

# **Komputasi Numerik**

**Konsep & Aplikasi menggunakan Python**

Mulaab Mulyo

Copyright © 2021 Mulaab Mulyo

PROYEK UNIVERSITAS TRUNOJOYO MADURA

Buku pegangan matakuliah Teknik Informatika.

*Cetakan Pertama 30 Maret 2021*



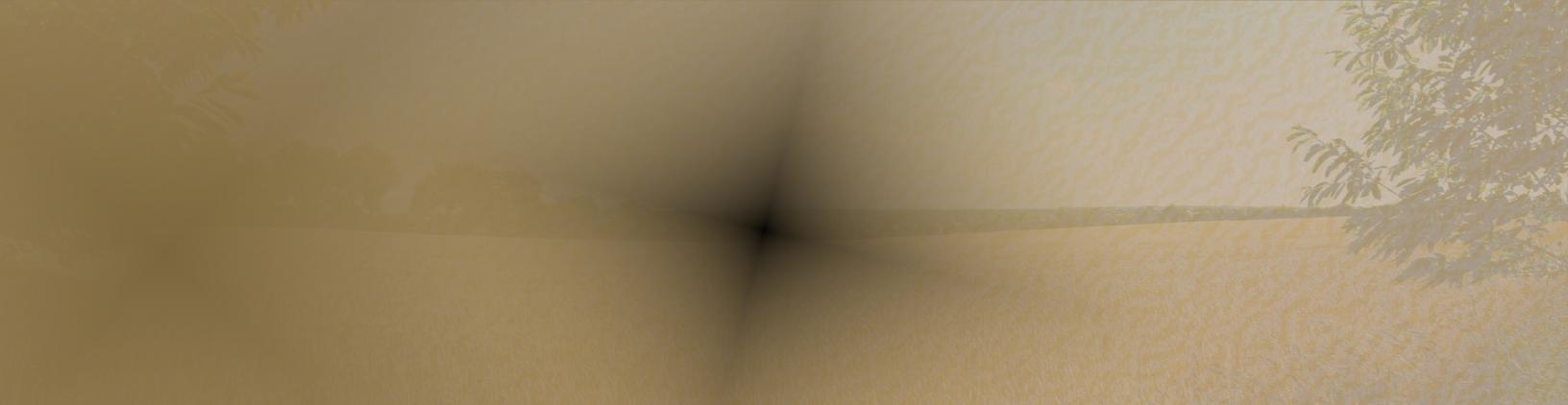
## Daftar Isi

1	01-Pendekatan dan Error Pembulatan .....	5
2	02-Penyelesaian persamaan linier .....	6
3	03-Penyelesaian Persamaan non Linier .....	7
4	04-Regresi Least Square .....	8
5	05-Interpolasi .....	9
5.1	Interpolasi Polynomial	9
5.2	Interpolasi Polynomial: Bentuk Lagrange	10
5.3	Interpolasi Polynomial: Bentuk Newton	13
5.4	Nested Form	14
5.5	Menghitung Koefisien $a_i$ menggunakan selisih terbagi (Divided Differences)	15
5.6	Algoaritma dan Pseudocode menghitung Selisih Terbagi	18
5.7	Interpolasi Polynomial dengan Algoritma Neville	19
6	06-Integrasi Numerik .....	23
7	07-Turunan Numerik .....	24

<b>8</b>	<b>08-Penyelesaian Persamaan Diferensial Biasa .....</b>	<b>25</b>
<b>9</b>	<b>09-Penyelesaian Persamaan Diferensial Parsial .....</b>	<b>26</b>
<b>10</b>	<b>10-Turunan Numerik .....</b>	<b>27</b>
<b>11</b>	<b>11-Metode Monte Carlo dan Simulasi .....</b>	<b>28</b>



## **1. 01-Pendekatan dan Error Pembulatan**



## **2. 02-Penyelesaian persamaan linier**



### **3. 03-Penyelesaian Persamaan non Linier**



## **4. 04-Regresi Least Square**

## 5. 05-Interpolasi

### 5.1 Interpolasi Polynomial

Kita mulai dengan titik-titik data pada tabel sebagai berikut

x	$x_0$	$x_1$	...	$x_n$
y	$y_0$	$x_2$	...	$y_n$

Dan asumsikan bahwa terdapat pasangan titik  $(x_i, y_i)$  terdiri dari  $n + 1$  titik titik data. Dan titik-titik tersebut dapat dinyatakan dalam  $n + 1$  titik dalam bidang Cartesian, dan kita ingin mencari kurva polinomial yang melalui semua titik tersebut. Sehingga, kita akan menentukan polinomial yang didefinisikan untuk seluruh titik  $x$  dan yang bersesuaian dengan nilai nilai dari  $y_i$  untuk setiap titik  $x_i$  dari  $n + 1$  titik dalam tabel diatas. Pernyataan ini dapat dinyatakan dalam bentuk

$$p(x_i) \cong \varphi_n(x_i) \text{ untuk semua } i = 0, 1, 2, 3, \dots, n \quad (5.1)$$

sehingga  $\varphi_n(x) = a_0 + a_1x + \dots + a_nx^n$ . Kemudian dari persamaan 5.1 kita dapatkan

$$a_0 + a_1x_i + \dots + a_nx_i^n = p(x_i) \text{ untuk semua } i = 0, 1, 2, 3, \dots, n \quad (5.2)$$

Ini adalah sekumpulan dari  $(n + 1)$  persamaan linier dalam  $(n + 1)$  dari  $a_0, a_1, \dots, a_n$ . Determinan koefisien dari sistem persamaan 5.2 adalah

$$\begin{vmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & x_n & \dots & x_n^n \end{vmatrix} = \prod_{\substack{0 \leq i, j \leq n \\ i \neq j}} (x_i - x_j) \neq 0$$

Determinan ini dikenal dengan *Vandermonde's determinant*. Nilai dari determinannya tidak sama dengan nol karena  $x_0, x_1, x_2, \dots, x_n$  tidak sama. Oleh karena itu dengan aturan Crammer, nilai  $a_0, a_1, a_2, \dots, a_n$  dapat diketahui. Sehingga polinomial  $\varphi_n(x)$  ada dan unik nilainya. Polinomial  $\varphi_n(x)$  disebut dengan polinomial interpolasi (interpolation polynomial). Dan titik titik  $x_0, x_1, x_2, \dots, x_n$  disebut dengan titik titik interpolasi (interpolating point) atau simpul (nodes)

Perhatikan pada kasus sederhana untuk  $n = 0$ . Maka fungsi yang dihasilkan adalah fungsi konstanta. Dengan kata lain, polinomial  $p$  dengan derajat 0 dinyatakan dengan  $p(x) = y_0$  menghasilkan tabel dengan satu simpul (one-node table). Untuk kasus  $n = 1$ , maka akan menghasilkan garis lurus atau fungsi linier. Sehingga polinomialnya  $p$  dapat didefinisikan dengan

$$\begin{aligned} p(x) &= \left( \frac{x - x_1}{x_0 - x_1} \right) y_0 + \left( \frac{x - x_0}{x_1 - x_0} \right) y_1 \\ &= y_0 + \left( \frac{y_1 - y_0}{x_1 - x_0} \right) (x - x_0) \end{aligned} \quad (5.3)$$

adalah derajat satu dan menghasilkan tabel seperti diatas. Itu artinya bahwa dalam kasus ini  $p(x_0) = y_0$  dan  $p(x_1) = y_1$  sehingga dengan mudah dapat diverifikasi. Disini  $p$  digunakan untuk interpolasi linier adalah derajat pertama dan menghasilkan tabel diatas.

- **Contoh 5.1** Tentukan polinomial dari interpolasi derajat terendah dari data pada tabel berikut:

x	1.4	1.25
y	3.7	3.9

**Penyelesaian** Dengan persamaan diatas polinomial tersebut dapat dicari dengan

$$\begin{aligned} p(x) &= \left( \frac{x - 1.25}{1.4 - 1.25} \right) 3.7 + \left( \frac{x - 1.4}{1.25 - 1.4} \right) 3.9 \\ &= 3.7 + \left( \frac{3.9 - 3.7}{1.25 - 1.4} \right) (x - 1.4) \\ &= 3.7 - \frac{4}{3}(x - 1.4) \end{aligned}$$

## 5.2 Interpolasi Polynomial: Bentuk Lagrange

Seandainya kita ingin menjadikan interpolasi dari fungsi sembarang pada simpul-simpul berikut  $x_0, x_1, \dots, x_n$ . Kita pertama mendefinisikan sistem  $n+1$  polinomial tertentu derajat  $n$  yang disebut dengan **cardinal polinomial** dalam teori interpolasi. Disini dinotasikan dengan  $\ell_0, \ell_1, \dots, \ell_n$  dan memiliki sifat

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Selanjutnya, kita dapat melakukan interpolasi fungsi sembarang  $f$  dengan bentuk Lagrange dari interpolasi polinomial

$$p_n(x) = \sum_{i=0}^n \ell_i(x) f(x_i) \quad (5.4)$$

Fungsi  $p_n$  yang merupakan kombinasi linier dari polinomial  $\ell_i$  dengan derajat  $n$ . Oleh karena itu, kita akan menghitung  $p_n$  pada  $x_j$ , yang akan menghasilkan  $f(x_j)$ :

$$p_n(x_j) = \sum_{i=0}^n \ell_i(x_j) f(x_i) = \ell_j(x_j) f(x_j) = f(x_j)$$

$p_n$  adalah interpolasi polinomial untuk fungsi  $f$  pada simpul-simpul  $x_0, x_1, \dots, x_n$ . Sehingga rumus untuk cardinal polynomial  $\ell_i$  yaitu

$$\ell_i(x) = \prod_{\substack{j \neq i \\ j=0}}^n \left( \frac{x - x_j}{x_i - x_j} \right) \quad (0 \leq i \leq n) \quad (5.5)$$

Dari rumus diatas menyatakan bahwa  $\ell_i(x)$  merupakan perkalian dari  $n$  faktor linier:

$$\ell_i(x) = \left( \frac{x - x_0}{x_i - x_0} \right) \left( \frac{x - x_1}{x_i - x_1} \right) \dots \left( \frac{x - x_{i-1}}{x_i - x_{i-1}} \right) \left( \frac{x - x_{i+1}}{x_i - x_{i+1}} \right) \dots \left( \frac{x - x_n}{x_i - x_n} \right)$$

Misalkan untuk  $n = 2$

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \quad l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \quad l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Dari persamaan tersebut tampak bahwa penyebutnya menghasilkan suatu nilai tertentu dan variabel  $x$  hanya ada pada pembilang. Sehingga  $\ell_i$  adalah polinomial derajat  $n$ . Perhatikan bahwa ketika  $\ell_i(x)$  dihitung pada  $x = x_i$ , setiap faktor dalam persamaan sebelumnya menjadi 1. Sehingga  $\ell_i(x_i) = 1$ . Tetapi ketika  $\ell_i(x)$  dihitung pada simpul lain, katakan  $x_j$  salah satu dari faktor dalam persamaan diatas menjadi 0 dan  $\ell_i(x_j) = 0$  untuk  $i \neq j$

Pada gambar 5.1 menunjukkan beberapa *cardinal polynomial Lagrange*  $\ell_0, \ell_1, \ell_2, \ell_3, \ell_4$ , dan  $\ell_5$

■ **Contoh 5.2** Tentukan *cardinal polinomial* yang sesuai untuk interpolasi tabel berikut dan nyatakan bentuk interpolasi polinomial Lagrange

$x$	$\frac{1}{3}$	$\frac{1}{4}$	1
$y = f(x)$	2	-1	7

**Penyelesaian** Dengan menggunakan persamaan 5.5 diperoleh

$$\begin{aligned}\ell_0(x) &= \frac{(x-\frac{1}{4})(x-1)}{(\frac{1}{3}-\frac{1}{4})(\frac{1}{3}-1)} = -18(x-\frac{1}{4})(x-1) \\ \ell_1(x) &= \frac{(x-\frac{1}{3})(x-1)}{(\frac{1}{4}-\frac{1}{3})(\frac{1}{4}-1)} = 16(x-\frac{1}{3})(x-1) \\ \ell_2(x) &= \frac{(x-\frac{1}{3})(x-\frac{1}{4})}{(1-\frac{1}{3})(1-\frac{1}{4})} = 2(x-\frac{1}{3})(x-\frac{1}{4})\end{aligned}$$

sehingga interpolasi polinomial dalam bentuk lagrange adalah

$$\begin{aligned}p_2(x_j) &= \sum_{i=0}^2 \ell_i(x_j) f(x_i) = \ell_j(x_j) f(x_j) \\ p_2(x) &= -36(x-\frac{1}{4})(x-1) - 16(x-\frac{1}{3})(x-1) + 14(x-\frac{1}{3})(x-\frac{1}{4})\end{aligned}$$

Secara umum, seandainya kita telah menyatakan  $n$  titik  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ . Untuk setiap  $k$  antara 1 dan  $n$ , defini derajat  $n$  polinomial

$$\ell_k(x) = \frac{(x-x_1) \cdots (x-x_{k-1})(x-x_{k+1}) \cdots (x-x_n)}{(x_k-x_1) \cdots (x_k-x_{k-1})(x_k-x_{k+1}) \cdots (x_k-x_n)}$$

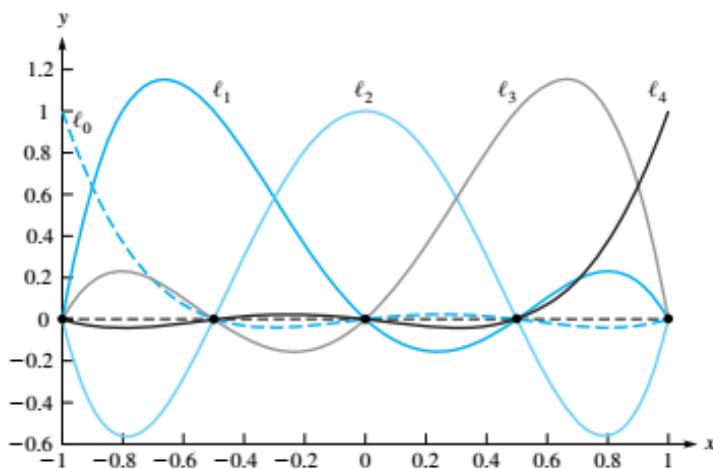
Yang menarik dari karakter  $\ell_k$  yaitu  $\ell_k(x_k) = 1$ , kemudian  $\ell_k(x_j) = 0$  dimana  $x_j$  salah satu dari titik titik lainnya. Maka definisi polinomial derajat  $n-1$  adalah

$$p_{n-1}(x) = y_1 \ell_1(x) + \cdots + y_n \ell_n(x).$$

Dengan mensubistusikan  $x_k$  pada  $x$  akan mendapatkan

$$p_{n-1}(x_k) = y_1 \ell_1(x_k) + \cdots + y_n \ell_n(x_k) = 0 + \cdots + 0 + y_k \ell_k(x_k) + 0 + \cdots + 0 = y_k$$


---



Gambar. 5.1: Cardinal Polinomial bentuk Lagrange

### 5.3 Interpolasi Polynomial: Bentuk Newton

Dalam contoh 5.2, kita telah menemukan bentuk Lagrange dari interpolasi polynomial:

$$p_2(x) = -36\left(x - \frac{1}{4}\right)\left(x - 1\right) - 16\left(x - \frac{1}{3}\right)\left(x - 1\right) + 14\left(x - \frac{1}{3}\right)\left(x - \frac{1}{4}\right)$$

Persamaan dapat disederhanakan kedalam bentuk

$$p_2(x) = -\frac{79}{6} + \frac{349}{6}x - 38x^2$$

Kita sekarang belajar bahwa polinomial ini dapat ditulis dalam bentuk lain yang disebut bentuk *nested newton*

$$p_2(x) = 2 + \left(x - \frac{1}{3}\right)[36 + \left(x - \frac{1}{4}\right)(-38)]$$

Itu terdiri dari beberapa operasi aritmetika dan direkomendasikan untuk menghitung  $p_2(x)$ . Bentuk Newton memiliki kelebihan jika ada titik-titik lagi yang akan ditambahkan. Terdapat metode yang disebut Algoritma Newton yang menghasilkan polinomial yang disebut dengan interpolasi polinomial bentuk Newton.

■ **Contoh 5.3** Dengan menggunakan Algoritma Newton tentukan interpolasi polinomial derajat terendah untuk tabel berikut

x	0	1	-1	2	-2
y	-5	-3	-15	39	-9

**Penyelesaian:** Dalam pembentukannya lima polinomial tersebut yaitu  $p_0, p_1, p_2, p_3, \text{ dan } p_4$ . Polinomial  $p_0$  didefinisikan dengan

$$p_0(x) = -5$$

Polinomial  $p_1$  mempunyai bentuk

$$p_1(x) = p_0(x) + c(x - x_0) = -5 + c(x - 0)$$

Interpolasi pada  $p_1$  adalah  $p_1(1) = -3$ . Sehingga kita memiliki  $-5 + c(1 - 0) = -3$ . Kemudian  $c = 2$  dan  $p_1$  adalah

$$p_1(x) = -5 + 2x$$

Polinomial  $p_2$  mempunyai bentuk

$$p_2(x) = p_1(x) + c(x - x_0)(x - x_1) = -5 + 2x + cx(x - 1)$$

Interpolasi pada  $p_2$  yaitu  $p_2(-1) = -15$ . Sehingga, kita mempunyai  $-5 + 2(-1) + c(-1)(-1 - 1) = -15$ . Ini menghasilkan  $c = -4$  sehingga

$$p_2(x) = -5 + 2x - 4x(x - 1)$$

Langkah langkah berikutnya sama, dan hasil akhir adalah bentuk Newton dari interpolasi polinomial

---


$$p_4(x) = -5 + 2x - 4x(x - 1) + 8x(x - 1)(x + 1) + 3x(x - 1)(x + 1)(x - 2)$$

#### 5.4 Nested Form

Sebelum melanjutkan, mari kita tulis lagi bentuk Newton dari interpolasi polinomial untuk perhitungan secara efisien. Perhatikan contoh berikut

- 
- **Contoh 5.4** Tulis polinomial  $p_4$  dari contoh 5.3 dalam bentuk nested dan gunakan untuk menghitung  $p_4(3)$

**Penyelesian:** Kita tulis  $p_4$  sebagai

$$p_4(x) = -5 + x(2 + (x - 1)(-4 + (x + 1)(8 + (x - 2)3)))$$

kemudian

$$\begin{aligned} p_4(3) &= -5 + 3(2 + 2(-4 + 4(8 + 3))) \\ &= 241 \end{aligned}$$

Penyelesain lain juga dalam bentuk nested adalah

$$p_4(x) = -5 + x(4 + x(-7 + x(2 + 3x)))$$

dari itu kita dapatkan

$$p_4(3) = -5 + 3(4 + 3(-7 + 3(2 + 3 \cdot 3))) = 241$$

Bentuk ini diperoleh dengan mengembangkan dan memfaktorkan secara sistematis dari polinomial aslinya. Ini dikenal sebagai nested form dan perhitungannya adalah dengan *nested multiplication*

## 5.5 Menghitung Koefisien $a_i$ menggunakan selisih terbagi (Divided Differences)

Untuk menjelaskan *nested multiplication* dalam cara formal ( sehingga nanti dapat dinyatakan ke bentuk kode program), perhatikan polinomial umum dalam bentuk Newton. Bentuknya adalah

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

Bentuk nested dari  $p(x)$  adalah

$$\begin{aligned} p(x) &= a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n)) \cdots \\ &= (\cdots ((a_n(x - x_{n-1}) + a_{n-1})(x - x_{n-2}) + a_{n-2}) \cdots )(x - x_0) + a_0 \end{aligned}$$

Interpolasi polinomial Newton dapat ditulis dengan

$$p_n(x) = \sum_{i=0}^n a_i \prod_{j=0}^{i-1} (x - x_j) \quad (5.6)$$

Dapat ditulis itu dengan

$$p_n(x) = \sum_{i=0}^n a_i \pi_i(x)$$

dimana

$$\pi_i(x) = \prod_{j=0}^{i-1} (x - x_j) \quad (5.7)$$

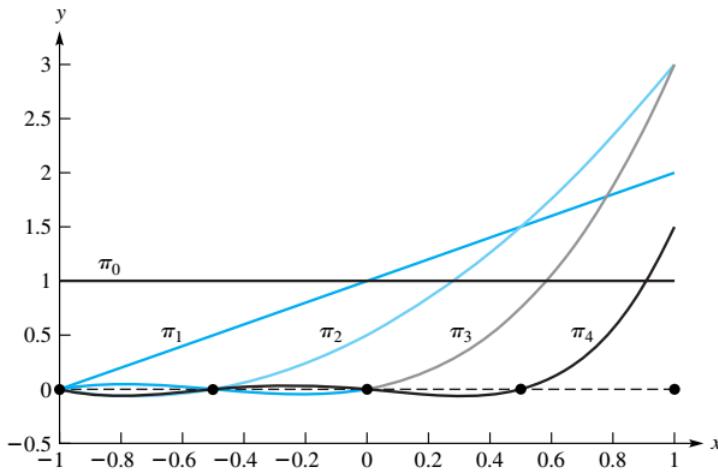
Gambar 5.2 menunjukkan beberapa polinomial pertama : $\pi_0(x), \pi_1(x), \pi_2(x), \pi_3(x), \pi_4(x)$ , dan  $\pi_5(x)$

Dalam menghitung  $p(t)$  untuk nilai numerik  $t$  tertentu, biasanya dimulai dari tanda kurung paling dalam, dan selanjutnya menghitung nilai-nilai berikutnya secara berurutan

$$\begin{aligned} v_0 &= a_n \\ v_1 &= v_0(t - x_{n-1}) + a_{n-1} \\ v_2 &= v_1(t - x_{n-2}) + a_{n-2} \\ &\vdots \\ v_n &= v_{n-1}(t - x_0) + a_0 \end{aligned}$$

## **5.5 Menghitung Koefisien $a_i$ menggunakan selisih terbagi (Divided Differences)**

Kembali lagi pada masalah menentukan koefisien  $a_0, a_1, \dots, a_n$ . Kita mulai dengan tabel nilai dari fungsi  $f$



Gambar. 5.2: Beberapa polinomial pertama bentuk Newton

$x$	$x_0$	$x_1$	$x_2$	$\dots$	$x_n$
$f(x)$	$f(x_0)$	$f(x_1)$	$f(x_2)$	$\dots$	$f(x_n)$

Titik titik  $x_0, x_1, \dots, x_n$  diasumsikan berbeda. Sebelumnya kita telah menetapkan bahwa masing masing  $n = 0, 1, \dots$ , terdapat polinomial tunggal  $p_n$  sehingga

- derajat dari  $p_n$  paling tinggi adalah  $n$
- $p_n(x_i) = f(x_i)$  untuk  $i = 0, 1, \dots, n$

. Itu menunjukkan bahwa  $p_n$  dapat dinyatakan dalam bentuk Newton.

$$\begin{aligned} p_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots \\ &\quad + a_n(x - x_0) \cdots (x - x_{n-1}) \end{aligned}$$

Yang sangat perlu diperhatikan tentang  $p_n$  adalah koefisien  $a_0, a_1, \dots, a_n$  yang tidak bergantung pada  $n$ . Dengan kata lain  $p_n$  diperoleh dari  $p_{n-1}$  dengan menambahkan lebih dari satu suku, tanpa merusak koefisien yang telah ada dalam  $p_{n-1}$  itu sendiri. Ini karena kita mulai dengan harapan bahwa  $p_n$  dapat dinyatakan dalam bentuk

$$p_n(x) = p_{n-1}(x) + a_n(x - x_0) \cdots (x - x_{n-1})$$

Suatu cara yang sistematis menentukan koefisien  $a_0, a_1, \dots, a_n$  adalah dengan cara menetapkan  $x$  sama dengan  $x_0, x_1, \dots, x_n$  dalam bentuk Newton (persamaan 5.6) dan menghasilkan persamaan

$$\begin{aligned} f(x_0) &= a_0 \\ f(x_1) &= a_0 + a_1(x_1 - x_0) \\ f(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\ &\vdots \end{aligned} \tag{5.8}$$

Dari persamaan (5.8) secara umum dapat dinyatakan dengan

$$f(x_k) = \sum_{i=0}^k a_i \prod_{j=0}^{i-1} (x_k - x_j) \quad (0 \leq k \leq n) \tag{5.9}$$

## 5.5 Menghitung Koefisien $a_i$ menggunakan selisih terbagi (Divided Differences)

Persamaan (5.8) dapat dipecahkan dengan  $a_i$  secara berurutan, yang dimulai dari  $a_0$ , perhatikan bahwa  $a_0$  bergantung pada  $f(x_0)$ , selanjutnya  $a_1$  bergantung pada  $f(x_0)$  dan  $f(x_1)$  dan seterusnya. Secara umum  $a_k$  bergantung pada  $f(x_0), f(x_1), \dots, f(x_k)$ . Dengan kata lain  $a_k$  bergantung pada nilai  $f$  pada simpul  $x_0, x_1, \dots, x_k$  dan dapat dinyatakan dengan

$$a_k = f[x_0, x_1, \dots, x_k] \quad (5.10)$$

Persamaan ini mendefinisikan  $f[x_0, x_1, \dots, x_k]$ . Nilai  $f[x_0, x_1, \dots, x_k]$  disebut dengan selisih terbagi k (divided difference of order k) untuk  $f$ . Perhatikan juga bahwa koefisien  $a_0, a_1, \dots, a_k$  adalah unik ditentukan dengan sistem persamaan (5.9).

Dengan menggunakan persamaan 5.8 kita dapat melihat bahwa beberapa selisih terbagi pertama ditulis dengan

$$\begin{aligned} a_0 &= f(x_0) \\ a_1 &= \frac{f(x_1) - a_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ a_2 &= \frac{f(x_2) - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} \end{aligned}$$


---

### ■ Contoh 5.5 Untuk tabel

x	1	-4	0
f(x)	3	13	-23

Tentukan nilai  $f[x_0], f[x_0, x_1]$  dan  $f[x_0, x_1, x_2]$

**Penyelesaian:** Kita tulis sistem persamaan dari persamaan 5.8 untuk kasus ini :

$$\begin{cases} 3 = a_0 \\ 13 = a_0 + a_1(-5) \\ -23 = a_0 + a_1(-1) + a_2(-1)(4) \end{cases}$$

Solusinya adalah  $a_0 = 3, a_1 = -2$  dan  $a_2 = 7$ . Sehingga untuk fungsi ini  $f[1] = 3, f[1, -4] = -2$  dan  $f[1, -4, 0] = 7$

Dengan notasi baru ini, bentuk Newton dari interpolasi polinomial didapatkan bentuknya

$$p_n(x) = \sum_{i=0}^n \left\{ f[x_0, x_1, \dots, x_i] \prod_{j=0}^{i-1} (x - x_j) \right\} \quad (5.11)$$

dengan ketentuan secara umum bahwa  $\prod_{j=0}^{-1}(x_j) = 1$ . Perhatikan bahwa koefisien dari  $x^n$  dalam  $p_n$  adalah  $[x_0, x_1, \dots, x_n]$  karena suku  $x^n$  terjadi hanya dalam  $\prod_{j=0}^{-1}(x - x_j)$ . Jika  $f$  adalah polinomial dari derajat  $\leq n-1$  maka  $f[x_0, x_1, \dots, x_n] = 0$  Untuk menghitung divided difference  $f[x_0, x_1, \dots, x_k]$  dilakukan berdasarkan persamaan (5.8) dan (5.9) dan melakukan komputasi secara rekursif. Dengan persamaan 5.9 untuk  $a_k$  sebagai berikut

$$f(x_k) = a_k \prod_{j=0}^{k-1} (x_k - x_j) + \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1} (x_k - x_j)$$

dan

$$a_k = \frac{f(x_k) - \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1} (x_k - x_j)}{\prod_{j=0}^{k-1} (x_k - x_j)}$$

Menggunakan persamaan (5.10) diperoleh

$$f[x_0, x_1, \dots, x_k] = \frac{f(x_k) - \sum_{i=0}^{k-1} f[x_0, x_1, \dots, x_i] \prod_{j=0}^{i-1} (x_k - x_j)}{\prod_{j=0}^{k-1} (x_k - x_j)} \quad (5.12)$$

---

**Algoritma 5.1:** Algoritma untuk menghitung Selisih Terbagi  $f$

---

Tetapkan  $f[x_0] = f(x_0)$ ;  
**for**  $k := 1$  to  $n$  **do**  
  └ Hitung  $f[x_0, x_1, x_2, \dots, x_k]$  dengan Persamaan 5.12;

---

## 5.6 Algoaritma dan Pseudocode menghitung Selisih Terbagi

---

**Algoritma 5.2:** Algoritma untuk menghitung Selisih Terbagi  $f$

---

Integer i,j,n    real array  $(a_{ij})_{0:n \times 0:n}, (x_i)_{0:n}$   
**for**  $i := 0$  to  $n$  **do**  
  └  $a_{i0} \leftarrow f(x_i)$   
**for**  $j := 1$  to  $n$  **do**  
  └ **for**  $i := 1$  to  $n - j$  **do**  
    └  $a_{ij} \leftarrow (a_{i+1,j-1} - a_{i,j-1}) / (x_{i+j} - x_i)$

---

Jika selisih terbagi dihitung hanya untuk membangun bentuk Newton dari interpolasi polinomial

$$p_n(x) = \sum_{i=0}^n a_i \prod_{j=0}^{i-1} (x - x_j)$$

dimana  $a_i = f[x_0, x_1, x_2, \dots, x_i]$  tidak diperlu disimpan seluruhnya. Hanya  $f[x_0], f[x_0, x_1], \dots, f[x_0, x_1, \dots, x_n]$  yang perlu disimpan. Ketika array satu dimensi  $(a_i)_{0:n}$  digunakan, selisih terbagi dapat algorit-

manyanya sebagai berikut

---

**Algoritma 5.3:** Algoritma untuk menghitung Selisih Terbagi array 1 dimensi  $f$

---

```

Input :Integer i,j,n    real array  $(a_{ij})_{0:n}, (x_i)_{0:n}$ 
Output:  $a_i$ 
for  $i := 0$  to  $n$  do
   $a_{i0} \leftarrow f(x_i)$ 
for  $j := 1$  to  $n$  do
  for  $i := n$  to  $j$  step  $-1$  do
     $a_i \leftarrow (a_i - a_{i-1}) / (x_i - x_{i-j})$ 
  
```

---

Berikut kode program menghitung selisih terbagi Newton

```

def newton_divided_diff(X, Y, n):
    V = []
    for j in range(n):
        V[j, 0] = Y[j]

    for i in range(1, n):
        for j in range(n-i):
            V[j, i] = (V[j+1, i-1] - V[j, i-1]) / (X[j+i] - X[j])

    a = np.zeros(n)
    for i in range(n):
        a[i] = V[0, i]

    return a
  
```

Kode Program 5.1: Menghitung Selisih Terbagi Newton

Contoh dari berikut untuk dari data berikut :

x	0	1	2	3
y	2	1	0	-1

```

dX = [0, 1, 2, 3]
Y = [2, 1, 0, -1]
a = newton_divided_diff(X, Y, 4)
print(a)
  
```

keluaran dari data:

```
[ 2. -1.  0.  0.]
```

Sehingga persamaannya adalah : $p(x) = 2 + (-1)(x - 0) = 2 - x$

## 5.7 Interpolasi Polynomial dengan Algoritma Neville

Metode lain untuk mendapatkan interpolasi polinomial dari tabel berikut dengan menggunakan algoritma Neville

x	$x_0$	$x_1$	...	$x_n$
y	$y_0$	$y_1$	...	$y_n$

Algoritma ini sama seperti algoritma Newton dalam tahapan mementuk polinomial. Jika  $P_{a,b,\dots,s}(x)$  adalah interpolasi polinomial dari serangkaian simpul-simpul  $x_a, x_b, \dots, x_s$ . Kita mulai dengan polinomial konstanta  $P_i(x) = f(x_i)$ . Dengan memilih dua simpul  $x_i$  dan  $x_j$  dengan  $i > j$ , kita definisikan secara rekursif

$$P_{a,b,\dots,s}(x) = \left(\frac{x - x_j}{x_i - x_j}\right) P_{a,b,\dots,j-1,j+1,\dots,v}(x) + \left(\frac{x_i - x}{x_i - x_j}\right) P_{a,b,\dots,i-1,i+1,\dots,s}(x)$$

Dengan menggunakan rumus tersebut secara berulang, kita dapat membuat array polinomial.

$x_0$	$P_0(x)$				
$x_1$	$P_1(x)$	$P_{0,1}(x)$			
$x_2$	$P_2(x)$	$P_{1,2}(x)$	$P_{0,1,2}(x)$		
$x_3$	$P_3(x)$	$P_{2,3}(x)$	$P_{1,2,3}(x)$	$P_{0,1,2,3}(x)$	
$x_4$	$P_4(x)$	$P_{3,4}(x)$	$P_{2,3,4}(x)$	$P_{1,2,3,4}(x)$	$P_{0,1,2,3,4}(x)$

Disini, setiap polinomial berikutnya dapat ditentukan dari dua polinomial tetangganya pada kolom sebelumnya. Kita dapat menyederhanakan notasi tersebut yaitu dengan

$$Q_{ij}(x) = P_{i-j,i-j+1,\dots,i-1,i}(x)$$

dimana  $Q_{ij}(x)$  untuk  $i \geq j$  menyatakan interpolasi polinomial derajat  $j$  pada  $j+1$  simpul  $x_{i-j}, x_{i-j+1}, x_{i-1}, x_i$ . Berikut kita menulis kembali relasi diatas dengan

$$Q_{ij}(x) = \left(\frac{x - x_{i-j}}{x_i - x_{i-j}}\right) Q_{i,j-1}(x) + \left(\frac{x_i - x}{x_i - x_{i-j}}\right) Q_{i-1,j-1}(x)$$

Sehingga ditampilkan array menjadi

$x_0$	$Q_{00}(x)$				
$x_1$	$Q_{10}(x)$	$Q_{11}(x)$			
$x_2$	$Q_{20}(x)$	$Q_{21}(x)$	$Q_{22}(x)$		
$x_3$	$Q_{30}(x)$	$Q_{31}(x)$	$Q_{32}(x)$	$Q_{33}(x)$	
$x_4$	$Q_{40}(x)$	$Q_{41}(x)$	$Q_{42}(x)$	$Q_{43}(x)$	$Q_{44}(x)$

■ **Contoh 5.6** Tabel berikut adalah hasil fungsi  $f(x) = \ln(x)$ . Dengan menggunakan metode interpolasi Neville dengan pembulatan 4 digit, hitunglah pendekatan  $f(2.1)$ .

$i$	$x_i$	$\ln x_i$
0	2.0	0.6931
1	2.2	0.7885
2	2.3	0.8329

**Penyelesaian:** Karena  $x - x_0 = 0.1, x - x_1 = -0.1, x - x_2 = -0.2$  dengan  $Q_{0,0} = 0.6931, Q_{1,0} = 0.7885$  dan  $Q_{2,0} = 0.8329$  maka

$$Q_{1,1} = \frac{1}{0.2}[(0.1)0.7885 - (-0.1)0.6931] = \frac{0.1482}{0.2} = 0.7410$$

dan

$$Q_{2,1} = \frac{1}{0.1}[(-0.1)0.8329 - (-0.2)0.7885] = \frac{0.07441}{0.1} = 0.7441$$

Sehingga pendekatan akhir dari data ini adalah

$$Q_{2,2} = \frac{1}{0.3}[(0.1)0.7441 - (-0.2)0.7410] = \frac{0.2276}{0.3} = 0.7420$$

Untuk menghitung interpolasi polynomial P dari  $n + 1$  nilai  $x_0, \dots, x_n$  dari x untuk fungsi  $f$  berikut adalah algoritma dan kode program interpolasi Neville

---

**Algoritma 5.4:** Algoritma Interpolasi Neville

---

Input:

real:  $x, x_0, x_1, \dots, x_n$

real :  $f(x_0), f(x_1), \dots, f(x_n)$  adalah nilai nilai dari  $Q_{0,0}, Q_{1,0}, Q_{2,0}, \dots, Q_{n,0}$  sebagai kolom pertama dari Q

Output : tabel Q dengan  $P(x) = Q_{n,n}$

**for**  $i := 1$  to  $n$  **do**

$$\quad \quad \quad \text{for } j := 1 \text{ to } i+1 \text{ do} \\ \quad \quad \quad \quad \quad \quad Q_{i,j} = \frac{(x-x_{i-j})Q_{i,j-1} - (x-x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$$


---

Kode program dari Algoritma Neville adalah sebagai berikut

```
def neville(x, y, x_int):
    """
    Menghitung Interpolasi menggunakan Neville polynomial
    Parameters:
        x: Array dari nilai x
        y: Array dari nilai y
        x_int: nilai untuk interpolate
    Keluaran:
        y_int: Nilai yang di interpolasi
        q: Matrik koefisien
    """

    n = x.size
    q = np.zeros((n, n - 1))
    # masukkan 'y' pada kolom pertama dari matrik 'q'
    q = np.concatenate((y[:, None], q), axis=1)

    for i in range(1, n):
```

```

for j in range(1, i + 1):
    q[i, j] = ((x_int - x[i - j]) * q[i, j - 1] -
                (x_int - x[i]) * q[i - 1, j - 1]) / (x[i]
                                                ] - x[i - j])

y_int = q[n - 1, n - 1]
return [y_int, q]

```

Kode Program 5.2: Menghitung interpolasi Neville

Contoh perhitungan :

```

x = [2, 2.2, 2.3]
y = [0.6931, 0.7885, 0.8329]

X = np.array(x)
Y = np.array(y)
[y_int, q] = neville(X, Y, 2.1)
print(q)

```

Hasil dari perhitungan diatas adalah

```

[[0.6931 0.      0.      ]
 [0.7885 0.7408 0.      ]
 [0.8329 0.7441 0.7419]]

```



## **6. 06-Integrasi Numerik**



## **7. 07-Turunan Numerik**



## **8. 08-Penyelesaian Persamaan Diferensial Biasa**



## **9. 09-Penyelesaian Persamaan Diferensial Parti**



## **10. 10-Turunan Numerik**



## **11. 11-Metode Monte Carlo dan Simulasi**



## Daftar Pustaka

- [1] Richard L Burden and J Douglas Faires. *Numerical analysis*. Cengage Learning, 2011.
- [2] Steven C Chapra, Raymond P Canale, et al. *Numerical methods for engineers*. Boston: McGraw-Hill Higher Education, 2010.
- [3] Santanu Saha Ray. *Numerical analysis with algorithms and programming*. CRC Press, 2018.