8.4}$$

Untuk menurunkan rumus hampiran lain untuk turunan pertama, kita mengambil ekspansi deret Taylor dari $f(x+h)$ dan $f(x-h)$ sampai orde kelima:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2!}h^2f''(x) + \frac{1}{3!}h^3f'''(x) + \frac{1}{4!}h^4f^{(4)}(x) \\ &\quad + \frac{1}{5!}h^5f^{(5)}(x) + \cdots, \end{aligned} \tag{8.5}$$

$$\begin{aligned} f(x-h) &= f(x) - hf'(x) + \frac{1}{2!}h^2f''(x) - \frac{1}{3!}h^3f'''(x) + \frac{1}{4!}h^4f^{(4)}(x) \\ &\quad - \frac{1}{5!}h^5f^{(5)}(x) + \cdots \end{aligned} \tag{8.6}$$

dan membagi selisih antara kedua persamaan dengan $2h$ untuk mendapatkan **hampiran beda pusat** (*central difference approximation*) untuk $f'(x)$:

$$\begin{aligned} D_{c2}(x,h) &= \frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{1}{3!}h^2f'''(x) + \frac{1}{5!}h^4f^{(5)}(x) + \cdots \\ &= f'(x) + O\left(h^2\right) \end{aligned}$$

yang mempunyai galat sebanding dengan h^2, serupa dengan persamaan (8.3) dan (8.4). Ini juga dapat diproses untuk menghasilkan suatu versi perbaikan yang mempunyai

galat pemotongan sebanding dengan h^4:

$$\begin{aligned} 2^2 D_{c2}(x,h) - D_{c2}(x,2h) &= 4\frac{f(x+h)-f(x-h)}{2h} - \frac{f(x+2h)-f(x-2h)}{2\cdot 2h} \\ &= 3f'(x) - \frac{12h^4}{5!}f^{(5)}(x) - \cdots \\ D_{c4}(x,h) &= \frac{2^2 D_{c2}(x,h) - D_{c2}(x,2h)}{2^2-1} \\ &= \frac{8f(x+h) - 8f(x-h) - f(x+2h) + f(x-2h)}{12h} \\ &= f'(x) + O\left(h^4\right) \end{aligned}$$

Contoh 8.1 *Hitung nilai turunan $f(x) = x^2$ pada $x = 2$ dengan $h = 0.1$.*

Penyelesaian. Hampiran beda maju:

$$\begin{aligned} O(h) &: D_{f1}(2, 0.1) = \frac{f(2.1) - f(2)}{0.1} = \frac{4.41 - 4}{0.1} = 4.1; \\ O\left(h^2\right) &: D_{f2}(2, 0.1) = \frac{-f(2.2) + 4f(2.1) - 3f(2)}{0.2} = \frac{-4.84 + 17.64 - 12}{0.2} = 4. \end{aligned}$$

Hampiran beda mundur:

$$\begin{aligned} O(h) &: D_{b1}(2, 0.1) = \frac{f(2) - f(1.9)}{0.1} = \frac{4 - 3.61}{0.1} = 3.9; \\ O\left(h^2\right) &: D_{b2}(2, 0.1) = \frac{3f(2) - 4f(1.9) + f(1.8)}{0.2} = \frac{12 - 14.44 + 3.24}{0.2} = 4. \end{aligned}$$

Hampiran beda pusat:

$$\begin{aligned} O\left(h^2\right) &: D_{c2}(2, 0.1) = \frac{f(2.1) - f(1.9)}{0.2} = \frac{4.41 - 3.61}{0.2} = 4; \\ O\left(h^4\right) &: D_{c4}(2, 0.1) = \frac{8f(2.1) - 8f(1.9) - f(2.2) + f(1.8)}{1.2} = 4. \end{aligned}$$

Berikut ini adalah kode MatLab untuk mencari hampiran beda pusat orde dua untuk fungsi $f(x) = \sin(x)$ di $x = \frac{1}{3}\pi$ untuk h yang bevariasi.

```
f = inline('sin(x)');
f1 = inline('cos(x)');
h = 0.1;
x = pi/3;

fprintf('    h    Galat Mutlak\n');
fprintf('=========================\n');
for i = 1:6
    Dc2 = (f(x+h) - f(x-h))/(2*h);
    fprintf('%7.1e %8.1e\n',h,abs(f1(x)-Dc2))
```

```
    h = h/10;
end
```

Kode MatLab tersebut menghasilkan tabel seperti berikut:

```
    h    Galat Mutlak
========================
1.0e-001 8.3e-004
1.0e-002 8.3e-006
1.0e-003 8.3e-008
1.0e-004 8.3e-010
1.0e-005 7.8e-012
1.0e-006 4.1e-011
```

8.2 Ekstrapolasi Richardson

Prosedur yang sudah diturunkan untuk hampiran beda pada bagian sebelumnya dapat dirumuskan menjadi suatu rumus umum, dinamakan ekstrapolasi Richardson, untuk memperbaiki hampiran beda dari turunan-turunan seperti berikut ini:

$$\begin{aligned} D_{f,n+1}(x,h) &= \frac{2^n D_{f,n}(x,h) - D_{f,n}(x,2h)}{2^n - 1}; \qquad [n\text{: orde galat}] \\ D_{b,n+1}(x,h) &= \frac{2^n D_{b,n}(x,h) - D_{b,n}(x,2h)}{2^n - 1}; \\ D_{c,2(n+1)}(x,h) &= \frac{4^n D_{c,2n}(x,h) - D_{c,2n}(x,2h)}{4^n - 1}. \end{aligned}$$

Seperti dalam beda bagi, penghitungan hampiran-hampiran beda juga dapat disusun dalam bentuk tabel piramida. Sebagai contoh, konstruksi ekstrapolasi Richardson untuk beda maju sampai hampiran orde 4 diberikan seperti dalam Tabel **??**. Sementara itu, untuk beda pusat sampai hampiran orde 8 diberikan seperti dalam Tabel 8.2.

Tabel 8.1: Tabel ekstrapolasi Richardson untuk beda maju sampai hampiran orde 4.

$O(h)$	$O(h^2)$	$O(h^3)$	$O(h^4)$
1: $D_{f1}(x,h)$	3: $D_{f2}(x,h)$	6: $D_{f3}(x,h)$	10: $D_{f4}(x,h)$
2: $D_{f1}(x,2h)$	5: $D_{f2}(x,2h)$	9: $D_{f3}(x,2h)$	
4: $D_{f1}(x,4h)$	8: $D_{f2}(x,4h)$		
7: $D_{f1}(x,8h)$			

Fungsi MatLab `RichDiff1()` mengkonstruksi tabel ekstrapolasi dan mendapatkan hampiran beda pusat untuk turunan pertama sampai orde n dimana n adalah bilangan genap positif.

Tabel 8.2: Tabel ekstrapolasi Richardson untuk beda pusat sampai hampiran orde 8.

$O\left(h^2\right)$	$O\left(h^4\right)$	$O\left(h^6\right)$	$O\left(h^8\right)$
1: $D_{c2}\left(x,h\right)$	3: $D_{c4}\left(x,h\right)$	6: $D_{c6}\left(x,h\right)$	10: $D_{c8}\left(x,h\right)$
2: $D_{c2}\left(x,2h\right)$	5: $D_{c4}\left(x,2h\right)$	9: $D_{c6}\left(x,2h\right)$	
4: $D_{c2}\left(x,4h\right)$	8: $D_{c4}\left(x,4h\right)$		
7: $D_{c2}\left(x,8h\right)$			

```
function [Dcf1,Dc] = RichDiff1(f,x0,h,n)
% RichDiff1  Mengkonstruksi ekstrapolasi Richardson untuk hampiran beda pusat dari
%            turunan pertama fungsi f(x).
% Input:  f = fungsi f(x)
%         x0 = nilai dimana turunan f'(x0) akan dicari
%         h = ukuran langkah
%         n = orde galat pemotongan (harus bilangan genap positif)
% Output: Dcf1 = nilai f'(x0)

N = n/2;
Dc = zeros(N,N); % matriks turunan-turunan Richardson

% kolom pertama dalam matriks Dc: Dc2(x,2^k*h)
for i = 1:N
    x = x0 + 2^(i-1)*h;
    f1 = f(x);
    x = x0 - 2^(i-1)*h;
    f2 = f(x);
    Dc(i,1) = (f1-f2)/(2^i*h); % hampiran turunan-turunan pertama untuk beda pusat
end

% matriks Dc mulai kolom kedua
for j = 2:N
    for i = 1:N+1-j
        Dc(i,j) = (4^(j-1)*Dc(i,j-1)-Dc(i+1,j-1))/(4^(j-1)-1);
    end
end

% Turunan pertama untuk beda pusat sampai orde n:
Dcf1 = Dc(1,end);
```

8.3 Turunan Kedua

Untuk memperoleh suatu rumus hampiran untuk turunan kedua, kita mengambil ekspansi deret Taylor dari $f\left(x+h\right)$ dan $f\left(x-h\right)$ sampai orde kelima seperti dalam persamaan (8.5) dan (8.6). Dijumlahkan kedua persamaan (untuk menghapus suku $f'\left(x\right)$) dan selanjutnya dilakukan pengurangan $2f\left(x\right)$ dari kedua sisi dan dibagi kedua sisi dengan h^2 untuk memperoleh hampiran beda pusat untuk turunan kedua:

$$\begin{aligned} D_{c2}^{(2)}\left(x,h\right) &= \frac{f\left(x+h\right)-2f\left(x\right)+f\left(x-h\right)}{h^2} \\ &= f''\left(x\right)+\frac{1}{12}h^2f^{(4)}\left(x\right)+\frac{2}{6!}h^4f^{(6)}\left(x\right)+\cdots \end{aligned}$$

yang mempunyai suatu galat pemotongan orde h^2.

Ekstrapolasi Richardson dapat digunakan untuk memanipulasi persamaan di atas sehingga suku h^2 terhapus. Hasil yang diperoleh adalah suatu versi perbaikan:

$$\frac{2^2 D_{c2}^{(2)}(x,h) - D_{c2}^{(2)}(x,2h)}{2^2-1} = \frac{-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)}{12h^2} = f''(x) - \frac{h^4}{90} f^{(5)}(x) + \cdots .$$

atau dituliskan kembali menjadi

$$\begin{aligned} D_{c4}^{(2)}(x,h) &= \frac{-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)}{12h^2} \\ &= f''(x) + O\left(h^4\right) \end{aligned}$$

yang mempunyai galat pemotongan orde h^4.

Kode MatLab untuk mengkonstruksi tabel ekstrapolasi dan mendapatkan hampiran beda pusat untuk turunan kedua adalah sama seperti untuk turunan pertama tetapi penghitungan hampiran turunan-turunan pertama orde 2 pada kolom pertama diganti dengan rumus turunan kedua:

```
Dc(i,1) = (f1-2*f(x0)+f2)/(2^(i-1)*h)^2;
```

Lebih lanjut, untuk mendapatkan rumus hampiran dari turunan tingkat tinggi dapat dilihat di [5].

Bab 9

Persamaan Diferensial Biasa

Tujuan Pembelajaran:
- Mengetahui metode-metode penyelesaian numeris untuk persamaan diferensial biasa.
- Mengetahui metode yang memberikan hasil paling dekat dengan hasil sesungguhnya.
- Mengaplikasikan metode untuk menyelesaikan persamaan diferensial biasa.

Dalam bab ini, kita akan membicarakan beberapa metode penyelesaian numeris dari persamaan diferensial biasa (disingkat PDB) dimana variabel tak bebas y tergantung pada variabel bebas tunggal t.

9.1 Metode Euler

Diperhatikan persamaan diferensial tingkat satu:

$$\frac{dy\,(t)}{dt} = f\,(t, y\,(t))\,, \qquad a \leq t \leq b, \qquad y\,(a) = \alpha. \tag{9.1}$$

Tahap awal penyelesaian pendekatan numerik adalah dengan menentukan titik-titik dalam jarak yang sama di dalam interval $[a, b]$, yaitu dengan menerapkan

$$t_i = a + i \cdot h, \qquad i = 0, 1, ..., n$$

dimana h menyatakan jarak antar titik yang dirumuskan oleh

$$h = \frac{b-a}{n}$$

yang juga biasa dikenal sebagai lebar langkah (*step size*).

Berikutnya, turunan dalam persamaan diferensial diganti dengan suatu turunan numerik (diperkenalkan dalam bab sebelumnya). Untuk kesepakatan diperkenalkan singkatan $y_i = y\,(t_i)$. Metode Euler menghampiri turunan pertama di $t = t_i$ dalam persamaan (9.1) dengan persamaan

$$y_i' = \frac{dy_i}{dt} \approx \frac{y_{i+1} - y_i}{t_{i+1} - t_i} = \frac{y_{i+1} - y_i}{h}.$$

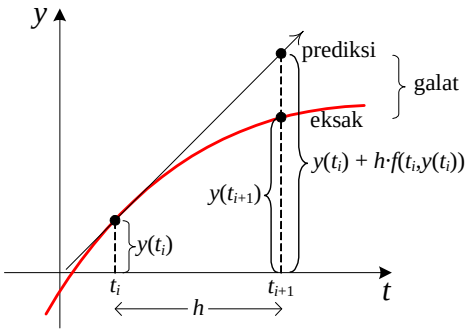
Karena itu, pada saat $t = t_i$, persamaan (9.1) dapat dituliskan sebagai

$$\frac{y_{i+1} - y_i}{h} \approx f\,(t_i, y_i)\,.$$

Jadi, metode Euler mendapatkan barisan numerik $\{y_i\}_{i=0}^{n}$ yang dinyatakan sebagai

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &\approx y_i + hf\left(t_i, y_i\right), \qquad i = 0, 1, 2, ..., n-1. \end{aligned} \tag{9.2}$$

Pengertian geometris untuk metode Euler diberikan dalam Gambar 9.1.



Gambar 9.1: Ilustrasi dari penurunan metode Euler.

Contoh 9.1 *Selesaikan persamaan diferensial*

$$y'\left(t\right) = y\left(t\right) - t^2 + 1, \qquad 0 \leq t \leq 2, \qquad y\left(0\right) = 0.5$$

menggunakan metode Euler dimana $n = 10$.

Penyelesaian. Dicari jarak antar titik dalam interval $[0, 2]$ yaitu

$$h = \frac{2-0}{10} = 0.2,$$

sehingga dipunyai titik-titik diskrit yang dirumuskan oleh

$$t_i = 0 + i\left(0.2\right) = \left(0.2\right) i, \qquad i = 0, 1, ..., 10,$$

yaitu

$$\begin{aligned} t_0 &= 0.0,\ t_1 = 0.2,\ t_2 = 0.4,\ t_3 = 0.6,\ t_4 = 0.8,\ t_5 = 1.0, \\ t_6 &= 1.2,\ t_7 = 1.4,\ t_8 = 1.6,\ t_9 = 1.8,\ t_{10} = 2.0. \end{aligned}$$

Karena diketahui bahwa

$$f\left(t, y\left(t\right)\right) = y\left(t\right) - t^2 + 1 \quad \text{dan} \quad y\left(0\right) = 0.5,$$

maka persamaan Euler dapat dinyatakan sebagai

$$\begin{aligned} y_0 &= 0.5, \\ y_{i+1} &\approx y_i + 0.2\left(y_i - t_i^2 + 1\right) \\ &= 1.2y_i - 0.2t_i^2 + 0.2, \qquad i = 0, 1, 2, ..., 9. \end{aligned}$$

Penyelesaian pada saat $i = 0$:

$$\begin{aligned} y_1 &= y(0.2) \approx 1.2y_0 - 0.2t_0^2 + 0.2 \\ &= 1.2(0.5) - 0.2\left(0^2\right) + 0.2 = 0.8, \end{aligned}$$

pada saat $i = 1$:

$$\begin{aligned} y_2 &= y(0.4) \approx 1.2y_1 - 0.2t_1^2 + 0.2 \\ &= 1.2(0.8) - 0.2\left(0.2^2\right) + 0.2 = 1.152, \end{aligned}$$

pada saat $i = 2$:

$$\begin{aligned} y_3 &= y(0.6) \approx 1.2y_2 - 0.2t_2^2 + 0.2 \\ &= 1.2(1.152) - 0.2\left(0.4^2\right) + 0.2 = 1.5504, \end{aligned}$$

dan seterusnya sampai pada $i = 9$:

$$\begin{aligned} y_{10} &= y(2) \approx 1.2y_9 - 0.2t_9^2 + 0.2 \\ &= 1.2(1.5504) - 0.2\left(0.8^2\right) + 0.2 = 4.8658, \end{aligned}$$

Berikut ini dibuat program MatLab `pdb_Euler` yang menggunakan metode Euler untuk menyelesaikan persamaan diferensial (9.1).

```
function yt = pdb_Euler(f,t,y0,n)
% pdb_Euler  Menyelesaikan PDB tingkat satu menggunakan metode Euler, dimana
%            y'(t) = f(t,y), t = [a,b], y(a) = y0.
% Input:  f = fungsi turunan pertama dari y, f=inline('(t,y)','t','y')
%         t = vektor yang memuat ujung-ujung interval dari t
%         y0 = syarat awal
%         n = lebar langkah/ banyak subinterval dalam interval t
% Output: yt = [t y(t)]

h = (t(2)-t(1))/n;
tvek(1) = t(1);
yvek(1) = y0;
for i = 2:n+1
    tvek(i) = tvek(i-1)+h;
    yvek(i) = yvek(i-1)+h*f(tvek(i-1),yvek(i-1)); % persamaan (9.2)
end
yt(:,1) = tvek'; yt(:,2) = yvek';
```

Dicatat bahwa metode Euler mempunyai galat yang sebanding dengan h. Terdapat dua modifikasi sederhana pada metode Euler yang memberikan keakuratan lebih baik. Dua modifikasi tersebut yaitu metode Heun dan metode titik tengah. Kedua metode ini mempunyai galat yang sebanding dengan h^2. Metode Heun dikenal juga sebagai metode prediktor-korektor.

9.2 Metode Heun

Metode Heun memperbaiki taksiran turunan pertama dengan mengambil rata-rata dari kedua turunan pada titik-titik ujung subinterval. Turunan di titik awal subinterval $[t_i, t_{i+1}]$ yaitu

$$y_i' = f(t_i, y_i).$$

Sementara itu, taksiran untuk y_{i+1} dihitung menggunakan metode Euler:

$$y_{i+1} \approx y_i + hf(t_i, y_i) \tag{9.3}$$

yang selanjutnya digunakan untuk menaksir turunan di titik akhir subinterval:

$$y_{i+1}' = f(t_{i+1}, y_{i+1}) \approx f(t_i + h, y_i + hf(t_i, y_i)).$$

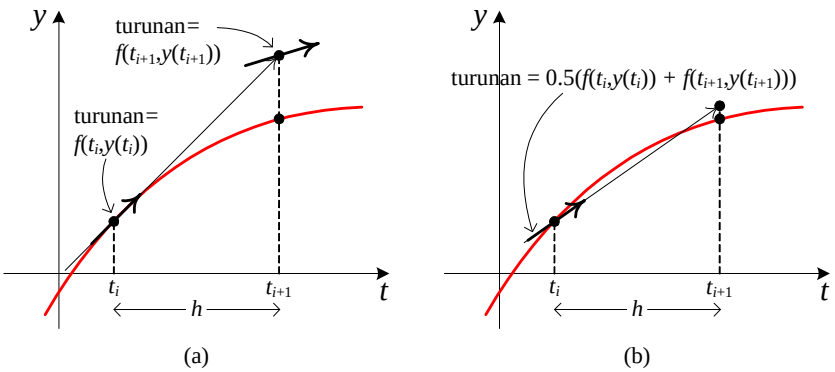
Karena itu, diperoleh rata-rata turunan pertama di $t = t_i$ yaitu

$$y_i' \approx \frac{f(t_i, y_i) + f(t_i + h, y_i + hf(t_i, y_i))}{2}. \tag{9.4}$$

Jadi, metode Heun diperoleh dengan mengganti $f(t_i, y_i)$ di persamaan (9.2) dengan ruas kanan dari persamaan (9.4):

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &\approx y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_i + h, y_i + hf(t_i, y_i))\right] \end{aligned}$$

dengan $i = 0, 1, 2, ..., n-1$. Di sini, persamaan (9.3) dinamakan prediktor dan persamaan (9.4) dinamakan korektor. Pengertian geometris untuk metode Heun diberikan dalam Gambar 9.2.



Gambar 9.2: Ilustrasi dari penurunan metode Heun.

Strategi dari metode Heun dinyatakan dalam Algoritma 7 dan diimplementasikan dalam fungsi MatLab `pdb_Heun()`.

Contoh 9.2 *Selesaikan persamaan diferensial*

$$y'(t) = y(t) - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

Algorithm 7 Algoritma Metode Heun

Masukan:

fungsi dari turunan pertama untuk $y(t)$: $f(t, y)$
interval dari t: $[t_0, t_n]$
syarat awal: $y_0 = \alpha$
banyak subinterval: n

Penghitungan Inti:

dihitung lebar langkah: $h = \dfrac{b-a}{n}$
untuk $i = 0 : n-1$, dihitung
$t_{i+1} = t_0 + (i+1) \cdot h$;
H1: $y'_i = f(t_i, y_i)$;
H2: $y_{i+1} = y_i + hy'_i$;
H3: $y'_{i+1} = f(t_{i+1}, y_{i+1})$;
H4: $y_{i+1} = y_i + \dfrac{h}{2}\left(y'_i + y'_{i+1}\right)$;

Hasil akhir: $y_1, y_2, ..., y_n$

menggunakan metode Heun dengan lebar langkah $h = 0.2$.

Penyelesaian. Dicatat bahwa

$$\begin{aligned} t_0 &= 0.0,\ t_1 = 0.2,\ t_2 = 0.4,\ t_3 = 0.6,\ t_4 = 0.8,\ t_5 = 1.0, \\ t_6 &= 1.2,\ t_7 = 1.4,\ t_8 = 1.6,\ t_9 = 1.8,\ t_{10} = 2.0. \end{aligned}$$

Strategi dari metode Heun dinyatakan oleh

$$\begin{aligned} y_0 &= 0.5, \\ \textbf{H1} &: y'_i = y_i - t_i^2 + 1 \\ \textbf{H2} &: y_{i+1} \approx y_i + 0.2y'_i \\ \textbf{H3} &: y'_{i+1} \approx y_{i+1} - t_{i+1}^2 + 1 \\ \textbf{H4} &: y_{i+1} \approx y_i + 0.1\left(y'_i + y'_{i+1}\right) \end{aligned}$$

dengan $i = 0, 1, 2, ..., 9$. Penyelesaian pada saat $i = 0$:

$$\begin{aligned} \textbf{H1} &: y'_0 = y'(0) = y_0 - 0^2 + 1 = 0.5 + 1 = 1.5 \\ \textbf{H2} &: y_1 = y(0.2) \approx y_0 + 0.2y'_0 = 0.5 + 0.2(1.5) = 0.8 \\ \textbf{H3} &: y'_1 = y'(0.2) \approx y_1 - 0.2^2 + 1 = 0.8 - 0.04 + 1 = 1.76 \\ \textbf{H4} &: y_1 = y(0.2) \approx y_0 + 0.1\left(y'_0 + y'_1\right) = 0.5 + 0.1(3.26) = 0.8260, \end{aligned}$$

pada saat $i = 1$:

$$\begin{aligned} \textbf{H1} &: y'_1 = y'(0.2) = y_1 - 0.2^2 + 1 = 0.8260 - 0.04 + 1 = 1.786 \\ \textbf{H2} &: y_2 = y(0.4) \approx y_1 + 0.2y'_1 = 0.8260 + 0.2(1.786) = 1.1832 \\ \textbf{H3} &: y'_2 = y'(0.4) \approx y_2 - 0.4^2 + 1 = 1.1832 - 0.16 + 1 = 2.0232 \\ \textbf{H4} &: y_2 = y(0.4) \approx y_1 + 0.1\left(y'_1 + y'_2\right) = 0.8260 + 0.1(3.8092) = 1.2069. \end{aligned}$$

pada saat $i = 2$:

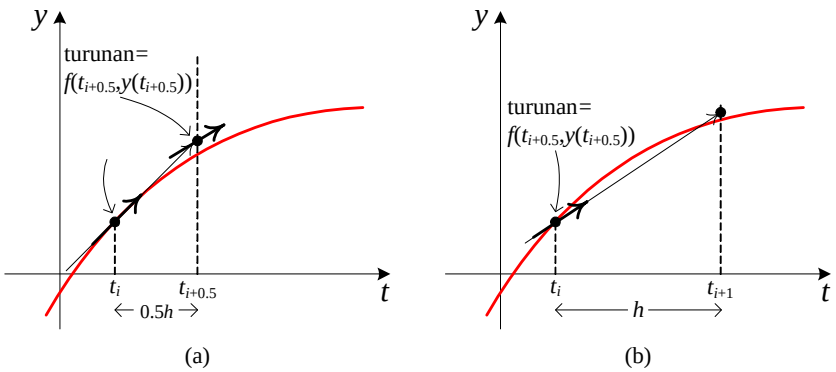
$$\begin{aligned}
\textbf{H1} \quad &: \quad y_2' = y'(0.4) = y_2 - 0.4^2 + 1 = 1.2069 - 0.16 + 1 = 2.0469 \\
\textbf{H2} \quad &: \quad y_3 = y(0.6) \approx y_2 + 0.2y_2' = 1.2069 + 0.2(2.0469) = 1.6163 \\
\textbf{H3} \quad &: \quad y'_3 = \ y'(0.6) \approx y_3 - 0.6^2 + 1 = 1.6163 - 0.36 + 1 = 2.2563 \\
\textbf{H4} \quad &: \quad y_3 = y(0.6) \approx y_2 + 0.1\left(y_2' + y_3'\right) = 1.2069 + 0.1(4.3032) = 1.6372,
\end{aligned}$$

dan seterusnya sampai pada $i = 9$ yang diserahkan kepada pembaca sebagai latihan. ▼

```
function yt = pdb_Heun(f,t,y0,n)
% pdb_Heun   Menyelesaikan PDB tingkat satu menggunakan metode Heun
%            (prediktor-korektor), dimana
%                    y'(t) = f(t,y), t = [a,b], y(a) = y0.
% Input:  f = fungsi turunan pertama dari y, f=inline('(t,y)','t','y')
%         t = vektor yang memuat ujung-ujung interval dari t
%         y0 = syarat awal
%         n = lebar langkah/ banyak subinterval dalam interval t
% Output: yt = [t y(t)]

h = (t(2)-t(1))/n;
tvek(1) = t(1);
yvek(1) = y0;
for i = 2:n+1
    tvek(i) = tvek(i-1)+h;
    fi_1 = f(tvek(i-1),yvek(i-1)); % H1
    yi = yvek(i-1)+h*f(tvek(i-1),yvek(i-1)); %H2
    fi = f(tvek(i),yi); % H3
    yvek(i) = yvek(i-1)+h/2*(fi_1+fi); %H4
end
yt(:,1) = tvek'; yt(:,2) = yvek';
```

9.3 Metode Titik Tengah



Gambar 9.3: Ilustrasi penurunan metode titik tengah.

Metode ini menggunakan metode Euler untuk memprediksi suatu nilai y di titik tengah

dari subinterval $[t_i, t_{i+1}]$:

$$y_{i+\frac{1}{2}} = y(t_i) + \frac{h}{2} f(t_i, y_i).$$

Nilai prediksi tersebut digunakan untuk menghitung turunan di titik tengah:

$$y'_{i+\frac{1}{2}} = f\left(t_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right).$$

Selanjutnya turunan tersebut digunakan untuk mengekstrapolasi secara linear dari t_i ke t_{i+1}:

$$y_{i+1} \approx y(t_i) + h\, y'_{i+\frac{1}{2}} = y_i + hf\left(t_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right).$$

Pengertian geometris untuk metode Heun diberikan dalam Gambar 9.3.

Strategi dari metode titik tengah dinyatakan dalam Algoritma 8 dan diimplementasikan dalam fungsi MatLab `pdb_tengah()`.

Algorithm 8 Algoritma Metode Titik Tengah

Masukan:

fungsi dari turunan pertama untuk $y(t)$: $f(t, y)$
interval dari t: $[t_0, t_n]$
syarat awal: $y_0 = \alpha$
banyak subinterval: n

Penghitungan Inti:

dihitung lebar langkah: $h = \dfrac{b-a}{n}$
untuk $i = 0 : n-1$, dihitung
$t_{i+\frac{1}{2}} = t_0 + \left(i + \frac{1}{2}\right) \cdot h$;
$t_{i+1} = t_0 + (i+1) \cdot h$;
T1: $y_{i+\frac{1}{2}} = y_i + \dfrac{h}{2} f(t_i, y_i)$;
T2: $y'_{i+\frac{1}{2}} = f\left(t_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right)$;
T3: $y_{i+1} \approx y_i + h y'_{i+\frac{1}{2}}$;

Hasil akhir: $y_1, y_2, ..., y_n$

Contoh 9.3 *Selesaikan persamaan diferensial*

$$y'(t) = y(t) - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

menggunakan metode titik tengah dengan lebar langkah $h = 0.2$.

Penyelesaian. Dicatat bahwa

$$\begin{aligned} t_0 &= 0.0,\ t_1 = 0.2,\ t_2 = 0.4,\ t_3 = 0.6,\ t_4 = 0.8,\ t_5 = 1.0, \\ t_6 &= 1.2,\ t_7 = 1.4,\ t_8 = 1.6,\ t_9 = 1.8,\ t_{10} = 2.0. \end{aligned}$$

Strategi dari metode titik tengah dinyatakan oleh

$$\begin{aligned} y_0 &= 0.5, \\ \mathbf{T1} &: \; y_{i+\frac{1}{2}} = y_i + 0.1\left(y_i - t_i^2 + 1\right) = 1.1y_i - 0.1t_i^2 + 0.1 \\ \mathbf{T2} &: \; y'_{i+\frac{1}{2}} = y_{i+\frac{1}{2}} - t_{i+\frac{1}{2}}^2 + 1 \\ \mathbf{T3} &: \; y_{i+1} \approx y_i + 0.2y'_{i+\frac{1}{2}} \end{aligned}$$

dengan $i = 0, 1, 2, ..., 9$. Penyelesaian pada saat $i = 0$:

$$\begin{aligned} \mathbf{T1} &: \; y_{\frac{1}{2}} = y\left(0.1\right) = 1.1y_0 - 0.1\left(0^2\right) + 0.1 = 1.1(0.5) + 0.1 = 0.65 \\ \mathbf{T2} &: \; y'_{\frac{1}{2}} = y'\left(0.1\right) = y_{\frac{1}{2}} - 0.1^2 + 1 = 0.65 - 0.01 + 1 = 1.64 \\ \mathbf{T3} &: \; y_1 = y\left(0.2\right) \approx y_0 + 0.2y'_{\frac{1}{2}} = 0.5 + 0.2\left(1.64\right) = 0.828 \end{aligned}$$

pada saat $i = 1$:

$$\begin{aligned} \mathbf{T1} &: \; y_{\frac{3}{2}} = y\left(0.3\right) = 1.1y_1 - 0.1\left(0.2^2\right) + 0.1 = 1.1\left(0.828\right) - 0.096 = 1.0068 \\ \mathbf{T2} &: \; y'_{\frac{3}{2}} = y'\left(0.3\right) = y_{\frac{3}{2}} - 0.3^2 + 1 = 1.0068 - 0.91 = 1.9168 \\ \mathbf{T3} &: \; y_2 = y\left(0.4\right) \approx y_1 + 0.2y'_{\frac{3}{2}} = 0.828 + 0.2\left(1.9168\right) = 1.2114, \end{aligned}$$

pada saat $i = 2$:

$$\begin{aligned} \mathbf{T1} &: \; y_{\frac{5}{2}} = y\left(0.5\right) = 1.1y_2 - 0.1\left(0.4^2\right) + 0.1 = 1.1\left(1.2114\right) - 0.084 = 1.4165 \\ \mathbf{T2} &: \; y'_{\frac{5}{2}} = y'\left(0.5\right) = y_{\frac{5}{2}} - 0.5^2 + 1 = 1.4165 + 0.75 = 2.1665 \\ \mathbf{T3} &: \; y_3 = y\left(0.6\right) \approx y_2 + 0.2y'_{\frac{5}{2}} = 1.2114 + 0.2\left(2.1665\right)1.6447, \end{aligned}$$

dan seterusnya sampai pada $i = 9$ yang diserahkan kepada pembaca sebagai latihan. ▼

```
function yt = pdb_tengah(f,t,y0,n)
% pdb_tengah   Menyelesaikan PDB tingkat satu menggunakan metode titik
%              tengah, dimana
%                   y'(t) = f(t,y), t = [a,b], y(a) = y0.
% Input:  f = fungsi turunan pertama dari y, f=inline('(t,y)','t','y')
%         t = vektor yang memuat ujung-ujung interval dari t
%         y0 = syarat awal
%         n = lebar langkah/ banyak subinterval dalam interval t
% Output: yt = [t y(t)]

h = (t(2)-t(1))/n;
tvek(1) = t(1);
yvek(1) = y0;
for i = 2:n+1
    tgh = tvek(i-1)+1/2*h;
    tvek(i) = tvek(i-1)+h;
    ymid = yvek(i-1)+h/2*f(tvek(i-1),yvek(i-1)); % T1
    fmid = f(tgh,ymid); %T2
    yvek(i) = yvek(i-1)+h*fmid; %T3
end
```

```
yt(:,1) = tvek'; yt(:,2) = yvek';
```

9.4 Metode Runge-Kutta

Metode Euler, Heun, dan titik tengah merupakan versi-versi dari suatu klas besar pendekatan satu langkah yang dinamakan metode Runge-Kutta. Metode Runge-Kutta mencapai keakuratan dari suatu pendekatan Taylor tanpa memerlukan turunan-turunan tingkat tinggi. Bentuk umumnya adalah

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + hF\left(t_i, y_i; h\right) \end{aligned}$$

dengan $i = 0, 1, 2, ...$ Fungsi F dinamakan fungsi kenaikan yang dapat dituliskan dalam bentuk

$$F = a_1 k_1 + a_2 k_2 + \cdots + a_m k_m$$

dimana m dinamakan orde dari metode Runge-Kutta, a_i adalah konstanta dan

$$\begin{aligned} k_1 &= f\left(t_i, y_i\right), \\ k_2 &= f\left(t_i + p_1 h, y_i + q_{11} k_1 h\right), \\ k_3 &= f\left(t_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h\right), \\ &\vdots \\ k_m &= f\left(t_i + p_{m-1} h, y_i + q_{m-1,1} k_1 h + q_{m-1,2} k_2 h + \cdots + q_{m-1,m-1} k_{m-1} h\right). \end{aligned}$$

Nilai-nilai a_i, p_i, dan q_{ij} untuk suatu orde m dicari dengan cara menyamakan persamaan tersebut dengan ekspansi deret Taylor orde m.

Metode Runge-Kutta orde 1 tidak lain adalah metode Euler. Serupa dengan itu, metode Heun adalah suatu contoh dari metode Runge-Kutta orde 2. Metode Runge-Kutta orde 4 adalah satu dari metode yang banyak digunakan untuk menyelesaikan persamaan diferensial. Metode ini mempunyai suatu galat pemotongan yang sebanding dengan h^4. Algoritmanya digambarkan seperti berikut ini dan diimplementasikan dalam fungsi MatLab `pdb_RK4`.

$$y_{i+1} = y_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad i = 0, 1, ..., n-1$$

dimana

$$\begin{aligned} k_1 &= f\left(t_i, y_i\right), \\ k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1\right), \\ k_3 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_2\right), \\ k_4 &= f\left(t_i + h, y_i + h k_3\right). \end{aligned}$$

Contoh 9.4 *Selesaikan persamaan diferensial*

$$y'(t) = y(t) - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5$$

menggunakan metode Runge-Kutta orde 4 *dengan lebar langkah* $h = 0.2$.

Penyelesaian. Dinyatakan bahwa

$$f(t, y(t)) = y(t) - t^2 + 1,$$

dan

$$\begin{aligned} t_0 &= 0.0,\ t_1 = 0.2,\ t_2 = 0.4,\ t_3 = 0.6,\ t_4 = 0.8,\ t_5 = 1.0, \\ t_6 &= 1.2,\ t_7 = 1.4,\ t_8 = 1.6,\ t_9 = 1.8,\ t_{10} = 2.0. \end{aligned}$$

Untuk menghitung $y_1 = y(0.2)$, dimana $i = 0$, tahap-tahap penghitungannya dimulai dari menghitung k_1 sampai k_5:

$$\begin{aligned} k_1 &= f(t_0, y_0) = f(0, 0.5) = 0.5 - 0^2 + 1 = 1.5, \\ k_2 &= f(t_0 + 0.1, y_0 + 0.1k_1) = f(0.1, 0.65) = 0.65 - 0.1^2 + 1 = 1.64, \\ k_3 &= f(t_0 + 0.1, y_0 + 0.1k_2) = f(0.1, 0.664) = 0.664 - 0.1^2 + 1 = 1.654, \\ k_4 &= f(t_0 + 0.2, y_0 + 0.2k_3) = f(0.2, 0.8308) = 0.8308 - 0.2^2 + 1 = 1.7908, \end{aligned}$$

sehingga akhirnya diperoleh

$$y_1 = y(0.2) = y_0 + \frac{0.2}{6}(1.5 + 2 \cdot 1.64 + 2 \cdot 1.654 + 1.7908) = 0.8293.$$

Dengan cara yang sama, $y(0.4)$, ..., $y(2)$ dapat dihitung dan diserahkan kepada pembaca sebagai latihan. ▼

Perbandingan hasil dari metode Runge-Kutta orde 4 dengan metode Euler, metode Heun, dan metode titik tengah untuk contoh di atas diberikan dalam Tabel 9.1. Dicatat bahwa penyelesaian eksak dari persamaan diferensial adalah

$$y(t) = (t+1)^2 - \frac{1}{2}e^t$$

Dari tabel tersebut ditunjukkan bahwa metode Runge-Kutta orde 4 adalah yang terbaik, sedangkan metode Euler adalah yang paling buruk. Selain itu, dari tabel juga dapat dilihat bahwa metode titik tengah lebih baik daripada metode Heun.

```
function yt = pdb_RK4(f,t,y0,n)
% pdb_RK4   Menyelesaikan PDB tingkat satu menggunakan metode Runge-Kutta
%           orde 4, dimana
%                   y'(t) = f(t,y), t = [a,b], y(a) = y0.
% Input:  f = fungsi turunan pertama dari y, f=inline('(t,y)','t','y')
%         t = vektor yang memuat ujung-ujung interval dari t
%         y0 = syarat awal
%         n = lebar langkah/ banyak subinterval dalam interval t
% Output: yt = [t y(t)]

h = (t(2)-t(1))/n;
```

Tabel 9.1: Perbandingan hasil dari metode-metode penyelesaian untuk persamaan diferensial $y'(t) = y(t) - t^2 + 1$, $0 \leq t \leq 2$, $y(0) = 0.5$.

t_i	y_i				
	Euler	Heun	Titik Tengah	Runge-Kutta orde 4	Eksak
0.0	0.5000	0.5000	0.5000	0.5000	0.5000
0.2	0.8000	0.8260	0.8280	0.8293	0.8293
0.4	1.1520	1.2069	1.2114	1.2141	1.2141
0.6	1.5504	1.6372	1.6447	1.6489	1.6489
0.8	1.9885	2.1102	2.1213	2.1272	2.1272
1.0	2.4582	2.6177	2.6332	2.6408	2.6409
1.2	2.9498	3.1496	3.1705	3.1799	3.1799
1.4	3.4518	3.6937	3.7212	3.7323	3.7324
1.6	3.9501	4.2351	4.2706	4.2834	4.2835
1.8	4.4282	4.7556	4.8010	4.8151	4.8152
2.0	4.8658	5.2331	5.2904	5.3054	5.3055

```
tvek(1) = t(1);
yvek(1) = y0;
for i=2:n+1
    tvek(i) = tvek(i-1)+h;
    k1 = f(tvek(i-1),yvek(i-1));
    k2 = f(tvek(i-1)+0.5*h,yvek(i-1)+0.5*k1*h);
    k3 = f(tvek(i-1)+0.5*h,yvek(i-1)+0.5*k2*h);
    k4 = f(tvek(i),yvek(i-1)+k3*h);
    yvek(i) = yvek(i-1)+(k1+2*k2+2*k3+k4)*h/6;
end
yt(:,1) = tvek'; yt(:,2) = yvek';
```

Bab 10

Integrasi Numerik

Tujuan Pembelajaran:
- Mendapatkan rumus hampiran untuk menghitung integral dari suatu fungsi atas interval terbatas.
- Mengetahui aturan yang memberikan hasil lebih dekat dengan hasil sebenarnya.

Pada bab ini kita tertarik untuk menghitung integral

$$I = \int_a^b f(x)\ dx$$

untuk $f(x) \in \mathbb{R}$ (dan tentu saja $x \in \mathbb{R}$). Bentuk umum integrasi numerik dari suatu fungsi $f(x)$ atas suatu interval $[a, b]$ adalah suatu jumlahan berbobot dari nilai-nilai fungsi di berhingga $(n+1)$ titik-titik diskrit, dan dinamakan kuadratur (*quadrature*):

$$\int_a^b f(x)\ dx = \sum_{i=0}^{n} w_i \cdot f(x_i)$$

dengan $a = x_0 < x_1 < \cdots < x_n = b$. Dalam kedua subbab berikut ini akan diturunkan empat bentuk persamaan integral numerik yang dikenal sebagai rumus eksplisit Newton-Cotes. Rumus ini ini didasarkan pada strategi penggantian fungsi $f(x)$ dengan suatu hampiran fungsi $f_m(x)$ yang lebih mudah diintegralkan. Fungsi $f_m(x)$ tersebut adalah suatu polinomial derajat m.

10.1 Aturan Trapesium

Aturan ini didasarkan pada interpolasi linear dari fungsi $f(x)$ pada $[x_{i-1}, x_i]$ untuk setiap $i = 1, 2, ..., n$. Jadi, pada setiap subinterval, hampiran untuk $\int_{x_{i-1}}^{x_i} f(x)\ dx$ diberikan oleh luas bidang datar trapesium, lihat Gambar 10.1:

$$\int_{x_{i-1}}^{x_i} f(x)\ dx \approx \frac{1}{2}\left[f(x_i) + f(x_{i-1})\right](x_i - x_{i-1}) = \frac{h}{2}\left[f(x_{i-1}) + f(x_i)\right]. \qquad (10.1)$$

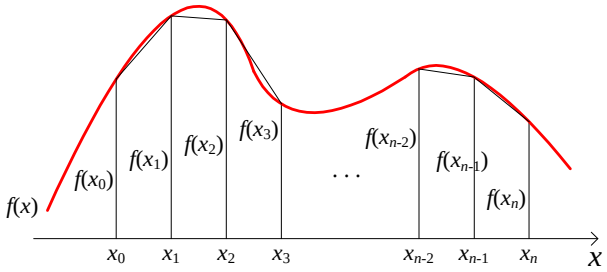
Dicatat bahwa $h = x_i - x_{i-1} = \dfrac{b-a}{n}$. Diaplikasikan (10.1) untuk $i = 1$ sampai $i = n$, dipunyai

$$\int_a^b f(x)\ dx \approx T(n) = \frac{h}{2}\sum_{i=1}^{n}\left[f(x_{i-1}) + f(x_i)\right],$$

dan jika jumlahan diekspansikan maka diperoleh

$$\begin{aligned} T(n) &= \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots 2f(x_{n-1}) + f(x_n)\right] \\ &= \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]. \end{aligned} \tag{10.2}$$

Dicatat bahwa aturan trapesium mempunyai galat pemotongan yang sebanding dengan h^3.



Gambar 10.1: Ilustrasi dari aturan trapesium.

Contoh 10.1 *Hitung luas bidang datar yang dibatasi oleh* $y = 2x^3$ *dan sumbu* x *untuk* $x \in [0, 1]$ *dengan menggunakan aturan trapesium dan lebar langkah* $h = 0.1$.

Penyelesaian. Diperoleh tabel

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$f(x)$	0	0.002	0.016	0.054	0.128	0.25	0.432	0.686	1.024	1.458	2

Jadi,

$$\begin{aligned} T(10) &= \int_0^1 2x^3\, dx \\ &\approx \frac{0.1}{2}\left[0 + 2\left(0.002 + 0.016 + \cdots + 1.024 + 1.458\right) + 2\right] = 0.505. \end{aligned}$$

Secara kalkulus,

$$L = \int_0^1 2x^3\, dx = \left[\frac{1}{2}x^4\right]_0^1 = 0.5.$$

Dengan $h = 0.1$ ternyata terjadi kesalahan $e = |0.505 - 0.5| = 0.005$.

Rumus integrasi menggunakan aturan trapesium diimplementasikan dalam fungsi MatLab `trapesium()`.

```
function Tn = trapesium(f,x,n)
% trapesium   Menyelesaikan integral f(x) atas interval [a,b] menggunakan
%             aturan trapesium dengan n subinterval
% Input:  f = fungsi yang diintegralkan
%         x = vektor [a b]
%         n = lebar langkah/ banyak subinterval dalam interval x
% Output: Tn = luas bidang datar yang dibatasi oleh f(x) dan sumbu x dalam
```

```
%              interval [a,b]
h = (x(2)-x(1))/n;
xvek = x(1):h:x(2);
yvek = f(xvek); % nilai f untuk semua titik diskrit
Tn = h/2*(yvek(1)+sum(yvek(2:n))+yvek(n+1)); % persamaan(10.2)
```

10.2 Aturan-aturan Simpson

Taksiran yang lebih akurat dari suatu integral diperoleh jika polinomial derajat tinggi digunakan untuk menghubungkan titik-titik diskrit. Rumus-rumus yang dihasilkan dengan pengambilan integral dari polinomial tersebut dinamakan aturan-aturan Simpson.

Pertama kali akan digunakan suatu polinomial derajat dua untuk mencocokkan titik-titik $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, dan $(x_{i+1}, f(x_{i+1}))$. Didefinisikan polinomial derajat dua

$$P_{i-1}(x) = a(x - x_i)^2 + b(x - x_i) + c. \tag{10.3}$$

Dicatat bahwa subskrip $(i-1)$ di sini tidak menyatakan derajat dari polinomial, tetapi menyatakan titik ujung kiri dari interval $[x_{i-1}, x_{i+1}]$ dimana kita mencocokkan polinomial derajat dua. Untuk kesepakatan diperkenalkan singkatan $y_i = f(x_i)$. Oleh karena itu, dari (10.3) kita membentuk tiga persamaan dalam a, b, c:

$$\begin{aligned} a(x_{i-1} - x_i)^2 + b(x_{i-1} - x_i) + c &= y_{i-1}, \\ a(x_i - x_i)^2 + b(x_i - x_i) + c &= y_i, \\ a(x_{i+1} - x_i)^2 + b(x_{i+1} - x_i) + c &= y_{i+1}. \end{aligned}$$

Ini adalah suatu sistem persamaan linear, dan kita akan mengasumsikan bahwa $h = x_i - x_{i-1} = x_{i+1} - x_i$, sehingga

$$\begin{aligned} a &= \frac{y_{i+1} - 2y_i + y_{i-1}}{2h^2}, \\ b &= \frac{y_{i+1} - y_{i-1}}{2h}, \\ c &= y_i. \end{aligned}$$

Diintegralkan polinomial derajat dua (10.3) dari $x = x_{i-1}$ sampai $x = x_{i+1}$ dengan koefisien-koefisien tersebut menghasilkan hampiran

$$\begin{aligned} \int_{x_{i-1}}^{x_{i+1}} f(x)\ dx &\approx \int_{x_{i-1}}^{x_{i+1}} P_{i-1}(x)\ dx = \int_{x_{i-1}}^{x_{i+1}} \left[a(x - x_i)^2 + b(x - x_i) + c \right] dx \\ &= \left[\frac{1}{3} a(x - x_i)^3 + \frac{1}{2} b(x - x_i)^2 + cx \right]_{x_{i-1}}^{x_{i+1}} = \frac{2}{3} ah^3 + 2ch \\ &= \frac{2h}{3} \left(\frac{y_{i+1} - 2y_i + y_{i-1}}{2} + 3y_i \right) = \frac{h}{3} (y_{i-1} + 4y_i + y_{i+1}). \end{aligned}$$

Seperti dalam bagian sebelumnya, kita ingin mengintegralkan $f(x)$ pada $[a, b]$. Jadi, seperti sebelumnya, $a = x_0$ dan $b = x_n$. Jika n adalah suatu bilangan genap, maka banyaknya subinterval dalam $[a, b]$ adalah suatu bilangan genap, dan karena itu kita

mempunyai hampiran

$$\begin{aligned} I &= \int_a^b f(x)\ dx \\ &\approx \int_{x_0}^{x_2} P_0(x)\ dx + \int_{x_2}^{x_4} P_2(x)\ dx + \cdots + \int_{x_{n-2}}^{x_n} P_{n-2}(x)\ dx \\ &= \frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 2y_{n-2} + 4y_{n-1} + y_n\right) \end{aligned}$$

Dengan menggunakan notasi jumlahan, dari persamaan terakhir kita memperoleh aturan $\frac{1}{3}$-Simpson:

$$S_3(n) = \frac{h}{3}\left(y_0 + 4\sum_{i=1}^{\frac{1}{2}n} y_{2i-1} + 2\sum_{i=1}^{\frac{1}{2}n-1} y_{2i} + y_n\right). \tag{10.4}$$

Dicatat bahwa aturan $\frac{1}{3}$-Simpson mempunyai galat pemotongan yang sebanding dengan h^5.

Rumus integrasi menggunakan aturan $\frac{1}{3}$-Simpson diimplementasikan dalam fungsi MatLab `Simpson3()`.

```
function S3n = Simpson3(f,x,n)
% Simpson3   Menyelesaikan integral f(x) atas interval [a,b] menggunakan
%            aturan 1/3-Simpson dengan n subinterval
% Input:  f = fungsi yang diintegralkan
%         x = vektor [a b]
%         n = lebar langkah/ banyak subinterval dalam interval x
% Output: S3n = luas bidang datar yang dibatasi oleh f(x) dan sumbu x dalam
%               interval [a,b]
h = (x(2)-x(1))/n;
xvek = x(1):h:x(2);
yvek = f(xvek); % nilai f untuk semua titik diskrit
S3n = h/3*(yvek(1)+4*sum(yvek(2:2:n))+2*sum(yvek(3:2:n-1))+yvek(n+1)); % persamaan(10.4)
```

Selanjutnya, jika digunakan suatu polinomial derajat tiga untuk mencocokkan titik-titik $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$, dan $(x_{i+2}, f(x_{i+2}))$, maka dengan cara yang sama seperti sebelumnya akan diperoleh hampiran

$$I = \int_{x_{i-1}}^{x_{i+2}} f(x)\ dx \approx \frac{3h}{8}\left(y_{i-1} + 3y_i + 3y_{i+1} + y_{i+2}\right).$$

Jika n adalah suatu bilangan kelipatan tiga, maka kita mempunyai hampiran

$$\begin{aligned} I &= \int_a^b f(x)\ dx \\ &\approx \int_{x_0}^{x_3} P_0(x)\ dx + \int_{x_3}^{x_6} P_3(x)\ dx + \cdots + \int_{x_{n-3}}^{x_n} P_{n-3}(x)\ dx \\ &= \frac{3h}{8}\left(y_0 + 3y_1 + 3y_2 + 2y_3 + 3y_4 + 3y_5 + 2y_6 + \cdots + 2y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n\right). \end{aligned}$$

Dengan menggunakan notasi jumlahan, dari persamaan terakhir kita memperoleh aturan $\frac{3}{8}$-Simpson:

$$S_4(n) = \frac{3h}{8}\left(y_0 + 3\sum_{i=1}^{\frac{1}{3}n}(y_{3i-2} + y_{3i-1}) + 2\sum_{i=1}^{\frac{1}{3}n-1} y_{3i} + y_n\right). \tag{10.5}$$

Dicatat bahwa aturan $\frac{3}{8}$-Simpson juga mempunyai galat pemotongan yang sebanding dengan h^5.

Rumus integrasi menggunakan aturan $\frac{3}{8}$-Simpson diimplementasikan dalam fungsi MatLab `Simpson4()`.

```
function S4n = Simpson4(f,x,n)
% Simpson4   Menyelesaikan integral f(x) atas interval [a,b] menggunakan
%            aturan 3/8-Simpson dengan n subinterval
% Input:  f = fungsi yang diintegralkan
%         x = vektor [a b]
%         n = lebar langkah/ banyak subinterval dalam interval x
% Output: S4n = luas bidang datar yang dibatasi oleh f(x) dan sumbu x dalam
%               interval [a,b]
h = (x(2)-x(1))/n;
xvek = x(1):h:x(2);
yvek = f(xvek); % nilai f untuk semua titik diskrit
sum1 = sum(yvek(2:3:n-1));
sum2 = sum(yvek(3:3:n));
sum3 = sum(yvek(4:3:n-2));
S4n = 3*h/8*(yvek(1)+3*(sum1+sum2)+2*sum3+yvek(n+1)); % persamaan(10.5)
```

Terakhir, jika digunakan suatu polinomial derajat empat untuk mencocokkan titik-titik $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$, $(x_{i+2}, f(x_{i+2}))$, dan $(x_{i+3}, f(x_{i+3}))$, maka dengan cara yang sama seperti sebelumnya akan diperoleh hampiran

$$I = \int_{x_{i-1}}^{x_{i+3}} f(x)\ dx \approx \frac{2h}{45}\left(7y_{i-1} + 32y_i + 12y_{i+1} + 32y_{i+2} + 7y_{i+3}\right).$$

Jika n adalah suatu bilangan kelipatan empat, maka kita mempunyai hampiran

$$\begin{aligned} I &= \int_a^b f(x)\ dx \\ &\approx \int_{x_0}^{x_4} P_0(x)\ dx + \int_{x_4}^{x_8} P_4(x)\ dx + \cdots + \int_{x_{n-4}}^{x_n} P_{n-4}(x)\ dx \\ &= \frac{2h}{45}\left(7y_0 + 32y_1 + 12y_2 + 32y_3 + 14y_4 + 32y_5 + 12y_6 + 32y_7 + 14y_8 + \cdots\right. \\ &\quad \left. 14y_{n-4} + 32y_{n-3} + 12y_{n-2} + 32y_{n-1} + 7y_n\right). \end{aligned}$$

Dengan menggunakan notasi jumlahan, dari persamaan terakhir kita memperoleh aturan $\frac{2}{45}$-Simpson:

$$S_4(n) = \frac{2h}{45}\left(7y_0 + \sum_{i=1}^{\frac{1}{4}n}(32y_{4i-3} + 12y_{4i-2} + 32y_{4i-1}) + 14\sum_{i=1}^{\frac{1}{4}n-1} y_{4i} + 7y_n\right). \tag{10.6}$$

Dicatat bahwa aturan $\frac{2}{45}$-Simpson mempunyai galat pemotongan yang sebanding dengan h^7.

Rumus integrasi menggunakan aturan $\frac{2}{45}$-Simpson diimplementasikan dalam fungsi MatLab `Simpson5()`.

```
function S5n = Simpson5(f,x,n)
% Simpson5   Menyelesaikan integral f(x) atas interval [a,b] menggunakan
%            aturan 2/45-Simpson dengan n subinterval
% Input:  f = fungsi yang diintegralkan
%         x = vektor [a b]
%         n = lebar langkah/ banyak subinterval dalam interval x
% Output: S4n = luas bidang datar yang dibatasi oleh f(x) dan sumbu x dalam
%               interval [a,b]
h = (x(2)-x(1))/n;
xvek = x(1):h:x(2);
yvek = f(xvek); % nilai f untuk semua titik diskrit
sum1 = sum(yvek(2:4:n-2));
sum2 = sum(yvek(3:4:n-1));
sum3 = sum(yvek(4:3:n));
sum4 = sum(yvek(5:3:n-3));
S5n = 2*h/45*(7*yvek(1)+32*sum1+12*sum2+32*sum3+14*sum4+7*yvek(n+1)); % pers.(10.6)
```

DAFTAR PUSTAKA

[1] Pav, S.E. (2005). *Numerical Methods Course Notes*. Naskah, Department of Mathematics, University of California.

[2] Steven, C.C. and Raymond, P.C. (1991). *Metode Numerik untuk Teknik: dengan Penerapan pada Komputer Pribadi*, penerjemah: S. Sardy dan pendamping: Lamyarni I.S., Universitas Indonesia.

[3] Subakti, I. (2003). *Metode Numerik*. Naskah, Jurusan Teknik Informatika ITS.

[4] Suparno, S. (2008). *Komputasi untuk Sains dan Teknik*. Naskah, Departemen Fisika-FMIPA UI.

[5] Yang, W.Y., et al. (2005). *Applied Numerical Methods using MATLAB*. John Wiley & Sons, Inc.

[6] Zarowski, C.J. (2004). *An Introduction to Numerical Analysis for Electrical and Computer Engineers*. John Wiley & Sons, Inc.