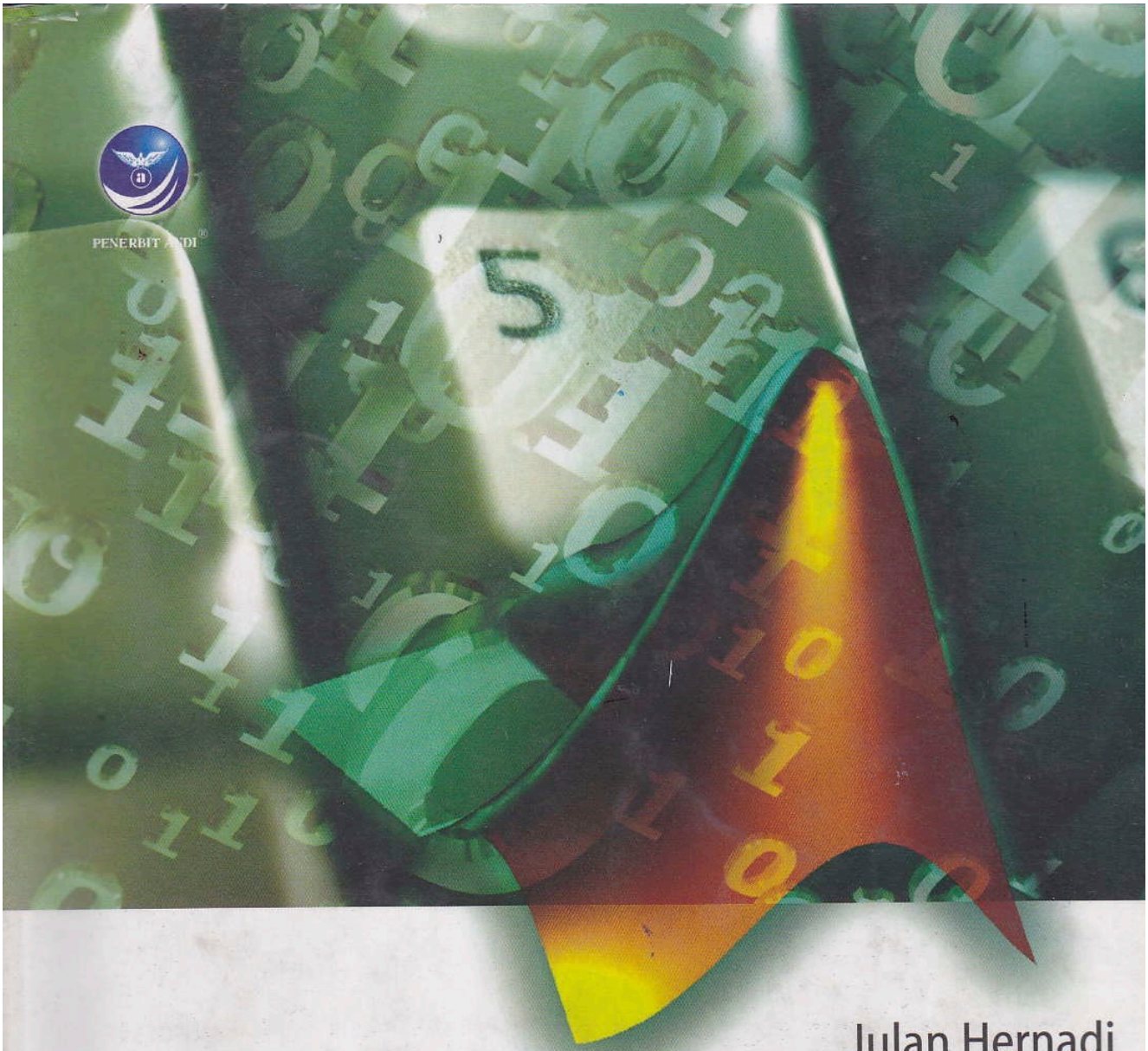


PENERBIT ANDI

Julan Hernadi

MATEMATIKA NUMERIK

dengan Implementasi MATLAB

MATEMATIKA NUMERIK dengan IMPLEMENTASI MATLAB

EDISI PERTAMA

Julan HERNADI

ANDI OFFSET YOGYAKARTA
2012

Didedikasikan untuk:

Alm. Ahmad Ismail (ayah) dan Hj Konaria (ibu)

Sri Purnama Surya (isteri), Ahmad Zhafir HERNADI (anak)

Herri Gusmedi, Aprillina, dan Yuniarni (adik-adik)

DAFTAR ISI

Kata Pengantar

Kebutuhan pada keterampilan komputasi dewasa ini meningkat tajam seiring pesatnya perkembangan teknologi komputer dan semakin rumitnya masalah yang harus diselesaikan dengan perhitungan cepat dan akurat. Perpaduan ilmu matematika dan ilmu komputer melahirkan bidang kajian baru yang dikenal dengan nama komputasi saintifik (*scientific computing*). Banyak algoritma komputasi yang telah dikembangkan puluhan tahun lalu tidak dapat diimplementasikan dikarenakan kendala rumit dan panjangnya perhitungan. Namun, pada era komputasi saintifik semua masalah tersebut dapat diatasi dengan baik, bahkan semakin berkembang dalam ilmu-ilmu terapan (*applied sciences*). Bidang-bidang pencitraan komputer (*computer vision*) seperti *bio and medical-imaging*, pengolahan signal digital dalam bidang telekomunikasi, dan lain-lain merupakan buah dari kemajuan di bidang komputasi saintifik.

Secara tradisional komputasi saintifik berkembang dari analisis numerik, yaitu suatu cabang ilmu matematika yang dirancang untuk menyelesaikan masalah matematika yang tidak dapat dilakukan secara analitis. Pada analisis numerik materi tersajikan dalam bentuk teorema sebagai dasar untuk menyusun dan menjustifikasi algoritma komputasi. Karena bentuknya abstrak dan sulit dipahami maka umumnya analisis numerik tidak begitu menarik bagi sebagian besar mahasiswa bidang sains dan teknologi, khususnya di Indonesia. Fenomena memprihatinkan di negeri tercinta ini adalah pembelajaran lebih diarahkan pada aspek keterampilan praktis dan pragmatis, kurang menyentuh aspek penalaran atau kognitif tingkat tinggi. Fenomena ini juga terjadi pada pembelajaran metode numerik. Para mahasiswa dan juga dosen cenderung menghindari hal-hal yang bersifat teoretis, tetapi lebih suka pada aspek praktis dan pragmatis. Mayoritas mahasiswa termasuk sebagian besar dosen lebih memilih fasilitas *instant* yang sudah dalam bentuk *toolbox* daripada memahami latar belakang teoretis. Bahkan persyaratan awal (*prerequisites*) penggunaan suatu metode komputasi tidak begitu dihiraukan. Padahal pemahaman latar belakang teoretis, mengapa dan bagaimana suatu metode digunakan adalah sangat penting.

Oleh karena itu, buku ini disusun untuk menanamkan keterampilan dasar komputasi praktis namun tetap memperhatikan landasan teoretis secara matematis. Berbagai metode

dasar dalam matematika numerik disajikan secara runtut, dimulai dari mengapa metode itu diciptakan, bagaimana ide dasar, penyusunan algoritma, kekurangan dan kelebihannya, sampai pada bagaimana melakukan implementasi numerik dengan komputer. Dasar teoretis matematika seperti teorema selalu diberikan sebagai alat justifikasi metode numerik yang sedang pelajari. Untuk kasus sederhana, teorema berikut buktinya diberikan secara lengkap. Sedangkan teorema yang buktinya cukup rumit dan membutuhkan pengetahuan matematika lebih dalam hanya diberikan teoremanya dan pembaca yang berminat memahami buktinya dirujuk ke referensi terkait. Implementasi numerik dilakukan dengan menggunakan MATLAB, yaitu sebuah bahasa pemrograman dengan performa tinggi untuk komputasi teknis. Hampir semua metode numerik yang dibahas dalam buku ini disusun algoritmanya berikut kode MATLAB yang sesuai, walaupun sudah disiapkan pada *toolbox* MATLAB. Hasil numerik keluaran komputer diulas secara kritis dikaitkan dengan dasar teoretisnya. Beberapa kode bawaan toolbox MATLAB yang relevan dijelaskan cara penggunaan termasuk kelebihan dan kekurangannya. Karena ada dua aspek penekanan, yaitu aspek teori matematika dan implementasi numerik maka buku ini diberi judul "Matematika Numerik dengan Implementasi MATLAB".

Buku ini dirancang untuk perkuliahan Metode Numerik Dasar dengan bobot 3 SKS ditambah 1 kali praktikum komputer per minggu untuk melakukan implementasi numerik. Materi pada buku ini mencakup topik-topik elementer dalam metode numerik, seperti sistem persamaan linear, persamaan taklinear, interpolasi polinomial dan aproksimasi integral dan diferensial. Walaupun buku ini membutuhkan keterampilan MATLAB, pembaca yang belum memiliki pengalaman dengan MATLAB tetap dapat mempelajarinya karena tutorial MATLAB diberikan pada Bab 1. Pada Bab 2 sengaja disampaikan topik Pendahuluan Komputasi Saintifik agar pembaca memiliki pengetahuan memadai tentang pentingnya bidang ini dalam sains dan teknologi. Beberapa topik penting tetapi jarang diberikan dalam perkuliahan metode numerik seperti Metode Iterasi Titik Tetap, Interpolasi Hermite, dan Metode Integral Gauss disajikan secara materi pengayaan. Soal-soal latihan diberikan pada akhir beberapa pokok bahasan atau setiap akhir Bab. Soal-soal diberikan berimbang antara pembentukan keterampilan (*skill*) dan pemahaman pengetahuan (*knowledge and understanding*). Materi pada buku ini akan menjadi dasar untuk mempelajari materi pada metode numerik lanjutan, seperti metode penyelesaian persamaan integral, persamaan diferensial parsial, masalah optimisasi lanjutan, dan lain-lain. Oleh karena itu buku ini sangat cocok digunakan oleh mahasiswa yang kuliah pada jenjang S-1 bidang MIPA (Matematika, Statistika, Ilmu Komputer, Fisika, Kimia, Biologi, Farmasi, dan lain-lain) dan teknologi (semua jurusan Teknik), termasuk mahasiswa yang kuliah pada jurusan Pendidikan MIPA di LPTK atau FKIP. Metode numerik merupakan mata kuliah wajib pada sebagian besar Program Studi yang disebutkan di atas.

Pengetahuan prasyarat untuk mempelajari buku ini adalah cukup dengan kalkulus biasa dan aljabar linear elementer atau matriks. Seperti keterampilan lainnya, keterampilan pada MATLAB dapat ditingkatkan sambil jalan. Bagi mahasiswa yang telah mempunyai pengalaman cukup dalam MATLAB maka Tutorial MATLAB pada Bab 1 dapat dilewati. Ketersediaan laboratorium komputer di kampus yang terinstal program MATLAB sangat penting dalam mempelajari buku ini.

Penulisan buku ini telah diupayakan sederhana dan menarik, baik dari aspek bahasa maupun urutan penyajianya sehingga diharapkan menambah minat pembaca dalam mempelajarinya. Banyaknya ilustrasi grafik dimaksudkan untuk memudahkan pembaca dalam memahami maksud yang disajikan dalam bentuk narasi/kalimat. Pengetikan buku ini menggunakan program LyX sehingga tampilan *equation* yang dihasilkan sangat sempurna, jauh lebih baik dari *equation* pada MS Words. Lebih dari itu pengetikan naskah ini menggunakan sistem operasi Linux dan beberapa *software open source,* seperti Inkscape untuk grafis dan program JabRef untuk penataan referensi.

Buku ini sudah disiapkan penulis sejak tiga tahun yang lalu dan selama itu sudah digunakan sebagai buku teks dalam kuliah metode numerik yang penulis ampu sendiri. Oleh karena itu pada kesempatan ini penulis ingin menyampaikan terima kasih kepada mahasiswa dan rekan dosen pada Program Studi Pendidikan Matematika FKIP Unmuh Ponorogo dan Program Studi Matematika FMIPA UAD Yogyakarta karena telah menggunakan naskah ini sebagai buku teks wajib kuliah metode numerik. Terima kasih kepada mahasiswa dan rekan dosen yang telah memberikan koreksi terhadap beberapa kekeliruan kecil baik kesalahan ketik maupun kesalahan hitung.

Pada kesempatan ini penulis menyampaikan terima kasih dan penghargaan kepada istri tercinta Sri Purnama Surya dan ananda tersayang Ahmad Zhafir Hernadi atas dorongan semangat yang diberikan selama ini. Kepada ibunda Hj Konaria dan adik-adik: Herri Gusmedi, Aprillina dan Yuniarni, penulis menyampaikan penghargaan dan terima kasih atas doa dan harapan kalian selama ini. Kepada ayahnda tercinta Ahmad Ismail yang telah kembali menghadap Sang Khalik, penulis selalu berdoa semoga senantiasa bahagia dalam surgaNya. Semoga karya kecil ini bermanfaat bagi orang banyak dan menjadi pemicu lahirnya karya-karya lebih besar berikutnya.

Akhirnya, penulis menyadari bahwa naskah ini masih banyak kekurangannya. Oleh karena itu, segala kritik dan saran untuk perbaikan pada penerbitan edisi berikutnya sangat diharapkan dari para pembaca. Saran dan kritik dapat disampaikan melalui email penulis julan_hernadi@yahoo.com.

Yogyakarta, Maret 2012
Penulis

1 PENGENALAN MATLAB

MATLAB adalah bahasa pemrograman dengan performa tinggi untuk komputasi teknis. MATLAB menggabungkan tiga unsur, yaitu komputasi, visualisasi dan pemrograman ke dalam satu wadah. MATLAB merupakan singkatan dari *matrix laboratory*. MATLAB merupakan sistem interaktif dengan elemen basis data berupa *array*. Artinya setiap data yang dimasukkan pada MATLAB diinterpretasikan sebagai *array*. *Array* berdimensi satu adalah vektor dan *array* berdimensi dua adalah matriks.

Pada bagian ini akan diperkenalkan MATLAB sekilas pandang. Penjabaran lebih lengkap dapat dilihat pada beberapa manual MATLAB, seperti pada [**?**, **?**]. Tutorial ini mengacu pada MATLAB versi 6.5, sedangkan untuk MATLAB versi lebih baru tinggal menyesuaikan seperlunya.

1.1 Desktop MATLAB

Ketika pertama kali kita memulai MATLAB, akan muncul desktop MATLAB yang memuat panel-panel untuk pengolahan file, variabel dan aplikasi apa pun yang berkaitan dengan MATLAB. Desktop yang muncul sebagai berikut (Gambar 1.1):

Tampilan ini dapat diubah sesuai selera, yaitu dengan cara membuka, menutup, atau memperbesar/memperkecil ukuran window yang tersedia. Unsur-unsur desktop MATLAB dan fungsinya diberikan sebagai berikut.

- **Command Window** adalah tempat untuk memasukkan variabel dan mengeksekusi program MATLAB atau M-files.
- **Command History** berisi rekaman perintah yang pernah ditulis pada *command window*.
- **Launch Pad** berfungsi untuk memudahkan dalam mengakses fasilitas, demonstrasi dan dokumentasi pada MATLAB.

Variabel merupakan nama untuk suatu *array* bilangan. Penulisan variabel dimulai dengan huruf, dapat diikuti dengan huruf atau angka atau *underscore*. MATLAB hanya dapat mengenal 31 karakter pertama dari nama variabel. Sebagai contoh, bila pada *command window* diketik

```
>> apel_malang1=25
```

dan setelah menekan enter maka diperoleh

```
apel_malang1 =

    25
```

Di sini `apel_malang1` adalah nama variabel yang disimpan pada ruang kerja (*workspace)* dengan nilai 25. Bila terdapat variabel baru dengan nama sama maka secara otomatis nilai variabel lama akan terhapus dan berlaku nilai variabel baru. MATLAB sangat sensitif dalam membaca variabel. Ia membedakan antara huruf besar dan huruf kecil. Contoh, `A` dan `a` adalah dua variabel yang berbeda. Pada satu baris kita dapat mendefinisikan lebih dari satu variabel dengan cara memberikan tanda koma (,) di antara dua variabel. Contoh

```
>> x=50, X=45
```

akan menghasilkan

```
x =

    50

X =

    45
```

Perhatikan, keduanya merupakan variabel yang berbeda. Semua variabel yang telah didefinisikan tersimpan di dalam *workspace*. Bila kita ingin melihat kembali variabel apa saja yang telah tersimpan, lihat pada *workspace*, atau pada *command window* dengan mengetik **who** atau **whos**.

Sebagaimana telah dijelaskan bahwa variabel-variabel yang telah didefinisikan tidaklah permanen tersimpan pada *workspace*. Untuk menyimpan variabel digunakan perintah

```
>>save {nama file}
```

untuk menyimpan semua variabel, atau

```
3,-99, 0.0001, 9.6397238, 1.60210e-20, 6.02252e+23, 1i,
-3.14159j, 3e5i.
```

Semua bilangan tersimpan secara internal dengan menggunakan format *long* yang dispesifikasi oleh standar titik mengambang (*floating-point*) IEEE. Bilangan titik mengambang mempunyai presisi kira-kira 16 digit desimal signifikan dalam jangkauan antara 10^{-323} sampai dengan 10^{308}. Besaran atau magnitud bilangan di atas 10^{308} dianggap ∞, sedangkan di bawah 10^{-323} dianggap 0. Lebih jelasnya dibahas pada bagian berikutnya.

Operator aritmatika

Ada tujuh operator aritmatika yang berlaku pada MATLAB, yaitu penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), pembagian kiri (\), pemangkatan (^) dan konjugasi (’). Perhatikan contoh berikut.

```
> a=3;b=6;c=2+5i;
>> a+b
ans =

    9

>> a-b
ans =

    -3

>> a*b
ans =

    18

>> a/b
ans =

    0.5000

>> a\b

    ans = 2

>> c’
ans =

    2.0000 - 5.0000i
```

singkatan dari *not-a-number,* menunjukkan ekspresi yang takterdefinisi, seperti $0/0$ atau `Inf-Inf`. Secara matematis, ekspresi $a/0$ dengan $a \neq 0$ tidak terdefinisi, tetapi MATLAB memberikan hasil takterhingga untuk ekspresi ini. Pertentangan ini dapat dipahami MATLAB menganggap 0 sebagai bilangan yang magnitudnya sangat kecil.

```
>> factorial(170)
ans =

    7.257415615307994e+306

>> factorial(171)
ans =

    Inf
```

Di sini `factorial(n)` adalah perintah untuk $n! := n(n-1)\cdots(3)(2)(1)$. Dalam contoh ini, MATLAB masih dapat menghitung 170! tetapi tidak mampu lagi menghitung faktorial yang lebih besar karena sudah melebihi bilangan realmax, yaitu sekitar 1.7977e+308.

```
>> 1/0
Warning: Divide by zero.
ans =

    Inf

>> 0/0
Warning: Divide by zero.
ans =

    NaN

>> Inf-Inf
ans =

    NaN

>> Inf+Inf
ans =

    Inf

>>
```

```
>> cos(60)
ans =

    -0.9524

>> asin(0.5)
ans =

    0.5236

>> atan(1)
ans =

    0.7854

>> exp(1)
ans =

    2.7183

>> sqrt(2)
ans =

    1.4142

>> log10(100)
ans =

    2

>> log(exp(1))
ans =

    1

>> log2(8)
ans =

    3
```

Silakan mengeksplorasi fungsi-fungsi lainnya pada MATLAB. Satu kelompok fungsi yang sering digunakan dalam pemrograman numerik adalah fungsi pembulatan.

c) Buatlah variabel x_y_z dengan x_y_z= $(xy)z$

d) Simpanlah variabel-variabel xy, p, dan x_y_z dengan nama file coba1, hilangkan dan hapus semua data yang ada pada *command window*, kemudian panggil kembali ketiga variabel ini.

2. Diberikan variabel x = 30, definisikan variabel θ yang menyatakan variabel x dalam derajat. Sebaliknya diketahui variabel $\lambda = 1050$, definisikan variabel y yang menyatakan λ dalam radian. (Petunjuk: gunakan perintah `rad2deg` dan `deg2rad`, buka help MATLAB kalau perlu)

3. Misalkan $x = 2 - i$ dan $y = -3 + 2i$. Lakukan operasi berikut.

   a) $(x + y)^2$

   b) x/y

   c) xy

   Apakah hasil yang diperoleh sesuai dengan teori matematika ?

4. Coba periksa hasil yang diberikan oleh MATLAB untuk kalkulasi berikut.

$$\infty^{\infty},\ 0^{\infty},\ \infty^{0},\ \frac{\infty}{\infty},\ \infty \cdot \infty,$$

   Apakah hasilnya sesuai dengan teori ∞ matematika ?

5. Gunakan MATLAB untuk menghitung nilai fungsi trigonometri berikut.

$$\sin(\pi/5),\ \cos(500),\ \arccos(3/2), \arctan(3/2).$$

   Cocokkan dengan hasil matematika teoretis.

6. Gunakan MATLAB untuk menghitung nilai fungsi trigonometri hiperbolik berikut.

$$\cosh(2\pi/3),\ \tanh(1.2),\ \operatorname{arccsch}(-1/3)$$

   Cocokkan dengan hasil matematika teoretis.

7. Hitunglah dengan menggunakan MATLAB kuantitas berikut.

   a) $\ln e^3$ dan $\log 10^3$

   b) $\ln 2 + \ln 3 + \ln 4 + e^2$

   c) $\sqrt{\sqrt{\sqrt{e \log 2}}}$

Contoh ini memberikan cara mendefinisikan vektor baris dan vektor kolom. Penggunaan tanda titik koma antarbilangan menghasilkan vektor kolom. Beberapa operasi yang biasa dilakukan pada vektor, antara lain besar vektor (*norm*), hasil kali silang antara dua vektor (*cross product*), hasil kali titik antara dua vektor (*dot product*), operasi aritmatika elemen per elemen (*piesewise operation*), elemen maksimum, elemen minimum.

Untuk memahami pembahasan ini seyogianya pembaca sudah memahami teori vektor dari Aljabar Linear. Misalkan $a = (a_1, \cdots, a_n)$ dan $b = (b_1, \cdots, b_n)$ dua vektor yang diketahui. Berikut beberapa fungsi vektor MATLAB yang sering digunakan.

TABEL 1.4: Fungsi dan operasi vektor pada MATLAB

Sintaksis MATLAB	Lambang matematika	Definisi
`a(k)`	a_k	komponen ke k vektor a
`a(end)`	a_n	komponen terakhir vektor a
`norm(a)`	$\|\|a\|\| := (\sum_{k=1}^{n} a_k)^{1/2}$	norma atau besar vektor a
α`*a`	αa	perkalian skalar vektor a
`max(a)`	$\max\{a_k\}_{k=1,\cdots n}$	komponen terbesar vektor a
`min(a)`	$\min\{a_k\}_{k=1,\cdots,n}$	komponen terkecil vektor a
`length(a)`	$n(a)$	panjang atau ukuran vektor a
`sum(a)`	$\sum_{k=1}^{n} a_k$	jumlah semua bil pada vektor a
`prod(a)`	$\prod_{k=1}^{n} a_k$	hasil perkalian semua bil pada vektor a
`dot(a,b)`	$\sum_{k=1}^{n} a_k b_k$	hasil kali titik vektor a dan b
`a.*b`	$(a_1 b_1, \cdots, a_n b_n)$	perkalian antar komponen

Berikut contoh pemakaian fungsi dan operasi vektor.

```
>> a=[1 2 -3];b=[2 2 1];
>> length(a)
ans =

    3
```

Ini dapat diverifikasi karena vektor `a` mempunyai 3 komponen.

```
>> norm(a)
ans =

    3.7417
```

```
    0.5000 1.0000 -3.0000
```

Satu lagi contoh fungsi yang sering didefinisikan pada vektor.

```
>> q=[4.7506 1.1557 3.0342 2.4299 4.4565 3.8105 2.2823...
0.0925 4.1070 2.2235];
>> max(q)
ans =

    4.7506

>> min(q)
ans =

    0.0925

>> sum(q)
ans =

    28.3429

>> prod(q)
ans =

    1.3255e+003

>> sort(q)
ans =

    Columns 1 through 8
    0.0925 1.1557 2.2235 2.2823 2.4299 3.0342 3.8105 4.1070
    Columns 9 through 10
    4.4565 4.7506
```

Baris terakhir adalah perintah untuk mengurutkan secara menaik elemen-elemen dalam vektor q.

Operator kolon untuk mendefinisikan *array*

Notasi titik dua (:) atau disebut kolon adalah salah satu operator yang cukup penting dalam MATLAB. Notasi ini banyak digunakan untuk mendefinisikan *array* dengan cepat.

Penulisan lebih dari satu baris ini dapat dilihat pada *array* `ddd` di atas. Bila pendefinisian *array* tidak sesuai dengan alur logika maka akan dihasilkan *array* kosong. Perhatikan perintah berikut.

```
>> ko=-5:-2:10
ko =

     Empty matrix: 1-by-0
```

Ini berarti tidak ada bilangan yang dapat dihasilkan oleh perintah ini. Logikanya, untuk membuat *array* dari -5 menuju 10 (*array* naik) tidaklah mungkin dilakukan dengan memasukkan step turun -2.

Matriks

Matriks pada MATLAB berupa *array* bilangan yang berbentuk persegi panjang. Seperti biasa ukuran matriks ditentukan oleh banyaknya baris dan kolom. Secara khusus, bilangan atau skalar adalah matriks berukuran 1×1. Sedangkan vektor dipandang sebagai matriks yang hanya mempunyai satu kolom atau satu baris. Jadi semua data numerik di dalam MATLAB dipandang sebagai matriks. Terdapat beberapa cara mendefinisikan matriks dalam MATLAB, antara lain

1. cara manual, yaitu memasukkan elemen matriks satu per satu,
2. mengambil matriks dari file data eksternal,
3. membangun matriks dengan menggunakan fungsi built-in, dan
4. mendefinisikan matriks dengan m-file.

Berikut contoh mendefinisikan matriks secara manual.

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =

    16   3   2  13
     5  10  11   8
     9   6   7  12
     4  15  14   1
```

Cara mendefinisikan matriks lainnya akan dibahas secara bertahap seiring dengan pendalaman pada tahap berikutnya.

```
     3  2  13
    10  11  8
     6  7  12

>> A(3,:)
ans =

    9  6  7  12

>> A(1:3,2)
ans =

    3
    10
    6

>> A(3,2:end)
ans =

    6  7  12
```

Beberapa matriks yang diperoleh dari elemen-elemen matriks A dapat juga disebut sub matriks A.

Operasi pada matriks

Operasi aritmatika pada matriks terdiri dari penjumlahan (+), pengurangan (-), perkalian (*), perkalian antarelemen (.*), pemangkatan (^), transpose ($'$), pembagian kiri (\), dan pembagian kanan (/). Untuk mengetahui bagaimana operasi-operasi ini bekerja perhatikan beberapa contoh berikut.

```
>> A=[1 2 3;3 2 2; 1 1 4]; B=[4 4 1;2 3 1; 6 5 2];
>> A-B
ans =

    -3  -2  2
     1  -1  1
    -5  -4  2
```

merupakan hasil pengurangan matriks A-B. Perkalian matriks A dan B dilakukan sebagai berikut.

```
>> Q=inv(A)
Q =

    -0.5455  0.4545  0.1818
     0.9091 -0.0909 -0.6364
    -0.0909 -0.0909  0.3636
```

Untuk meyakinkan kebenaran hasil perhitungan MATLAB ini kita dapat mengujinya dengan melakukan perintah berikut.

```
>>Q*A
```

atau

```
>>A*Q
```

Hasil kedua operasi ini tentunya menghasilkan matriks identitas.

Berikut ini diperkenalkan salah satu kemampuan MATLAB dalam menyelesaikan sistem persamaan linear (SPL). Misalkan kita mempunyai SPL yang disajikan dalam bentuk matriks berikut

$$Ax = b$$

di mana A matriks berukuran $n \times n$ dan b vektor berukuran $n \times 1$, keduanya diketahui. Vektor x yang dicari adalah penyelesaian SPL dan secara teoretis diberikan oleh

$$x = A^{-1}b$$

Jadi kita dapat menggunakan perintah `inv(A)*b` untuk mendapatkan vektor `x` ini. Namun, lebih disarankan untuk menggunakan perintah pembagian kiri `A\b` karena teknik ini sudah dirancang pada MATLAB dengan algoritma terbaik. Diperhatikan contoh SPL berikut.

$$\begin{pmatrix} 2 & 3 & 5 \\ 3 & 2 & 1 \\ 5 & 6 & 4 \end{pmatrix} x = \begin{pmatrix} 28 \\ 11 \\ 33 \end{pmatrix}$$

Sistem ini sudah dirancang sedemikian rupa hingga mempunyai penyelesaian $x = \begin{pmatrix} 1 & 2 & 4 \end{pmatrix}^T$. Berikut cara menyelesaikan SPL dengan menggunakan MATLAB.

```
>> A=[2 3 5;3 2 1;5 6 4];
>> b=[28;11;33];
>> x= A\b
x =
```

```
    3  3  3  3
    3  3  3  3
    3  3  3  3
    3  3  3  3
```

3. `diag(a)` mendefinisikan matriks diagonal yang dibangun oleh vektor `a`.

4. `diag(a,k)` mendefinisikan matriks yang dibangun oleh vektor `a` di mana `k` menyatakan posisi terhadap diagonal utama, khususnya, `diag(a,0)=diag(a)`. Untuk `k` positif, vektor `a` menjadi elemen diagonal atas berjarak `k` dari diagonal utama. Sedangkan untuk `k` negatif, vektor `a` menjadi elemen diagonal bawah berjarak `k` dari diagonal utama.

```
>> a=[1 2 3];
>> diag(a,0)
ans =
    1  0  0
    0  2  0
    0  0  3
>> diag(a,1)
ans =
    0  1  0  0
    0  0  2  0
    0  0  0  3
    0  0  0  0
>> diag(a,-1)
ans =
    0  0  0  0
    1  0  0  0
    0  2  0  0
    0  0  3  0
```

5. `rand(m,n)` mendefinisikan matriks berukuran $m \times n$ yang dibangkitkan secara random oleh distribusi seragam. Sedangkan `randn(m,n)` mendefinisikan matriks berukuran $m \times n$ yang dibangkitkan secara random oleh distribusi normal. Pembahasan bilangan random yang dibangkitkan oleh distribusi probabilitas seharusnya diperoleh pada kuliah statistika matematika.

```
>> BR=rand(1,6)
BR =
```

beberapa baris atau beberapa kolom sehingga menjadi lebih kecil. Perhatikan contoh berikut.

```
>> A=magic(3)
A =

    8  1  6
    3  5  7
    4  9  2

>> B=[1;2;3]
B =

    1
    2
    3

>> C=[0 0 0 0;1 1 1 1;2 2 2 2;8 8 8 8]
C =

    0  0  0  0
    1  1  1  1
    2  2  2  2
    8  8  8  8

>> D=[A B;C]
D =

    8  1  6  1
    3  5  7  2
    4  9  2  3
    0  0  0  0
    1  1  1  1
    2  2  2  2
    8  8  8  8
```

Matriks-matriks yang akan digabungkan haruslah kompatibel berdasarkan baris atau kolom. Artinya, jika kita ingin mengembangkan kolom matriks A dengan matriks B maka banyak baris kedua matriks harus sama. Sebaliknya, jika kita ingin mengembangkan baris matriks A dengan matriks B maka banyak kolom kedua matriks harus sama. Bila tidak kompatibel maka kedua matriks tidak dapat digandengkan. Perhatikan contoh berikut.

a) Hitunglah besar vektor a dan b dalam norma Euclides.

b) Hitunglah vektor hasil operasi berikut $a+b$, $a-b$, a^2, b^2, a^2-b^2, dan $(a-b)(a+b)$. Apakah $a^2 - b^2$ sama dengan $(a-b)(a+b)$?

c) Tentukan elemen terbesar dan elemen terkecil pada kedua vektor.

d) Hitunglah $a \cdot b$ dan $a \times b$.

e) Tentukan besar sudut lancip yang dibentuk oleh kedua vektor.

f) Hitunglah nilai dari $\sum_{k=1}^{8} a_k$.

g) Hitunglah $\prod_{k=1}^{8} a_k$ di mana $\prod_{k=1}^{n} a_k = a_1 \cdot a_2 \cdot \cdots \cdot a_n$.

h) Hitunglah rata-rata bilangan yang ada pada masing-masing vektor.

i) Bila kedua vektor ini digabungkan, hitunglah rata-ratanya.

2. Buatlah *array* menaik yang terdiri dari bilangan-bilangan yang habis dibagi 3 di antara 1 dan 100.

3. Diberikan *array* a yang didefinisikan dengan menggunakan perintah `a = linspace(0,1,20);`.

   a) Berapa panjang *array* `a`

   b) Kumpulkan suku-suku genapnya, $a_2, a_4, \cdots$.

   c) Kumpulkan 5 suku terakhir.

4. Diberikan matriks dan vektor berikut.

$$W = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix} \text{ dan } b = \begin{bmatrix} 2 \\ 1 \\ 3 \\ 5 \end{bmatrix}$$

   Selesaikan dengan MATLAB masalah berikut.

   a) $W^2 + 2W + I$ dan $(W+I)^2$. Apakah hasil keduanya sama?

   b) Tentukan vektor x yang memenuhi $Ax = b$.

5. Gunakan perintah MATLAB untuk mendefinisikan matriks berukuran $n \times n$ dengan semua elemennya 0 kecuali pada baris pertama di mana semua elemennya 2. Cobakan hasil Anda untuk $n = 4, 5$.

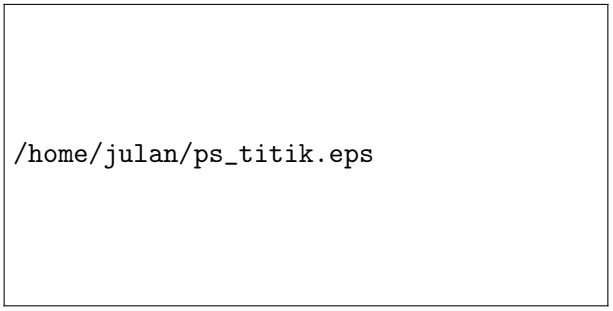
```
>> x=0:0.05:2*pi;
>>y=sin(2*pi*x);
>> plot(x,y)
>>title('Grafik fungsi y=sin(2\pi x');
```

menghasilkan grafik pada panel kiri Gambar 1.3. Terlihat domainnya tidak persis sama dengan interval $[0, 2\pi]$. Untuk membatasi grafik pada domain yang dimaksud digunakan perintah `xlim` seperti berikut.

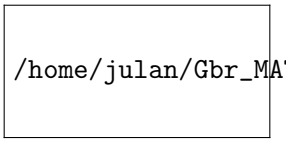
```
>> xlim([0 2*pi]);
>>xlabel('x'); ylabel('y');
```

Hasilnya ditampilkan pada Gambar 1.3. Penjelasannya sebagai berikut.

- Pada baris pertama, `x=0:0.05:2*pi;` berarti kita mempersiapkan data berupa *array* $x = 0, 0.05, 0.1, 0.15, \cdots$ sampai bilangan yang paling dekat dengan 2π. Lihat kembali definisi operator kolon (:) sebelumnya. Dapat juga menggunakan perintah `linspace`.
- Pada baris kedua, kita mempersiapkan data berupa *array* `y` yang berupa nilai fungsi pada $y = \sin(2\pi x)$ untuk x yang sudah didefinisikan pada baris sebelumnya. Jadi `x` dan `y` merupakan pasangan *array* yang mempunyai panjang sama.
- `plot(x,y)` meminta MATLAB untuk menggambar pasangan data `x` dan `y` pada bidang koordinat. Gambar ini berupa titik-titik (noktah) yang dihubungkan oleh garis patah-patah. Kemulusan grafik yang diperoleh tergantung dari kerapatan *array* x yang didefinisikan.
- `title('Grafik fungsi y=sin(2\pi x')` memberikan judul grafik
- `xlim([0 2*pi])` untuk membatasi range sumbu x



Gambar 1.2: Plot pasangan titik



Gambar 1.5: Grafik beberapa fungsi pada bidang koordinat berbeda

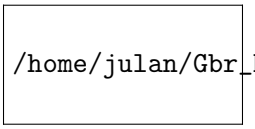
```
>>subplot(2,2,1)
>>y1=cos(pi*x);
>>plot(x,y1)
>>xlim([0 2*pi])
>>title('grafik fungsi y = sin \pi x')
>>subplot(2,2,2)
>>y2=cos(2*pi*x);
>>plot(x,y2)
>>xlim([0 2*pi])
>>title('grafik fungsi y = sin 2\pi x')
>>subplot(2,2,3)
>>y3=cos(3*pi*x);
>>plot(x,y3)
>>xlim([0 2*pi])
>>title('grafik fungsi y = sin 3\pi x')
>>subplot(2,2,4)
>>y4=cos(4*pi*x);
>>plot(x,y4)
>>xlim([0 2*pi])
>>title('grafik fungsi y = sin 4\pi x')
```

Hasilnya ditunjukkan pada Gambar 1.5.

Bila keempat grafik ini mau ditempatkan pada satu bidang koordinat, kita dapat menggunakan dua cara, yaitu menggunakan perintah `plot` secara paralel atau menggunakan perintah `hold on`. Berikut contoh penggunaan plot secara paralel.

```
>>plot(x,y1,x,y2,x,y3)
>>xlim([0 2*pi])
```

Hasilnya ditampilkan pada panel kiri Gambar 1.6. Untuk penggunaan `hold on` semestinya ditutup dengan perintah `hold off`. Perintah `hold on` pada prinsipnya meminta MATLAB



Gambar 1.7: Grafik dengan aksesorinya

```
r  merah
c  cyan
m  magenta (pink)
y  kuning
```

Bentuk noktah, yaitu titik pasang atau data yang bersesuaian pada bidang koordinat, dapat diberikan tanda-tanda berikut.

```
.  titik           o  lingkaran
x  tanda silang    +  tanda tambah
*  tanda asterik   s  tanda kotak
d  tanda diamon    v  segitiga bawah
^  segitas atas    <  segitiga kiri
>  segitiga kanan  p  pentagram
h  heksagram
```

Ada 4 jenis garis penghubung noktah pembangun grafik, yaitu:

```
-   garis padat (tidak putus)
:   titik-titik atau dot
-.  gabungan garis putus-putus dan dot
--  garis putus-putus
```

Perintah berikut ini

```
>> x=-6:0.2:6;
>> y=1/sqrt(pi)*exp(-x.^2/2);
>> plot(x,y,'dr:')
```

menghasilkan grafik seperti ditunjukkan pada Gambar 1.7. Pada perintah `plot(x,y,'dr:')` ada tambahan opsi `'dr:'`. Terdapat 3 paramater di sini, yaitu `d` untuk tanda diamond, `r` untuk warna merah dan `:` untuk garis penghubung putus-putus.

Cara kedua adalah dengan menggunakan jendela editor grafik. Caranya pada jendela grafik yang aktif, tekan tanda panah yang terdapat pada barisan shortcut seperti ditunjukkan

Soal-soal untuk Latihan

1. Gambarkan pada satu bidang koordinat grafik fungsi $y = \sin(\pi x)$ dan $y = e - x$ pada interval $[0, 1]$. Perkirakanlah titik potong kedua grafik.
2. Gambarkan pada satu bidang koordinat grafik fungsi $y = x^{\alpha}$ untuk $\alpha = 0.5, 1, 2$ pada interval $[0, 1]$. Kemudian, coba untuk $\alpha = -2, -1, -0.5, 0.5, 1, 2$. Apa yang dapat Anda simpulkan dari eksperimen ini ?
3. Fungsi kepadatan peluang (fkp) distribusi normal dengan mean μ dan varians σ^2 diberikan oleh
$$f(x) := \frac{1}{\sigma\sqrt{\pi}} e^{-\left(\frac{x-\mu}{\sigma}\right)^2}, -\infty < x < \infty.$$
Gambarkan beberapa grafik fkp ini pada satu bidang koordinat untuk kombinasi $\mu = -0.5, 0, 0.5$ dan $\sigma = 0.5, 1, 2$ pada interval $-6 < x < 6$. Apa yang dapat Anda tarik kesimpulan dari eksperimen ini?

Selain dari grafik 2 dimensi, terdapat pula grafik 3 dimensi yang dapat digambarkan pada MATLAB. Banyak sekali bentuk-bentuk visualisasi grafis yang dapat dihasilkan oleh MATLAB. Karena pembahasan di sini hanya pengenalan MATLAB maka fasiltas grafis lainnya dapat dipelajari sendiri pada MATLAB.

1.5 Pemrograman pada MATLAB

Untuk kebutuhan standar, MATLAB sudah menyediakan banyak fungsi bawaan (*built-in*) maupun berbentuk m-file. Tetapi untuk keperluan tertentu kita harus menulis kode tersendiri. Dalam penulisan kode ini kita dapat menggunakan fungsi-fungsi yang telah tersedia pada MATLAB. Bahasa atau kode MATLAB ini ditulis pada jendela editor. Setiap algoritma pada metode numerik seharusnya dapat diimplementasikan pada komputer, salah satunya dengan menulis m-file pada MATLAB.

m-file

Bahasa pemrograman MATLAB yang ditulis pada jendela editor/debugger disebut m-file. Terdapat tiga macam m-file yaitu :

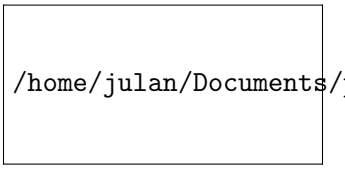
- **script file**, yaitu m-file yang tidak memerlukan variabel masukan dan variabel keluaran. Program ini dapat dijalankan dari *command window* dengan cara mengetik nama filenya.

- baris keempat perintah grid dimaksudkan untuk membuat garis skala atau kotak persegi pada bidang gambar.

Selanjutnya pada *command window*, diketik

```
>> file1
```

kemudian tekan enter maka akan dihasilkan grafik seperti pada Gambar 1.10 berikut.



Gambar 1.10: Hasil output script-file

Misalkan kita ingin membuat suatu perintah dan kode MATLAB untuk menghitung rata-rata dan standar deviasi suatu data yang disajikan dalam *array* **x**. Di sini kita membutuhkan variabel masukan yaitu data **x** dan dua variabel keluaran yaitu **mean** untuk rata-rata dan **stdev** untuk standar deviasi. Untuk menyusun m-file ini kita harus dapat mengimplementasikan formula statistika berikut untuk rata-rata dan standar deviasi ke dalam kode MATLAB, yaitu

$$\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i \text{ dan } s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}}.$$

Untuk rumus standar deviasi sampel di mana pembaginya adalah $n-1$, silakan dilakukan modifikasi sendiri. Berikut function-file untuk menghitung hasil kedua rumus tersebut.

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
```

Ini adalah sebuah function file dengan nama **stat**. Setelah file disimpan, katakan namanya stat.m maka kita dapat menjalankan program ini untuk data x yang diberikan sebagai variabel masukan.

```
>> x =rand(1,6)
x =
```

```
-0.6277 -6.3723
```

Ini berarti persamaan kuadrat tersebut mempunyai akar-akar $x_1 = -0.6277$ dan $x_2 = -6.3723$.

Operator logika

Beberapa relasi dan operator logika pada MATLAB diberikan pada tabel berikut.

TABEL 1.5: Relasi dan operator logika

Simbol	Nama relasi/operator
<	kurang dari
<=	kurang dari atau sama dengan
>	lebih dari
>=	lebih dari atau sama dengan
==	sama dengan
~=	tidak sama dengan
|	atau (disjungsi)
&	dan (konjungsi)
~	tidak (ingkaran)

Pengendali alur

Beberapa perintah untuk pengendali alur (*control loop*), antara lain adalah **if**, **switch** and **case**, **for**, **while**, **continue**, **break**.

if - elseif - else - end

Pernyataan **if** akan mengevaluasi suatu kondisi atau syarat, biasanya berupa pernyataan logika. Bila syarat ini dipenuhi, serangkaian perintah dikerjakan atau dieksekusi. Pilihan **elseif** dan **else** menyediakan alternatif lain bila kondisi ini tidak dipenuhi. Kata **end** yang berpasangan dengan **if** digunakan untuk mengakhiri tugas jika semua kondisi sudah dipenuhi. Format umum penggunaan **if-elseif-else-end** adalah sebagai berikut.

```
if syarat 1 dipenuhi
```

```
    BAIK
>> konversi
masukkan nilai x: 11
    SALAH DATA, nilai x harus di dalam [0,10]
```

Jadi setiap kita memberikan nilai numerik untuk x maka akan muncul keluaran berupa kategori KURANG, CUKUP, BAIK, atau SALAH DATA. Satu lagi contoh penggunaan pengendali alur **if** pada pendefinisian fungsi cabang

$$f(x) = \begin{cases} 1 & \text{jika } x > 0 \\ 0 & \text{jika } x = 0 \\ -1 & \text{jika } x < 0. \end{cases}$$

Tulis dan simpan kode berikut dengan nama file **tanda.m**. Dalam matematika, fungsi ini biasanya disebut fungsi *signum* atau fungsi tanda.

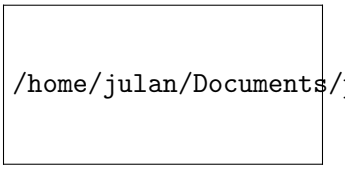
```
function y = tanda(x)
if x>0
    y = 1;
    elseif x == 0
    y = 0;
    else
    y = -1;
end
```

Kita dapat mengecek pada *command window* bahwa kode ini mendefinisikan fungsi cabang yang dimaksud.

```
>> y = tanda(2)
   y = 1
>> y = tanda(0)
   y = 0
>> y = tanda(-0.3)
   y = -1
```

Untuk melihat grafik fungsi ini, coba diberikan perintah berikut.

```
>>x=-1:0.05:1;y=tanda(x);
>>plot(x,y,'.')
```



Gambar 1.12: Keluaran swicth case

for - end

Kendali alur **for** digunakan untuk mengulangi serangkaian statemen yang banyaknya telah ditentukan diawal. Kita berikan contoh pemakaian pengendali alur ini untuk membangun matriks Hilbert H berukuran $m \times n$ yang didefinisikan sebagai $(H)_{i,j} := \frac{1}{i+j}$.

```
function H=hilb(m,n)
for i = 1:n
    for j = 1:m
    H(i,j)=1/(i+j);
    end
end
```

Setelah disimpan dengan nama **hilb.m** maka perintah hilb(4,5) akan menghasilkan matriks Hilbert berukuran 4×5, yaitu

```
>> H=hilb(4,5)
H =

    0.5000 0.3333 0.2500 0.2000 0.3333
    0.2500 0.2000 0.1667 0.2500 0.2000
    0.1667 0.1429 0.2000 0.1667 0.1429
    0.1250 0.1667 0.1429 0.1250 0.1111
```

Pengendali alur **for** dan **if** dapat digunakan secara bersamaan. Misalkan kita akan mendefinisikan fungsi berikut dan menggambar grafiknya.

$$f(x) := \begin{cases} x \sin \frac{1}{x} & \text{jika } x \neq 0 \\ 0 & \text{jika } x = 0. \end{cases}$$

Untuk lebih menarik lagi, kita gambarkan grafik fungsi $y_1 = x$ dan$y_2 = -x$ pada bidang koordinat yang sama.

2, hasilnya dibagi lagi dengan 2, dan seterusnya sampai berhenti bilamana hasil baginya kurang dari TOL. Kita ingin tahu bilangan pertama yang kurang dari TOL ini dan berapa kali proses pembagian dengan 2 dilakukan.

```
function [hasil,iter]=bagi_phiby2(TOL)
q = pi;
it = 0;
while q >= TOL;

    q = q/2;
    it = it+1;

end
hasil=q;
iter = it;
```

Setelah kode ini disimpan dengan nama **bagi_phiby2.m**, kita dapat memperoleh hasil pada *command window*

```
>>[hasil,iter]=bagi_phiby2(0.0013)
hasil =

    7.669903939428206e-004

iter =

    12
```

Ini artinya bila ditetapkan TOL=0.0013 maka bilangan pertama yang kurang dari 0.0013 setelah π dibagi terus-menerus dengan 2 adalah 0.00076699, dengan banyak iterasi 12 langkah.

Untuk sementara hanya ini materi pengenalan MATLAB yang dapat disampaikan. Seiring berjalannya waktu dan tuntutan implementasi numerik maka keterampilan MATLAB akan semakin meningkat.

Soal-soal untuk Latihan

1. Buat m-file dengan parameter masukan adalah koefisien-koefisien pada fungsi kuadrat

   $$y = ax^2 + bx + c.$$

   dan sebuah bilangan real x_0 sebarang dengan keluarannya adalah

2 PENDAHULUAN KOMPUTASI SAINTIFIK

2.1 Motivasi

Pemodelan matematika merupakan suatu proses di mana permasalahan pada dunia riil disajikan dalam bentuk permasalahan matematika, seperti sekumpulan persamaan baik linear maupun taklinear, persamaan diferensial yang memuat masalah nilai awal dan syarat batas, persamaan integral, masalah optimisasi dan lain sebagainya. Jadi model matematika menggambarkan sistem dunia nyata dalam bentuk persamaan matematika. Persamaan matematika tersebut menyajikan relasi antara variabel dan parameter yang terlibat dalam sistem. Selanjutnya permasalahan matematika tersebut ditentukan penyelesaiannya. Penyelesaian yang diperoleh biasanya divalidasi melalui data nyata semisal hasil pengukuran, sebelum pada akhirnya digunakan untuk menyelesaikan masalah dunia nyata tersebut.

Model matematika yang berasal dari berbagai masalah keteknikan dan ilmu pengetahuan lainnya menuntut adanya suatu prosedur atau metode yang digunakan untuk menemukan penyelesaiannya. Salah satu cara untuk menemukan penyelesaian tersebut adalah dengan metode analitik. Metode ini digunakan untuk menemukan penyelesaian eksak. Metode integral parsial dalam menyelesaikan hitung integral, metode separasi variabel dalam persamaan diferensial adalah beberapa metode analitik untuk mendapatkan penyelesaian eksak. Tetapi dalam bidang aplikasi matematika metode analitik sulit diterapkan karena bentuk masalah matematika yang dihadapi lebih kompleks dan tidak memenuhi persyaratan teoretis, misalnya fungsi yang dibutuhkan hanya tersedia dalam bentuk data numerik, tidak tersedia dalam bentuk eksplisit seperti banyak terdapat dalam buku pelajaran. Oleh karena itu, dalam kasus di mana penyelesaian eksak tidak dapat diperoleh maka dibutuhkan metode numerik untuk mengaproksimasi penyelesaian eksak tersebut. Sebagai motivasi ada baiknya kita perhatikan contoh berikut.

Contoh 2.1. Sebuah atap berbentuk seng gelombang terbuat dari plat tipis rata yang dipres seperti terlihat pada Gambar 2.1 berikut.

Kalau pada kedua contoh di atas kita berhadapan dengan formula matematika eksplisit maka contoh berikut hanya menyediakan data numerik tanpa diketahui formula eksplisit fungsi.

Contoh 2.3. Misalkan sebuah mobil berjalan pada jalan tol. Jarak tempuh dan kecepatan mobil tersebut dicatat pada beberapa titik waktu tertentu, dengan data sebagai berikut.

Waktu (detik)	0	3	6	8	12
Jarak (m)	0	75	130	210	325
Kecepatan (m/det)	25	29	31	27	26

Pertanyaan seperti di mana posisi mobil dan berapa kecepatannya pada saat $t = 10$ detik, atau apakah mobil pernah melebihi kecepatan yang diperbolehkan di jalan tol, misalnya 80km/jam, semuanya tidak dapat dijawab secara eksak karena rumus fungsi jarak dan fungsi kecepatan tidak tersedia. Hanya cara aproksimasi sajalah yang digunakan untuk menyelesaikan masalah ini.

Dari ketiga contoh sederhana ini kiranya jelas bahwa metode numerik sangat dibutuhkan khususnya pada matematika terapan (*applied mathematics*). Semakin kompleks permasalahan yang dihadapi semakin rumit bentuk model matematikanya, dan tentunya semakin sulit model tersebut diselesaikan. Sebagai contoh, model yang berasal dari masalah kardiovaskular memuat puluhan persamaan diferensial taklinear dengan puluhan parameter yang harus diestimasi melalui data hasil observasi. Dibutuhkan algoritma dan teori matematika untuk menciptakan metode yang baik. Secara tradisional, perancangan dan analisis terhadap algoritma untuk pemecahan masalah dalam sains dan keteknikan dikenal dengan analisis numerik. Pada era pra komputer, pekerjaan kalkulasi untuk mengimplementasikan sebuah algoritma rumit menghabiskan waktu yang sangat banyak sehingga merupakan pekerjaan sangat yang membosankan. Belum lagi hasil yang diperoleh sangat rentan terhadap kesalahan hitung yang disebabkan oleh manusia (*human error*). Namun seiring kemajuan di bidang teknologi komputer, analisis numerik berkembang menjadi bidang tersendiri yang dewasa ini disebut **komputasi saintifik** (*scientific computing*). Menurut Golub dan Ortega (1992) dalam [**?**]:

> *Scientific computing is the collection of tools, techniques, and theories required to solve on a computer mathematical models of problems in science and engineering.*

Jadi komputasi saintifik melibatkan perangkat keras berupa komputer, latar belakang teoretis berupa teori matematika, metode dan algoritma serta perangkat lunak yang digunakan untuk memecahkan masalah berbantuan komputer. Tujuan utama dari komputasi

a) Pemodelan
   Beberapa keadaan sesungguhnya dari unsur-unsur masalah yang sedang dipelajari kemungkinan sudah disederhanakan atau malah dihilangkan. Misalnya faktor gesekan, kekentalan atau viskositas diasumsikan konstan atau bahkan barangkali dianggap tidak ada.

b) Pengukuran empiris
   Instrumen laboratorium yang digunakan untuk melakukan pengukuran memiliki presisi terbatas sehingga berpengaruh pada hasil pengukuran. Akurasi pengukuran juga dapat disebabkan oleh kecilnya sampel, atau oleh nois atau bias sistemik. Sebagai contoh, pengukuran konstanta gravitasi atau konstanta Planck hanya memiliki akurasi sekitar 8 atau 9 digit.

c) Komputasi terdahulu
   Data yang dimasukkan dalam proses komputasi terkadang hasil yang telah diperoleh pada step sebelumnya yang juga hasil aproksimasi. Jadi, ini melakukan aproksimasi dengan data hasil aproksimasi.

2. Selama komputasi mencakup kesalahan dalam

a) Diskretisasi atau pemotongan
   Dalam analisis matematika, teori dikembangkan dalam bentuk fungsi kontinu dan proses takhingga. Dalam penerapannya tidak mungkin kita dapat mengimplementasikan proses takhingga ini termasuk jika menggunakan komputer. Sebagai gantinya, bentuk kontinu biasanya dilakukan diskretisasi dan bentuk takhingga seperti deret takberujung 'diganti' dengan jumlah parsial beberapa suku pertamanya saja.

b) Pembulatan
   Representasi bilangan pada komputer dan operasi aritmatika pada bilangan-bilangan ini pada umumnya tidak eksak karena sudah mengalami pembulatan otomatis oleh komputer. Oleh karena itu, bilangan keluaran (*output*) hasil komputasi sudah disertai kesalahan.

Akurasi hasil akhir suatu komputasi dipengaruhi oleh semua kesalahan tersebut. Karena banyaknya iterasi dalam algoritma dapat mencapai ratusan atau bahkan ribuan kali, maka akumulasi kesalahan dapat tak terkendali walaupun hanya terjadi kesalahan kecil pada setiap iterasinya. Oleh karena itu akurasi dan stabilitas algoritma perlu mendapat perhatian serius dalam melakukan aproksimasi. Stabilitas merujuk pada tidak rentan perubahan keluaran terhadap gangguan kecil pada masukan.

Definisi 2.1. Misalkan p^*adalah nilai yang digunakan untuk mengaproksimasi nilai eksak p maka dua macam kesalahan berikut sering digunakan dalam mengukur kualitas aproksimasi, yaitu

$$\begin{aligned} \text{kesalahan mutlak} &:= |p^* - p| \\ \text{kesalahan relatif} &:= \frac{|p^* - p|}{|p|} \end{aligned}$$

asalkan $p \neq 0$ untuk kesalahan relatif. Terkadang kesalahan relatif disajikan dalam bentuk persentase.

Secara ekuivalen,

$$\text{nilai aproksimasi=(nilai eksak)(1+kesalahan relatif).}$$

Contoh 2.5. Mana yang lebih signifikan:

1. kekurangan 1 kilo dari 10 kilo beras, atau
2. kekurangan 5 kilo dari 100 kilo beras.

Penyelesaian. Kita bandingkan kesalahan relatif kedua kasus ini, yaitu $\frac{1}{10}$ atau 10% untuk kasus pertama, dan $\frac{5}{100}$ atau 5% untuk kasus kedua. Dilihat dari kesalahan relatif ini maka kehilangan pada kasus pertama lebih signifikan daripada kehilangan pada kasus kedua. ■

2.3 Representasi Bilangan pada Komputer

Perhitungan yang dilakukan oleh komputer atau kalkulator berbeda dengan perhitungan dalam kalkulus atau aljabar. Ketika kita menghitung nilai $\sqrt{3}$ pada kalkulator atau komputer maka akan diperoleh angka desimal

1.73205080756888.

Hasil ini bukanlah nilai eksak $\sqrt{3}$ tetapi hanya sebagai pendekatan, lebih tepat ditulis sebagai $\sqrt{3} \approx 1.73205080756888$. Hal ini disebabkan bilangan irasional tidak dapat disajikan dalam bentuk desimal digit terbatas, sedangkan kalkulator ataupun komputer hanya mampu menyajikan bilangan desimal dengan banyak digit berhingga.

Dikarenakan sifat inilah maka kalkulasi dengan komputer $\sqrt{3} \times \sqrt{3}$ hasilnya tidak persis sama dengan 3 namun terdapat kekeliruan walaupun sangat kecil. Kekeliruan yang diberikan oleh komputer atau kalkulator ketika kita melakukan kalkulasi bilangan real disebut kekeliruan pembulatan (*rounding error*). Kesalahan pembulatan ini disebabkan oleh sistem representasi bilangan pada komputer.

- c diperoleh dari 11 bit berikutnya, yang menunjukkan eksponen. Basis untuk eksponen ini adalah 2,
- f diperoleh dari 52 bit terakhir yang merupakan bagian pecahan binair, disebut juga **mantisa**.

Dengan format ini, nilai c berjalan pada interval $0 < \mathbf{c} < 2047$ dan eksponen yang dapat dijangkau berada dalam interval $(-1023, 1024)$. Bentuk representasi lainnya dan batasan parameter yang dapat dijangkau, lihat [**?**].

Contoh 2.6. Misalkan suatu bilangan mesin komputer direpresentasikan sebagai berikut

0 10000000011 1010100100010000000000000000000000000000000000000000000000

Tentukan bilangan real apa yang sedang disajikan oleh notasi ini.

Penyelesaian. Bit paling kiri adalah $s = 0$. Sebelas bit berikutnya 10000000011 memberikan eksponen, yaitu ekuivalen dengan bilangan desimal

$$c = 1 \times 2^{10} + 0 \times 10^9 + \cdots + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 1027$$

Selanjutnya, 52 bit terakhir sebagai mantisa berarti

$$f = 1 \times \left(\frac{1}{2}\right)^1 + 1 \times \left(\frac{1}{2}\right)^3 + 1 \times \left(\frac{1}{2}\right)^5 + 1 \times \left(\frac{1}{2}\right)^8 + 1 \times \left(\frac{1}{2}\right)^{12}$$

Jadi bilangan mesin ini sesungguhnya menyajikan bilangan desimal berikut, yaitu

$$\begin{aligned}(-1)^s \times 2^{\mathbf{c}-1023} \times (1+f) &= (-1)^0 \times 2^{1027-1023}(1+f) \\ &= 27.56640625\end{aligned}$$

Sebaliknya, contoh berikut menerjemahkan dari lambang desimal ke dalam bentuk bahasa mesin.

Contoh 2.7. Sajikan bilangan 32.75 dalam bahasa mesin seperti di atas.

Penyelesaian. Pertama bilangan 32.75 disajikan dalam bentuk basis 2, yaitu

$$\begin{aligned}32.75 &= (-1)^0 \times 2^5 \times \left(1 + \frac{0.75}{2^5}\right) \\ &= (-1)^0 \times 2^5 \times \left(1 + \left(\frac{(1/2) + (1/4)}{2^5}\right)\right) \\ &= (-1)^0 \times 2^5 \times \left(1 + \left(\frac{1}{2}\right)^6 + \left(\frac{1}{2}\right)^7\right)\end{aligned}$$

2.3.2 Aturan Pemotongan Digit Bilangan Desimal

Penggunaan sistem digital binair menjadi sulit ketika hanya terdapat berhingga bilangan mesin digunakan untuk menyajikan semua bilangan real. Faktanya, komputer hanya memiliki jumlah digit yang sangat terbatas, misalnya sistem 16-bit, 32-bit atau 64-bit. Jadi tidak semua bilangan real dapat disajikan dalam bentuk titik mengambang oleh komputer. Jadi bilangan dengan panjang digit melebihi presisi $t = 53$ harus dibulatkan atau dipotong. Aproksimasi titik mengambang suatu bilangan real x ditulis $\text{fl}(x)$. Proses pemilihan $\text{fl}(x)$ untuk mengaproksimasi x disebut **pembulatan** (*rounding*), dan kesalahan yang disebabkan oleh pembulatan ini disebut kesalahan pembulatan (*rounding error*) atau (*roundoff error*).

Dua metode yang sering digunakan dalam pembulatan adalah

- Pemotongan (*chooping*): ekspansi x dalam basis β dibuang setelah digit ke $t-1$. Aturan ini disebut juga **pembulatan menuju nol** sebab $\text{fl}(x)$ adalah bilangan titik mengambang berikutnya yang menuju nol dari x.
- Pembulatan terdekat (*rounding*): $\text{fl}(x)$ adalah bilangan titik mengambang terdekat dengan x. Dalam kasus seimbang, diambil bilangan titik mengambang yang digit terakhirnya genap. Oleh karena itu aturan ini juga disebut **pembulatan ke genap**.

Pembulatan ke bilangan terdekat adalah paling akurat dan merupakan aturan standar yang digunakan oleh sistem IEEE DP.

Contoh 2.8. Berikut ini contoh diambil dari Heath (1997) dalam [**?**] tentang penggunaan aturan pemotongan dan pembulatan bilangan desimal menjadi 2 digit.

TABEL 2.1: Contoh aturan pemotongan dan pembulatan

bilangan	pemotongan	pembulatan	bilangan	pemotongan	pembulatan
1.649	1.6	1.6	1.749	1.7	1.7
1.650	1.6	1.6	1.750	1.7	1.8
1.651	1.6	1.7	1.751	1.7	1.8
1.699	1.6	1.7	1.799	1.7	1.8

Perhatikan bahwa 1.650 dan 1.750 adalah kasus seimbang. Pada kasus pertama 1.650 dibulatkan ke 1.6 bukan ke 1.7 karena diambil digit terakhir genap, sedangkan pada 1.750 dibulatkan ke 1.8 tidak ke 1.7 karena yang harus diambil digit terakhir genap. Aturan seperti inilah yang terkadang menyebabkan akurasi hasil akhir perhitungan oleh komputer

yang 'jauh' lebih kecil dari presisi mesin ϵ_{mach} , biasa disimbolkan oleh $\epsilon \ll \epsilon_{mach}$ maka dalam bentuk titik mengambang diperoleh $(1+\varepsilon)-(1-\varepsilon) = 1-1 = 0$, padahal secara matematis hasilnya adalah 2ε.

2.4 Order Konvergensi

Mengaproksimasi suatu nilai eksak x biasanya dilakukan dengan membangun barisan aproksimasi (x_n) di mana $\lim_{n\to\infty} x_n = x$. Terdapat banyak kemungkinan barisan (x_n) yang berbeda tetapi sama-sama konvergen ke x. Perbedaannya adalah terletak dari kecepatan konvergensi. Definisi berikut memberikan karakterisasi kekonvergen barisan ke limitnya.

Definisi 2.2. Misalkan $(\alpha_n)_{n=1}^{\infty}$ konvergen ke bilangan α untuk n membesar. Jika terdapat konstanta positif p dan K sehingga

$$|\alpha - \alpha_n| \leq \frac{K}{n^p} \text{ untuk semua } n \text{ besar}, \tag{2.3}$$

maka kita katakan bahwa $(\alpha_n)_{n=1}^{\infty}$ konvergen ke α dengan **rate** atau **order kekonvergenan** $\mathcal{O}\left(\frac{1}{n^p}\right)$ (dibaca: big-oh dari $\frac{1}{n^p}$).

Dengan definisi ini, relasi $\alpha_n = \alpha + \mathcal{O}\left(\frac{1}{n^p}\right)$ menyatakan bahwa $\alpha_n \to \alpha$ dengan rate $\frac{1}{n^p}$. Dalam membangun barisan aproksimasi $(\alpha_n)_{n=1}^{\infty}$ kita seharus mengetahui nilai terbesar p sehingga $\alpha_n = \alpha + \mathcal{O}\left(\frac{1}{n^p}\right)$. Semakin tinggi p semakin cepat kekonvergenannya.

Contoh 2.10. Diberikan dua barisan (x_n) dan (y_n) sebagai berikut

$$x_n := \frac{1}{n},\ y_n := \frac{3}{n^2}.$$

Kedua barisan ini sama-sama konvergen ke 0, namun barisan (y_n) lebih cepat daripada (x_n). Dalam hal ini kekonvergenan (x_n) mempunyai order satu dan (y_n) mempunyai order dua, sebab

$$|x_n - 0| = \mathcal{O}\left(n^{-1}\right) \text{ dan } |y_n - 0| = \mathcal{O}\left(n^{-2}\right),$$

yakni dengan mengambil $K = 1$ untuk (x_n) dan $K = 3$ untuk (y_n). Walaupun pada awalnya nilai (x_n) lebih kecil dari nilai (y_n) khususnya ketika $n = 1$ dan 2, namun selanjutnya barisan (y_n) lebih cepat mengecil menuju nol daripada barisan (x_n). Perhatikan peluruhan (*decay*) kedua barisan tersebut diberikan pada Gambar 2.2.

Walau secara matematika teoretis $(1+eps/3)-(1-eps/4) = eps/12 \neq 0$ namun hasil yang diberikan oleh MATLAB adalah 0. Padahal bilangan $eps/12$ masih jauh di atas *underflow*. Fakta seperti ini seharusnya diperhitungkan dalam melakukan komputasi numerik, khususnya dengan MATLAB.

Aproksimasi faktorial dengan formula Stirling

Faktorial suatu bilangan bulat positif n adalah $n! := n(n-1)(n-2)\cdots(3)(2)(1)$. Sebelum adanya komputer digital, pekerjaan menghitung nilai faktorial bilangan besar merupakan pekerjaan yang sangat sulit karena harus melakukan perkalian berulang bilangan-bilangan sangat besar. Sebagai gantinya orang menggunakan formula Stirling untuk mengaproksimasi faktorial. Salah satu formula Stirling yang sering digunakan untuk aproksimasi adalah

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

di mana e adalah bilangan natural. Pada kesempatan ini formula diimplementasikan untuk mengaproksimasi faktorial, berikut kasalahan mutlak dan kesalahan relatif yang ditimbulkan. Untuk keperluan ini disusun m-file MATLAB berikut.

```
n = 15;
eks = zeros(n,1);ap=zeros(n,1);
err=zeros(n,1); err_re=zeros(n,1);
for k = 1:n

    eks(k) = factorial(k);
    ap(k) = sqrt(2*pi*k)*(k/exp(1))^k;
    err(k)=abs(ap(k)-eks(k));
    err_re(k)=err(k)/eks(k);

end
[eks ap err err_re]
```

Misalkan m-file ini disimpan dengan nama `stirling.m`. Selanjutnya dieksekusi dengan memberikan perintah pada *command window* dengan

```
>>stirling
```

maka diperoleh keluaran seperti dirangkum pada Tabel 2.2.

Walaupun kesalahan mutlak aproksimasi semakin membesar tetapi kesalahan relatifnya semakin mengecil. Karena itu sangat masuk akal formula Stirling ini cocok digunakan

TABEL 2.3: Aproksimasi e dengan limit

k	$\left(1+\frac{1}{n}\right)^n, n=10^k$	Kesalahan mutlak
1	2.59374246010000	0.12453936835904
2	2.70481382942153	0.01346799903752
3	2.71692393223559	0.00135789622345
4	2.71814592682493	0.00013590163412
5	2.71826823719230	0.00001359126675
6	2.71828046909575	0.00000135936329
7	2.71828169413208	0.00000013432696
8	2.71828179834736	0.00000003011169
9	2.71828205201156	0.00000022355251
10	2.71828205323479	0.00000022477574
11	2.71828205335711	0.00000022489806
12	2.71852349603724	0.00024166757819
13	2.71611003408690	0.00217179437214
14	2.71611003408702	0.00217179437202
15	3.03503520654926	0.31675337809022
16	1.00000000000000	1.71828182845905
17	1.00000000000000	1.71828182845905

```
for k = 1: m

    n(k) = 10^k;
    ap(k) = (1+1/n(k))^n(k);
    err(k)=abs(ap(k)-exp(1))

end
[ap err]
```

Mari kita simpan m-file ini dengan nama `eksponen1.m`. Keluaran MATLAB disajikan pada Tabel 2.3. Bila diperhatikan secara saksama, formula ini hanya konsisten sampai dengan $k = 8$, setelah itu aproksimasi semakin menjauh dari nilai yang diharapkan. Barisan aproksimasi ini sudah menunjukkan kedivergenan. Faktor utama yang menyebabkan hal ini adalah kesalahan pembulatan pada suku $1/n$ untuk n sangat besar, juga dengan epsilon mesin. Diperhatikan bahwa dimulai dari $k = 16$ kelihatannya nilai $1/n$ di mana $n = 10^k$ sudah dianggap nol. Dari eksperimen ini dapatlah dimaklumi bahwa benar secara teoretis belum tentu benar dalam implementasi numerik.

TABEL 2.4: Aproksimasi e dengan deret Taylor

n	P_n	Kesalahan mutlak
1	2.00000000000000	0.71828182845905
2	2.50000000000000	0.21828182845905
3	2.66666666666667	0.05161516179238
4	2.70833333333333	0.00994849512571
5	2.71666666666667	0.00161516179238
6	2.71805555555556	0.00022627290349
7	2.71825396825397	0.00002786020508
8	2.71827876984127	0.00000305861778
9	2.71828152557319	0.00000030288585
10	2.71828180114638	0.00000002731266
11	2.71828182619849	0.00000000226055
12	2.71828182828617	0.00000000017288
13	2.71828182844676	0.00000000001229
14	2.71828182845823	0.00000000000082
15	2.71828182845899	0.00000000000005
16	2.71828182845904	0.00000000000000
17	2.71828182845905	0

```
    n(k) = 10^(-k);
    ap(k) = (exp(n(k))-1)/n(k);
    err(k)=abs(ap(k)-1);

end
[ap err]
```

Hasilnya diberikan pada tabel 2.5. Implementasi numerik kembali menunjukkan ketidakkonsistenan dengan teori matematika yang sesungguhnya. Dimulai dari $k = 12$ dan seterusnya aproksimasi menyimpang dari yang diharapkan. Tidak ada yang aneh dengan hasil ini karena komputer sebagai ciptaan manusia memiliki keterbatasan khususnya dalam penyajian bilangan real. Namun demikian teknologi komputer keuntungannya jauh lebih banyak daripada kelemahan yang hanya sedikit ini. Untuk implementasi komputasi normal, umumnya tidak ada masalah. Contoh di atas adalah kasus-kasus ekstrem. Apa pun hasil yang diberikan komputer harus divalidasi dengan teori matematika yang berlaku, tidak langsung diterima mentah-mentah. Penguasaan latar belakang matematika teoretis menjadi keharusan dalam mendalami keterampilan komputasi. Khusus untuk komputasi

yang berkaitan dengan pemodelan matematika bagi masalah dunia nyata membutuhkan validasi data nyata sebelum model tersebut digunakan untuk prediksi, dan lain-lain.

Soal-soal untuk Latihan

1. Mengapa perlu adanya teori aproksimasi, khususnya metoda numerik? Jelaskan dan berikan ilustrasinya.
2. Dalam banyak kasus, untuk menghitung nilai sesungguhnya orang tidak melakukan pengukuran langsung terhadap objek atau benda tersebut tetapi hanya melalui model yang sedapat mungkin mirip dengan fakta sesungguhnya. Hasil akhir yang diperoleh dipastikan memuat kesalahan. Sebutkan sumber kesalahan yang mungkin terjadi dalam proses perhitungan yang dimaksud. Berikan contoh ilustrasinya selain dari menghitung luas permukaan bumi seperti pada contoh sebelumnya.
3. Apa yang dimaksud dengan istilah presisi dan akurasi dalam aproksimasi numerik. Berikan ilustrasi untuk menjelaskan kedua istilah ini.
4. Bilangan real apa yang disajikan oleh bentuk binair berikut.

   $$0\,10000000011\,1010100100010000000000000000000000000000000000000000000000$$

5. Tuliskan bentuk binair sistem 64-bit untuk bilangan desimal 2.7561.
6. Sebagaimana diketahui persamaan kuadrat $ax^2 + bx + c = 0$ mempunyai akar yang diformulasikan dalam bentuk standar $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. Untuk $a = 0.05010$, $b = -98.78$, dan $c = 5.015$, akar persamaan ini eksak dalam 10 digit signifikan adalah $x_1 = 1971.605916$ dan $x_2 = 0.05077069387$.

   a) Tunjukkan bahwa formula ini ekuivalen dengan formula alternatif $x = \frac{2c}{-b \mp \sqrt{b^2-4ac}}$.

   b) Dengan menggunakan koefisien yang diketahui di atas, selesaikan persamaan kuadrat dengan menggunakan kedua formula. Hitunglah kesalahan relatif yang ditimbulkan oleh masing-masing formula.

7. Kita telah melakukan eksperimen numerik bahwa nilai e diaproksimasi dengan menggunakan deret Taylor

   $$e^x = \lim_{n\to\infty} \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}\right)$$

3 AKAR PERSAMAAN TAKLINEAR

3.1 Pendahuluan

Persamaan linear satu variabel sudah diperkenalkan sejak di bangku SMP, yaitu persamaan yang berbentuk

$$ax + b = c \tag{3.1}$$

di mana $a \neq 0$, b dan c sebagai konstanta yang diberikan, dan x sebagai variabel yang tidak diketahui. Nilai x yang memenuhi persamaan ini disebut **penyelesaian, solusi** atau **akar** persamaan. Dalam kasus ini penyelesaian mudah diperoleh dalam bentuk yang sederhana, yaitu

$$x = \frac{c - b}{a}.$$

Dikatakan persamaan linear karena fungsi yang membentuk persamaan (3.1) berbentuk fungsi linear, yaitu $f(x) = ax + b$ dan grafiknya berbentuk garis lurus. Secara geometri, penyelesaian persamaan linear ini adalah absis titik potong antara grafik $y = ax + b$ dan garis $y = c$, seperti ditunjukkan pada Gambar 3.1(a).

Persamaan linear di atas dapat ditulis dalam bentuk umum sebagai $f(x) = c$. Dapat pula disajikan secara implisit $f(x) - c = 0$ atau dengan notasi lain $g(x) = 0$ di mana $g(x) = f(x) - c$. Secara geometris penyelesaian persamaan linear $g(x) = 0$ adalah absis titik potong grafik $y = g(x)$ dengan sumbu $\mathbb{X}$, yaitu nilai x di mana $g(x) = 0$, seperti ditunjukkan pada Gambar 3.1(b). Oleh karena itu, penyelesaian persamaan kadang pula disebut pembuat nol. Dalam masalah sehari-hari kita sering dihadapkan pada persamaan linear satu variabel seperti contoh berikut.

Contoh 3.1. Misalkan ongkos naik taksi diberlakukan dengan sistem biaya buka pintu Rp 10.000,00 dan biaya jarak tempuh dengan tarif Rp 5.000,00 setiap kilometernya. Bila seseorang naik taksi menghabiskan Rp 50.000,00 berapa kilometer jarak yang ditempuh orang tersebut.

- **Persamaan eksponensial**

$$x^2 + e^{-x} = 0.$$

  Diperhatikan untuk setiap bilangan real x, suku x^2 tidak akan pernah negatif dan suku e^{-x} selalu positif sehingga jumlah kedua suku ini tidak akan pernah nol. Ini berarti persamaan ini tidak mempunyai penyelesaian.

Dari beberapa contoh ini terlihat bahwa permasalahan pada persamaan taklinear tidak sesederhana seperti pada persamaan linear. Beberapa permasalahan yang dapat kita rangkum sementara adalah sebagai berikut.

1. Bentuk persamaan taklinear sangat beragam.
2. Tidak ada formula eksplisit untuk penyelesaian persamaan taklinear, kecuali pada bentuk sangat sederhana seperti persamaan kuadarat.
3. Eksistensi akar persamaan taklinear sulit dideteksi.
4. Banyaknya akar persamaan taklinear sulit dipastikan.
5. Menentukan akar persamaan taklinear cukup sulit.

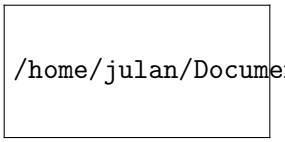
3.2 Eksistensi Akar Persamaan Taklinear

Persamaan taklinear satu variabel adalah persamaan yang mempunyai bentuk umum

$$f(x) = 0 \tag{3.2}$$

di mana f suatu fungsi taklinear. Fungsi taklinear adalah fungsi yang berbentuk selain dari $f(x) = ax + b$. Dengan definisi ini, fungsi yang bersesuaian dengan Contoh 3.2 sebelumnya adalah berturut-turut: $f(x) = ax^2 + bx + c$, $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + a_nx^n$, $f(x) = \sin x - 1$ dan $f(x) = x^2 + e^{-x}$.

Penyelesaian atau akar persamaan taklinear adalah semua nilai x yang memenuhi persamaan (3.2). Akar persamaan taklinear dapat tunggal, dapat lebih dari satu, atau bahkan dapat tidak mempunyai akar sama sekali. Secara grafik, akar persamaan taklinear (3.2) adalah absis titik potong grafik dengan sumbu X. Berikut ini diberikan ilustrasi grafik mengenai akar persamaan taklinear di dalam suatu interval. Pada Gambar (3.2)(a) ditunjukkan situasi di mana hanya terdapat satu akar dalam interval $[a, b]$, ada dua akar pada Gambar (3.2)(b) dan tidak satu pun akar seperti pada Gambar (3.2).



Gambar 3.4: Situasi yang memenuhi $f(a)f(b) > 0$

toh berikut diambil dari Faires dan Burden (2003) dalam [**?**], menyajikan permasalahan cukup rumit yang memuat persamaan taklinear.

Contoh 3.3. Misalkan $N(t)$ menyatakan ukuran populasi pada saat. Bila λ menyatakan angka kelahiran dan ν besar imigrasi tetap maka ukuran populasi memenuhi persamaan diferensial

$$\frac{dN(t)}{dt} = \lambda N(t) + \nu.$$

Dengan menggunakan teori persamaan diferensial, diperoleh penyelesaiannya sebagai berikut

$$N(t) = N_0 e^{\lambda t} + \frac{\nu}{\lambda}\left(e^{\lambda t} - 1\right)$$

di mana N_0 besar populasi awal. Misalkan populasi tertentu pada awalnya terdiri dari 1.000.000 individu, 435.000 populasi berimigrasi ke dalam komunitas ini dalam satu tahun pertama, dan terdapat 1.564.000 total individu pada akhir tahun pertama, berapakah angka kelahiran pada populasi tersebut?

Penyelesaian Berdasarkan data ini diketahui $N_0 = 1000000$, $\nu = 435000$, $N(1) = 1564000$, sehingga dengan cara subtitusi kedalam penyelesaian persamaan diferensial di atas maka permasalahan ini menjadi masalah menentukan λ akar persamaan taklinear berikut.

$$1564000 = 1000000e^{\lambda} + \frac{435000}{\lambda}\left(e^{\lambda} - 1\right)$$

atau

$$\frac{435}{\lambda}\left(e^{\lambda} - 1\right) + 1000e^{\lambda} - 1564 = 0.$$

Penyelesaian persamaan ini tidak mudah diperoleh secara analitis atau eksak. Untuk itu diperlukan metode aproksimasi untuk menyelesaikannya. ■

Selanjutnya diberikan beberapa metode numerik untuk mengaproksimasi akar persamaan taklinear.

3.3.1 Algoritma Metode Bagidua

Untuk memudahkan dalam melakukan komputasi terutama dalam menyusun program komputer, diberikan algoritma berikut.

- Mulailah dengan interval yang memuat akar $[a, b]$.
- Ambil $a_1 := a$ dan $b_1 := b$.
- Untuk $n = 1, 2, \cdots$, bangunlah barisan (p_n), (a_{n+1}) dan (b_{n+1}) sebagai berikut:

$$p_n := \frac{a_n + b_n}{2} \tag{3.3}$$

dan

$$\begin{cases} a_{n+1} := a_n,\ b_{n+1} := p_n & \text{bila } f(a_n)f(p_n) < 0 \\ a_{n+1} := p_n,\ b_{n+1} := b_n & \text{bila } f(a_n)f(p_n) > 0 \end{cases}$$

Barisan (p_n) yang dihasilkan akan konvergen ke akar $f(x) = 0$, yaitu $\lim_{n\to\infty} p_n = p$. Aproksimasi akar biasanya cukup diambil p_N untuk N yang cukup besar.

Contoh 3.4. Perhatikan persamaan taklinear $x^2 - 4\sin x = 0$. Persamaan ini terdeteksi langsung mempunyai penyelesaian $x = 0$ sebab $0^2 - 4\sin 0 = 0$. Penyelesaian lainnya tidak dapat dideteksi secara langsung. Coba perhatikan interval $[1, 2]$. Dengan menetapkan $f(x) = x^2 - 4\sin x$ maka diperoleh

$$f(1) = 1^2 - 4\sin 1 = -2.3659 < 0 \text{ dan } f(2) = 2^2 - 4\sin 2 = 0.3628 > 0$$

sehingga $f(1)f(2) < 0$. Karena f kontinu maka berdasarkan Teorema 3.1 disimpulkan interval $[1, 2]$ pasti memuat salah satu akarnya. Selanjutnya kita akan aproksimasi akar ini dengan menggunakan metode bagidua sebagai berikut.

Aproksimasi ke-1

Langkah 1 : Ambil $a_1 := 1$ dan $b_1 := 2$ dan $p_1 = \frac{1+2}{2} = 1.5$

Langkah 2 : Periksa lokasi akar sebagai berikut

$$f(p_1) \quad = \quad f(1.5) = 1.5^2 - 4\sin 1.5 = -1.7400 < 0.$$

Karena $f(a) = f(1) < 0$ maka $f(a_1)f(p_1) > 0$. Jadi akarnya terletak pada subinterval $[p_1, b_1] = [1.5, 2]$.

Langkah 3: Tetapkan interval $[a_2, b_2]$ dengan $a_2 = 1.5$ dan $b_2 = 2$

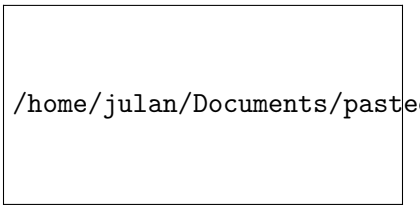
Teorema 3.2. *Misalkan persamaan taklinear $f(x) = 0$ mempunyai akar di dalam interval $[a, b]$ dengan akar eksak p. Bila (p_n) barisan aproksimasi yang diperoleh dengan metode bagidua maka berlaku*

$$|p_n - p| \leq \frac{b-a}{2^n} \text{ untuk setiap } n \geq 1. \tag{3.4}$$

Bukti. Perhatikan $b_2 - a_2 = \frac{1}{2}(b_1 - a_1)$, $b_3 - b_2 = \frac{1}{2}(b_2 - a_2) = \frac{1}{2^2}(b_1 - a_1)$. Dengan cara ini maka diperoleh

$$\begin{aligned} b_n - a_n &= \frac{1}{2}(b_{n-1} - a_{n-1}) = \frac{1}{2^2}(b_{n-2} - a_{n-2}) = \cdots \\ &= \frac{1}{2^{n-1}}\left(b_{n-(n-1)} - a_{n-(n-1)}\right) \\ &= \frac{1}{2^{n-1}}(b_1 - a_1) = \frac{1}{2^{n-1}}(b-a) \quad (*) \end{aligned}$$

Karena $p_n = \frac{1}{2}(a_n + b_n)$ maka jarak p_n terhadap p memenuhi estimasi $|p_n - p| \leq \frac{1}{2}(b_n - a_n)$. Fakta ini diilustrasikan pada Gambar 3.6 berikut.



Gambar 3.6: Jarak antara aproksimasi dan eksak

Karena itu dengan menggunakan (*) diperoleh

$$|p_n - p| \leq \frac{1}{2}(b_n - a_n) \leq \frac{1}{2} \cdot \frac{1}{2^{n-1}}(b-a) = \frac{1}{2^n}(b-a) \quad \blacksquare.$$

Berdasarkan teorema ini, semakin sempit interval $[a, b]$ yang memuat akar semakin cepat kekonvergenan barisan aproksimasi. Bila aproksimasi yang diinginkan kesalahannya tidak melebihi suatu toleransi TOL yang diberikan maka banyak iterasi dapat ditetapkan terlebih dahulu dengan menggunakan fakta berikut.

$$\frac{1}{2^n}(b-a) < TOL \rightarrow |p_n - p| < TOL.$$

1. Banyak iterasi N ditentukan di muka di mana $N = \lceil \log_2 \left(\frac{b-a}{TOL}\right) \rceil$. Dengan kata lain N adalah bilangan asli pertama yang lebih besar atau sama dengan $\log_2 \left(\frac{b-a}{TOL}\right)$. Kriteria ini didasarkan pada persamaan 3.5.

2. Berhenti bilamana $|f(p_N)| < TOL$. Kriteria ini didasarkan pada asumsi semakin $f(p_N)$ dekat ke nol semakin p_N dekat pula ke p. Asumsi ini dapat dipenuhi jika f kontinu di p dan disekitarnya.

3. Berhenti bilamana $b_N - a_N < TOL$. Kriteria ini didasarkan pada kenyataan bahwa akar eksaknya terletak di dalam interval $[a_N, b_N]$.

4. Berhenti bilamana $|p_N - P_{N-1}| < TOL$. Kriteria ini mengatakan bahwa jika selisih antara dua aproksimasi berurutan sudah sangat kecil maka dipastikan aproksimasinya sudah mendekati eksak. Kriteria ini didasarkan pada kekonvergenan barisan Cauchy pada interval tertutup $[a, b]$.

Pada kriteria 1 banyak iterasi N didefinisikan di awal sebelum algoritma dijalankan, sedangkan pada kriteria 2, 3, dan 4 banyak iterasi diperoleh setelah algoritma berakhir. Kriteria *stopping* 1 dan 3 adalah khusus untuk metode bagidua, sedangkan kriteria 2 dan 4 dapat dipakai untuk metode iterasi lainnya.

Contoh 3.6. Diperhatikan persamaan taklinear $x - e^{-x/c} = 0$ di mana c suatu konstanta. Persamaan ini akan kita selesaikan dengan menggunakan MATLAB.

Dengan mendasarkan pada algoritma sebelumnya maka m-file MATLAB berikut disusun. Untuk membatasi banyak iterasi perlu ditetapkan kriteria *stopping* terlebih dahulu. Beberapa kode MATLAB sederhana untuk metode bagidua diberikan sebagai berikut.

Kode MATLAB dengan kriteria *stopping* 1[1]

```
function [akar,langkah]=bagidua1(a,b,tol)
%INPUT : titik ujung interval a, b, toleransi = tol
%OUTPUT : akar : barisan akar aproksimasi
% langkah : banyaknya langkah yang dibutuhkan.
langkah = 0; akar =[];
N=ceil(log2((b-a)/tol));%banyak iterasi minimal yang dibutuhkan
for k=1:N
```

[1]baris yang ditandai oleh % hanya komentar atau keterangan, tidak dieksekusi

5. Selanjutnya untuk menjalankan program ini kembalilah ke *command window* yaitu posisi ketika pertama kali MATLAB dibuka. Melalui kursor tuliskan perintah
   `>>[akar,langkah]=bagidua1(a,b,tol)`
   di mana `a` dan `b` diganti oleh bilangan sehingga interval $[a, b]$ terjamin memuat akar, `tol` diganti dengan toleransi yang diinginkan. Sebagai contoh kita ambil interval $[0, 1]$ dan tol $= 0.001$. Pada interval ini dapat ditunjukkan bahwa persamaan $x - e^{-x/3} = 0$ terjamin mempunyai sebuah akar. Setelah dimasukkan nilai-nilai parameter ini, kemudian tekan **enter** maka diperoleh tampilan pada *command windows* sebagai berikut.

```
akar=

    0.5000
    0.7500
    0.8750
    0.8125
    0.7813
    0.7656
    0.7734
    0.7695
    0.7715
    0.7725

langkah=

    10
```

Hasil keluaran (*output*) ini menunjukkan bahwa untuk mendapatkan aproksimasi dengan kesalahan tidak melebihi 0.001 diperlukan 10 iterasi dan 0.7725 sebagai aproksimasi akarnya. Bila diinginkan untuk nilai c yang lain, misalnya $c = 5$ maka pada baris terakhir m-file di atas $c = 3$ cukup diganti dengan $c = 5$. Selanjutnya dijalankan seperti prosedur sebelumnya.

Permasalahan pokok metode bagidua ini adalah terletak pada bagaimana cara memastikan suatu interval memuat akar. Hati-hati, metode ini tidak berlaku bila interval awal yang diambil tidak memuat akar. Salah satu caranya adalah dengan cara coba-coba yang didasarkan pada Teorema lokasi akar. Cara yang lebih efisien adalah melalui grafik fungsi yang juga dapat dikerjakan dengan bantuan MATLAB. Prosedur untuk menggambar grafik fungsi $f(x) = x - e^{-x/5}$ dengan $c = 0.5$, ketik pada *command window* kalimat berikut.

Selanjutnya kode ini disimpan dalam m-file yang lain dengan nama `bagidua2.m`. Melalui *command window* diperoleh output berikut.

```
>> [akar,langkah]=bagidua2(0,1,0.001)

    akar =
         0.5000
         0.7500
         0.8750
         0.8125
         0.7813
         0.7656
         0.7734
    langkah =
          7
```

Ini berarti pada iterasi ke-7 algoritma berhenti dan diperoleh aproksimasi $p_7 = 0.7734$ yang memenuhi kriteria $|f(0.7734)| < 0.001$. Untuk kriteria *stopping* 3 cukup mengganti baris `while abs(fp)>tol` pada m-file `bagidua2.m` sebelumnya dengan `while abs(a-b)>tol` atau dengan `while b-a>tol`, sedangkan yang lainnya tetap. Sedangkan untuk kriteria 4 sedikit berbeda sengaja ditinggalkan untuk bahan latihan.

Contoh 3.7. Perhatikan kembali persamaan taklinear $x - e^{-x/c} = 0$. Persamaan ini muncul pada masalah arus dan tegangan pada suatu rangkaian listrik dan telah diperkenalkan oleh Armstrong and Kulesza (AK), pada 1981 ditulis kembali oleh Penny and Lindfield (2000) dalam [**?**]. Menariknya, dengan menggunakan ekspansi deret mereka berhasil menemukan formula eksplisit untuk mengaproksimasi akarnya yaitu

$$x = cu\left(1 - \frac{\ln\left[(1+c)u\right]}{1+u}\right) \text{ dimana } u = \ln\left(1 + \tfrac{1}{c}\right).$$

Formulasi ini ternyata cukup ampuh untuk nilai-nilai c yang berada pada rentang $[10^{-3}, 100]$. Tabel 3.2 menyajikan perbandingan hasil antara aproksimasi yang diberikan oleh formula AK, metode bagidua dengan tol=0.001, dan akar eksak 4 digit dengan menggunakan fungsi `fzero` pada MATLAB untuk beberapa nilai c.

Soal-soal untuk Latihan

1. Diberikan persamaan taklinear $\sqrt{x} - \cos x = 0$.

/home/julan/Documents/pasted12.png

Gambar 3.9: Skema metode *secant*

a) $[-3, 2.5]$

b) $[-0.5, 3]$

(c) $[-2.5, 3]$

(d) $[-3, -0.5]$

5. Buatlah m-file MATLAB untuk metode bagidua dengan menggunakan kriteria *stopping* 3 dan 4. Kemudian terapkan keempat m-file tersebut pada persamaan taklinear $x^3 + x - 4 = 0$ pada interval $[1, 4]$ dengan derajat akurasi 10^{-5}. Apakah banyak iterasi yang dihasilkan oleh masing-masing m-file berbeda?

6. Selidikilah apakah persamaan $\cos x + \frac{1}{1+e^{-2x}} = 0$ mempunyai akar. Gunakan bantuan grafik untuk mendeteksi keberadaan akar tersebut. Apakah akar tersebut dapat diaproksimasi dengan menggunakan metode bagidua, jelaskan.

3.4 Metode Secant

Barisan aproksimasi yang diperoleh pada metode bagidua pasti konvergen ke akar eksaknya, tetapi biasanya agak lambat, khususnya dalam kasus di mana lokasi akarnya terletak dipinggir interval. Suatu pertimbagan rasional bahwa jika $|f(a)| < |f(b)|$ maka akarnya lebih dekat pada a daripada b, asalkan gradien atau kemiringannya tidak terlalu tajam. Inilah mungkin ide awal metode *secant*.

Kalau pada metode bagidua nilai aproksimasi diambil sebagai titik tengah interval tanpa tergantung pada fungsi maka pada metode *secant* aproksimasi ini diambil sebagai titik potong **garis *secant*** dengan sumbu X. Garis *secant* adalah garis yang melalui $(a, f(a))$ dan $(b, f(b))$ sehingga penentuan nilai aproksimasi bergantung pada fungsi. Ilustrasi metode *secant* diberikan pada Gambar 3.9(a) dan bedanya dengan metode bagidua ditunjukkan pada Gambar 3.9(b).

Untuk menentukan formula iterasi metode *secant* ini mari perhatikan garis *secant* yang melalui $(a, f(a))$ dan $(b, f(b))$. Untuk memudahkan indeks selanjutnya, ambil $p_{-1} := a$ dan $p_0 := b$. Selanjutnya aproksimasi $p_1$diambil sebagai absis titik potong garis *secant* tersebut dengan sumbu X. Persamaan garis *secant* sekarang yang melalui $(p_{-1}, f(p_{-1}))$

Kriteria stopping metoda *secant*

1. Berhenti bila $|f(p_n)| < TOL$. Kriteria menggunakan pertimbangan pada sifat kontinu fungsi f seperti yang telah dijelaskan sebelumnya.
2. Berhenti bila $|p_n - p_{n-1}| < TOL$.

Kriteria 2 ini didasarkan pada kenyataan bahwa jika jarak antara kedua aproksimasi berurutan semakin lama semakin kecil, yakni $|p_n - p_{n-1}| < TOL$ maka barisan aproksimasi (p_n) akan konvergen ke akar eksaknya mengingat (p_n) merupakan barisan Cauchy pada $\mathbb{R}$ sehingga ia akan konvergen. Pengertian barisan Cauchy dan sifat-sifatnya dapat merujuk ke buku teks analisis real, misalnya Bartle dan Sherbet (1994) dalam [?].

Contoh 3.8. Gunakan metode *secant* untuk persamaan taklinear $x - e^{-x/3} = 0$ pada interval $[0, 1]$.

Penyelesaian. Ambil $p_{-1} = 0$, $p_0 = 1$. Diperoleh $f(p_{-1}) = 0 - e^0 = -1$ dan $f(p_0) = 1 - e^{-1/3} = 0.2835$. Diperoleh aproksimasi pertama sebagai berikut

$$p_1 = p_0 - \frac{f(p_0)(p_0 - p_{-1})}{f(p_0) - f(p_{-1})} = 1 - \frac{0.2835(1-0)}{0.2835 - (-1)} = 0.7791.$$

Untuk memperoleh aproksimasi kedua dihitung dulu $f(p_1) = f(0.7791) = 0.0078$. Diperoleh

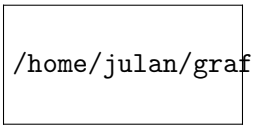
$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} = 0.7791 - \frac{0.0078(0.7791 - 1)}{0.0078 - 0.2835} = 0.7728.$$

Akurasi aproksimasi kedua ini sudah cukup akurat karena $|f(p_2)| = |f(0.7728)| = 0.000104$. ■

3.4.2 Komputasi dengan MATLAB

Untuk melakukan komputasi numerik dengan komputer, berikut diberikan kode MATLAB dengan menggunakan kriteria *stopping* 2.

```
function [akar,langkah]=secant2(a,b,tol)
p_1 = a; p0 = b;
f_1 = feval('fun',p_1); f0 = feval('fun',p0);
langkah = 0; akar=[];
while abs(p_1-p0) > tol
```



Gambar 3.10: Deteksi lokasi akar melalui grafik

fungsi yang bersesuaian, yaitu y=x^3-4*x^2+x+1. Selanjutnya diselesaikan dengan MATLAB pada tiap-tiap interval yang memuat akar. ■

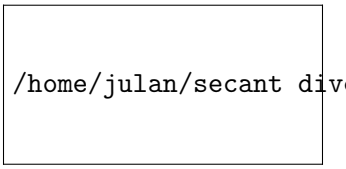
Untuk akar di dalam $[-1, 0]$ diberikan perintah

```
>>[akar,langkah]=secant1(-1,0,0.0001)
akar =

    -0.1667
    -0.5902
    -0.3261
    -0.3664
    -0.3779
    -0.3772

langkah

    = 6

>>[akar,langkah]=bagidua2(-1,0,0.0001)
akar =

    -0.5000
    -0.2500
    -0.3750
    -0.4375
    -0.4063
    -0.3906
    -0.3828
    -0.3789
    -0.3770
    -0.3779
    -0.3774
    -0.3772
```

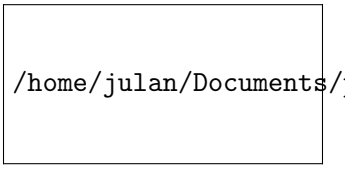
Terakhir pada interval $[3, 4]$ diperoleh akar sebagai berikut.

```
>>[akar,langkah]=secant1(3,4,0.0001)
akar =

    3.5000
    3.6226
    3.6539
    3.6510
    3.6511

langkah =

    5

>> [akar,langkah]=bagidua2(3,4,0.0001)
akar =

    3.5000
    3.7500
    3.6250
    3.6875
    3.6563
    3.6406
    3.6484
    3.6523
    3.6504
    3.6514
    3.6509
    3.6511
    3.6510
    3.6511
    3.6511

langkah =

    15
```

Dari semua kasus terlihat bahwa kekonvergenan metode *secant* sangat cepat dibandingkan metode bagidua. Namun ada kasus di mana metode *secant* tidak berjalan sama sekali seperti ditunjukkan pada contoh berikut.



Gambar 3.11: Situasi metode *secant* tidak konvergen



Gambar 3.12: Skema aproksimasi metode *false position*

kedua titik ini tidak lagi memuat akar eksak sehingga titik potongnya dengan sumbu X semakin menjauh dari akar eksak tersebut, yakni $p_2 = 3.4721$ sudah keluar interval $[2, 3]$. Skema ini ditunjukkan pada Gambar 3.11. Begitu juga seterusnya, tidak mungkin kembali lagi ke dalam interval awal yang memuat akar. Bahkan dalam perjalanan komputasinya kemungkinan bertemu operasi aritmatika yang menghasilkan NaN atau INF. Keluaran INF muncul bilamana bilangan yang dihasilkan melebihi `realmax,` yaitu bilangan terbesar pada MATLAB sekitar 10^{308}.

Untuk mengatasi kelemahan ini muncul ide penggabungan kedua metode ini menjadi satu metode yang lebih baik, yaitu metode *false position* atau sering disebut juga **regula falsi**.

3.5 Metode *False Position*

Pada metode ini, pengambilan aproksimasi mengikuti metode *secant,* namun pembentukan garis *secant* menggunakan ide pada metode bagidua yaitu dengan cara mempertahankan subinterval yang memuat akar. Dengan cara ini maka barisan aproksimasi dipastikan konvergen ke akar eksaknya.

Metoda ini diilustrasikan pada Gambar 3.12. Pada skema ini, p_1 diambil seperti pada metode *secant*. Sedangkan p_2 diperoleh melalui garis *secant* yang ditentukan oleh p_{-1} dan p_1, bukan oleh p_0 dan p_1 seperti pada metode *secant*. Menentukan apakah garis *secant* dibentuk oleh p_{k-1} dan p_k atau oleh p_{k-2} dan p_k tergantung pada interval mana posisi

3.5.2 Komputasi dengan MATLAB

Selanjutnya pekerjaan komputasi ini diselesaikan dengan MATLAB. Berikut salah satu kode yang dapat digunakan.

```
function [akar,langkah]=falsepos1(a,b,tol)
fa = feval('fun',a); fb = feval('fun',b);
p = a-fa*(b-a)/(fb-fa);
langkah = 1; akar =[p];
fp = feval('fun',p);
while abs(fp) > tol

    langkah = langkah+1;
    if sign(fa)*sign(fp)<0

        a = a; b = p;
        else
        a = p; b = b;

    end

fa = feval('fun',a); fb = feval('fun',b);
p = a-fa*(b-a)/(fb-fa);
akar = [akar;p];
fp = feval('fun',p);
end
%fungsi yang diberikan
function y = fun(x)

    y = (x^3-4*x^2+x+1)*sin(3*x);
```

Selanjutnya, Contoh 3.11 diselesaikan dengan perintah sebagai berikut.

```
>> [akar,langkah]=falsepos1(2,3,0.0001)
akar =

    2.4041
    2.0932
    2.0945
    2.0944

langkah =
```

/home/julan/Documents/pasted16.png

Gambar 3.13: Skema metode Newton

3.6 Metode Newton

Metode ini merupakan metode yang paling populer karena secara umum kekonvergenannya lebih cepat dari metode lainnya dan implementasinya sederhana. Kalau metode *secant* dan metode *false position* menggunakan garis *secant* untuk mendapatkan aproksimasi, metode Newton menggunakan garis tangen, yaitu garis singgung kurva. Pada metode Newton ini hanya dibutuhkan satu titik awal untuk membuat garis tangen.

Misalkan p_0 titik awal yang dipilih maka p_1 diambil sebagai absis titik potong garis singgung kurva $y = f(x)$ dititik $(p_0, f(p_0))$. Selanjutnya, melalui titik $(p_1, f(p_1))$ dibuat garis singgung untuk mendapatkan p_2. Untuk lebih jelasnya, skema Newton ini ditunjukkan pada Gambar 3.13.

Selanjutnya kita pelajari cara menemukan formula iterasi metode Newton ini. Persamaan garis singgung kurva $y = f(x)$ dititik $(p_0, f(p_0))$ mempunyai bentuk

$$y - f(p_0) = m(x - p_0)$$

di mana $m = f'(p_0)$ derivatif pertama f di titik p_0. Titik potong garis ini dengan sumbu X diperoleh dengan menetapkan $y = 0$. Setelah dilakukan beberapa langkah diperoleh

$$x = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Selanjutnya, aproksimasi pertama diperoleh dengan mengambil $p_1 := x$ sehingga diperoleh

$$p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Pola ini dilanjutkan dengan memandang p_1 sebagai sebagai pengganti p_0 sebelumnya, dan seterusnya hingga diperoleh algoritma berikut.

3.6.2 Komputasi dengan MATLAB

Selanjutnya untuk melakukan komputasi dengan MATLAB kita perlu menyusun kode yang diperlukan. Salah satu kode yang dapat digunakan diberikan sebagai berikut.

```
[akar,langkah]=newton1(p0,tol)
%INPUT: iterasi awal p0 dan toleransi tol
%OUTPUT: akar:barisan aproksimasi, langkah: banyak langkah
fp = feval('fun',p0); dfp = feval('dfun',p0);
if dfp==0

    error('metode Newton gagal!')

end
akar =[]; langkah = 0;
while abs(fp)>tol
langkah = langkah+1;
p = p0-fp/dfp;
akar = [akar;p];
p0=p;
fp = feval('fun',p0);
dfp = feval('dfun',p0);

    if dfp==0
        error('metode Newton gagal!')
    end

end
%fungsi yang diketahui
function y = fun(x)
y = x^3+4*x^2-10;
%derivatifnya
function dy = dfun(x)
dy = 3*x^2+8*x;
```

Pada kode ini diberikan perintah berikut

```
if dfp==0
error('metode Newton gagal!')
end
```

```
>> [akar,langkah]=falsepos1(1,2,0.0001)
akar = 1.26315789473684

    1.33882783882784
    1.35854634182478
    1.36354744004209
    1.36480703182678
    1.36512371788438
    1.36520330366260
    1.36522330198554
    1.36522832702552

langkah =

    9
```

Ternyata pada iterasi ke-9 metode *secant* sudah menghasilkan NaN dan tidak dapat dilanjutkan. Sedangkan metode *falseposition* mencapai akurasi yang sama pada iterasi ke-9. Pada kasus ini metode Newton memberikan kecepatan konvergensi paling tinggi.

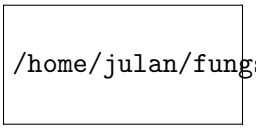
3.6.3 Masalah Kekonvergenan Iterasi Newton

Secara intuitif, iterasi pada metode Newton akan menghasilkan barisan aproksimasi yang konvergen jika pemilihan iterasi awal cukup dekat dengan akar eksaknya. Tetapi intuisi ini tidak sepenuhnya benar, seperti ditunjukkan pada contoh berikut [?].

Contoh 3.13. Kita akan selesaikan persamaan

$$\frac{1}{2} + \frac{1}{4}x^2 - x \sin x - \frac{1}{2} \cos 2x = 0$$

dengan $p_0 = \frac{\pi}{2}$. Kemudian kita gunakan $p_0 = 5\pi$ dan $p_0 = 10\pi$.
Pertama kita deteksi dulu lokasi akar melalui grafik fungsinya. Gambar 3.14 panel kiri menyajikan grafik fungsi $f(x) = \frac{1}{2} + \frac{1}{4}x^2 - x \sin x - \frac{1}{2} \cos 2x$ pada range $[-10\pi, 10\pi]$ dan Gambar 3.14 panel kanan menyajikan *zoom* grafik ini pada range $[-\pi, \pi]$ sehingga detailnya lebih jelas.



Gambar 3.15: Derivatif berosilasi di sekitar nol

```
     269

>> [akar,langkah]=newton1(10*pi,1e-5)

     akar =
     1.78539816339745
     1.79866520894184
     1.80886602322048
     1.81700550338525
     ⋮
     1.89162044410159
     1.89163704438203

langkah =

     238
```

Berdasarkan eksperimen numerik ini ketiga pemilihan titik awal menghasilkan barisan aproksimasi yang konvergen, khususnya akar yang berada pada interval $[1, 2]$. Hasil ini menunjukkan bahwa pemilihan titik awal yang cukup jauh dengan akar eksaknya belum tentu lebih lambat daripada titik awal yang lebih dekat. Perhatikan titik awal $p_0 = 5\pi$ malah lebih lambat kekonvergenannya daripada titik awal $p_0 = 10\pi$. Bahkan titik awal yang sangat dekat $p_0 = \frac{\pi}{2}$ tidaklah begitu signifikan berbeda kecepatan konvergensi dibandingkan titik awal yang lebih jauh.

Ini merupakan kasus spesifik yang menunjukkan bahwa metode Newton tidak memberikan konvergensi yang cepat. Salah satu ciri yang ada pada kasus ini yang kemungkinan berpengaruh terhadap kekonvergenan adalah profil derivatifnya yang cenderung berosilasi dengan di sekitar titik aproksimasi. Bahkan banyak titik yang derivatifnya bernilai nol di mana metode Newton tidak berlaku sama sekali bila titik ini terakomodasi dalam iterasi. Grafik fungsi derivatif ini ditunjukkan pada Gambar 3.15.

Secara teoretis sulit untuk memastikan apakah metode Newton konvergen untuk suatu nilai awal yang diambil. Tetapi eksistensi abstrak adanya titik awal yang menjamin metode Newton konvergen diungkapkan pada teorema berikut.

- Bila formula eksplisit untuk derivatif fungsi tidak ada, misalnya dikarenakan fungsi yang diketahui hanya berupa data diskret.

Masalah kedua masih dapat diatasi dengan menggunakan aproksimasi derivatif, salah satunya menggunakan formula mundur sebagai berikut

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

Bila ini disubstitusikan ke dalam formula iterasi Newton maka diperoleh

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})} \tag{3.9}$$

yang tidak lain merupakan metode *secant*. Jadi metode *secant* merupakan versi diskret dari metode Newton. Sedangkan metode yang lebih umum lagi adalah metode titik tetap (*fixed point*) seperti dibahas berikut ini.

3.7 Metode Titik Tetap

Metode titik tetap merupakan bentuk umum dari beberapa metode iterasi yang ada, termasuk metode Newton. Ide dasar metode ini adalah dengan mengubah bentuk umum persamaan taklinear

$$f(x) = 0 \tag{3.10}$$

dengan bentuk lain yang ekuivalen, yaitu

$$x = \phi(x). \tag{3.11}$$

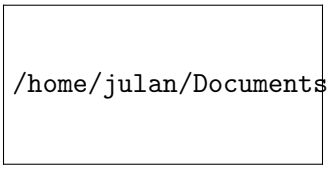
Kedua persamaan (3.10) dan (3.11) dikatakan ekuivalen jika keduanya mempunyai akar atau penyelesaian yang sama. Selanjutnya untuk suatu nilai awal x_0 tertentu, didefinisikan iterasi berikut

$$x_{n+1} = \phi(x_n),\ n = 1, 2, 3, \cdots \tag{3.12}$$

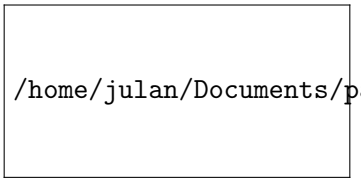
Teorema 3.4. *Bila ϕ kontinu dan (x_n) barisan yang dibangun oleh iterasi (3.12) konvergen maka ia konvergen ke akar persamaan taklinear (3.10).*

Bukti. Misalkan $\alpha = \lim_{n\to\infty} x_n$. Karena kontinu maka berlaku

$$\lim_{n\to\infty} \phi(x_n) = \phi(\lim_{n\to\infty} x_n).$$



Gambar 3.16: Ilustrasi adanya titik tetap



Gambar 3.17: Ilustrasi ketidaktunggalan titik tetap

maka begitu juga dengan h. Berlaku juga fakta berikut. Karena ϕ kontinu maka begitu juga dengan h. Berlaku juga fakta berikut

$$h(a) = \phi(a) - a > 0 \text{ dan } h(b) = \phi(b) - b < 0.$$

Berdasarkan Teorema lokasi akar (Teorema 3.1) maka pasti ada $\alpha \in (a, b)$ sehingga

$$h(\alpha) = 0 \leftrightarrow \phi(\alpha) - \alpha = 0 \leftrightarrow \alpha = \phi(\alpha).$$

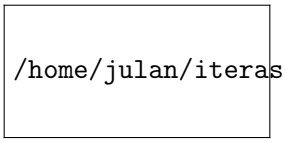
Jadi α adalah salah satu titik tetapnya. ■

Teorema ini hanya memberikan syarat cukup bagi adanya titik tetap, tidak memberikan jaminan ketunggalan titik tetap tersebut. Padahal pada metode iterasi titik tetap dibutuhkan ketunggalan titik tetap agar ada kepastian menuju ke mana kekonvergenan barisan aproksimasi. Gambar 3.17 menyajikan ilustrasi ketidaktunggalan titik tetap.

Salah satu teorema tentang eksistensi dan ketunggalan titik tetap diungkapkan sebagai berikut.

Teorema 3.6. *Jika ϕ memenuhi kondisi seperti pada Teorema 3.5 dan terdiferensial pada interval terbuka (a, b) dengan $|\phi'(x)| < 1$ untuk setiap $x \in (a, b)$ maka titik tetapnya tunggal.*

Bukti. Eksistensi titik tetap sudah dijamin oleh Teorema 3.5. Selanjutnya kita tunjukkan bahwa titik tetap ini tunggal. Andai titik tetapnya tidak tunggal, katakan α dan



Gambar 3.18: Pola spiral iterasi yang konvergen

TABEL 3.3: Nilai iterasi metoda titik tetap

n	$x_{n+1} = \phi_1(x_n)$	$x_{n+1} = \phi_2(x_n)$	$x_{n+1} = \phi_3(x_n)$
0	1.2040	0.7408	0.5204
1	-0.1856	0.4767	0.5573
2	1.6840-3.1416i	0.6208	0.5650
3	-1.2710+1.0787i	0.5375	0.5667
4	-0.5111-2.4378i	0.5842	0.5670
5	-0.9126+1.7774i	0.5576	0.5671
6	-0.6922-2.04511i	0.5726	0.5671

Berdasarkan pola bilangan yang terdapat pada tabel ini terlihat bahwa fungsi iterasi pertama tidak memberikan kekonvergenan sebab pada iterasi kedua sudah menghasilkan bilangan kompleks. Sedangkan iterasi kedua dan iterasi ketiga menunjukkan konvergen karena cenderung menuju bilangan yang sama. ■

3.7.2 Ilustrasi Grafis Iterasi Titik Tetap

Khusus untuk iterasi yang konvergen kita dapat melihat pola spiral pada tahapan iterasinya. Gambar 3.18 memberikan ilustrasi pada iterasi $x_{n+1} = e^{-x_n}$ di mana ia berangkat dari $x_0 = 0.3$. Pertama, dari $x_0 = 0.3$ dibentuk garis lurus sampai menyentuh grafik $y = e^{-x}$ sehingga diperoleh $x_1 = y = e^{-0.3}$. Selanjutnya dibuat garis lurus mendatar sampai menyentuh garis $y = x$, kemudian diproyeksikan ke sumbu X dan diperoleh nilai x_1 yang sama (ingat karena $y = x$). Bila sebelum sampai menyentuh garis $y = x$, dimulai dari titik potong kurva dibuat garis horizontal ke kiri sampai menyentuh garis $y = x$, kemudian diproyeksikan ke sumbu X maka akan diperoleh x_2. Begitu seterusnya. Bila iterasi ini konvergen maka akan tampak secara visual penggal-penggal garis berarah yang semakin lama semakin menuju titik potong $y = x$ dan $y = e^{-x}$, yaitu titik tetap pemetaan $x \mapsto \phi(x)$ di mana $\phi(x) = e^{-x}$.

```
function x=titiktetap1(x0,N)
%INPUT: x0 iterasi awal, N banyak iterasi
%OUTPUT: vektor x berukuran N+1 dimulai dari x0
x=zeros(N+1,1);
x(1)=x0;
for k=1:N
x(k+1)=feval('fun_iter',x(k));
end
%iterator
function y=fun_iter(x);
y=(x+exp(-x))/2;
```

Untuk m-file `titiktetap2` adalah sebagai berikut.

```
function x=titiktetap2(x0,tol)
x=[x0];
x1=feval('fun_iter',x0);
langkah=1;
while abs(x0-x1)>tol
langkah = langkah+1;
xp = feval('fun_iter',x1);
x = [x;xp];
x0=x1; x1=xp;
end
%iterator
function y=fun_iter(x);
y=(x+exp(-x))/2;
```

Pada kode di atas, baris terakhir adalah fungsi iterasi. Pada tampilan di atas fungsi iterasi yang aktif adalah $\phi(x) = \frac{x+e^{-x}}{2}$. Untuk fungsi iterasi lainnya tinggal mengganti fungsi pada baris terakhir tersebut. Sebagai tip, beberapa fungsi iterasi dapat di tulis di mana fungsi yang tidak aktif cukup diberikan tanda % di depannya. Ketika mau diaktifkan tanda % dihilangkan, dan seterusnya. Jika tidak, maka harus menulis fungsi berulang-ulang. Berikut ini hasil iterasi jika menggunakan m-file `titiktetap2`.

```
>> [x,langkah]=titiktetap2(0.3,0.00001)

    x =

        0.30000000000000
```

```
>> help fzero.m
FZERO Scalar nonlinear zero finding.
      X = FZERO(FUN,X0) tries to find a zero of the function
FUN near X0. FUN accepts real scalar input X and returns a
real scalar function value F evaluated at X. The value X
returned by FZERO is near a point where FUN changes sign (if
FUN is continuous), or NaN if the search fails.
```

Berdasarkan bantuan ini fungsi fzero digunakan untuk mengaproksimasi akar persamaan taklinear $f(x) = 0$ di sekitar x_0. Fungsi f di sini diimplementasikan oleh FUN pada MATLAB dan x_0 dapat dipandang sebagai titik awal pada proses iterasi.

Langkah-langkah penggunaan fungsi fzero

1. Buka jendela editor, definisikan fungsi pada persamaan taklinear $f(x) = 0$. Misalkan kita akan menyelesaikan persamaan $x^3 + 4x^2 - 10 = 0$, maka fungsi m-file MATLAB yang perlu didefinsikan adalah sebagai berikut. Tulis pada jendela editor kalimat berikut.

   function y = fun1(x)
   y = x^3+4*x^2-10;

2. Setelah itu simpan dengan nama, katakan `fun1.m`.
3. Selanjutnya pada *command window*, berikan perintah berikut.

```
>> format long
>> x=fzero('fun1',1)
x =

    1.36523001341410
```

Perintah `format long` di sini untuk menampilkan bilangan dengan panjang digit 15 (ingat, defaultnya hanya 5 digit). Berdasarkan keluaran ini maka didapat akar persamaan adalah 1.3652 akurat sampai 5 digit.

Selanjutnya cobakan alat canggih MATLAB ini pada persamaan seperti pada Contoh yang pernah dibahas sebelumnya, yaitu

$$\frac{1}{2} + \frac{1}{4}x^2 - x\,\sin x - \frac{1}{2}\,\cos 2x = 0.$$

Dengan mengganti fungsi pada fun1 sebagai berikut

Gunakan metode Newton untuk mengaproksimasi nilai λ. Cobakan beberapa nilai awal seperti $\lambda_0 = 0, 0.25, 0.5, 1$.

3. Perhatikan fungsi $f(x) = 3^{3x+1} - 7 \cdot 5^{2x}$. Persamaan taklinear $f(x) = 0$ dapat diselesaikan secara analitis sebagai berikut.

$$\begin{aligned} 3^{3x+1} - 7 \cdot 5^{2x} &= 0 \\ 3 \cdot 3^{3x} &= 7 \cdot 5^{2x} \\ \left(\frac{3^3}{5^2}\right)^x &= \frac{7}{3} \\ \left(\frac{27}{25}\right)^x &= \frac{7}{3} \rightarrow x = \log_{\frac{27}{25}} \frac{7}{3}. \end{aligned}$$

   a) Gambarkan grafik fungsi f untuk mempertimbangkan pengambilan iterasi awal. Kemudian gunakan metode Newton untuk mendapatkan aproksimasi akar dengan akurasi 10^{-10}. Berapa langkah yang diperlukan.

   b) Gunakan fungsi `fzero` pada MATLAB dengan nilai awal yang sama seperti metode Newton di atas.

   Konfirmasikan hasil yang diperoleh melalui metode Newton dan fungsi `fzero` MATLAB dengan akar eksaknya.

4. Untuk persamaan taklinear $f(x) = x^2 - x - 2 = 0$ masing-masing fungsi berikut memenuhi sebagai iterator pada metode iterasi titik tetap

$$\begin{aligned} \phi_1(x) &= x^2 - 2, \\ \phi_2(x) &= \sqrt{x+2}, \\ \phi_3(x) &= 1 + \frac{2}{x}, \\ \phi_4(x) &= \frac{x^2+2}{2x-1}. \end{aligned}$$

   Selidikilah kekonvergenan metode iterasi titik tetap untuk masing-masing iterator tersebut. Karena persamaan mempunyai akar eksak $x = -1$ dan $x = 2$, identifikasilah ke akar mana masing-masing iterasi konvergen (bila ia konvergen).

5. Perhatikan persamaan taklinear $\cos x + \frac{1}{1+e^{-2x}} = 0$. Persamaan ini dipastikan mempunyai akar, tetapi syarat $f(a)f(b) < 0$ tidak dipenuhi untuk setiap bilangan real a dan b. Jadi metode bagidua, *secant,* dan *false position* akan gagal untuk menyelesaikan persamaan ini. Untuk itu metode Newton dan iterasi titik tetap kemungkinan dapat digunakan.

4 INTERPOLASI POLINOMIAL

4.1 Pendahuluan

Dalam kehidupan sehari-hari kita sering dihadapkan pada tabel yang terdiri atas angka-angka hasil pengukuran beberapa variabel. Misalnya data mengenai jarak tempuh suatu benda yang diukur setiap menit, frekuensi denyut nadi yang diukur setiap 5 detik, jumlah penduduk atau besar populasi yang dihitung setiap tahun, dan lain sebagainya. Di antara beberapa variabel tersebut biasanya terdapat paling tidak satu variabel bebas. Variabel waktu dalam satuan detik, menit, jam, bulan, tahun dan lain-lain, biasanya ditetapkan sebagai variabel bebas.

Misalkan $s(t)$ jarak tempuh suatu benda (dalam meter) setelah berjalan selama t menit. Dari pengukuran pada 10 menit pertama diperoleh data sebagai berikut.

TABEL 4.1: Waktu versus jarak tempuh

t	1	2	3	4	5	6	7	8	9	10
$s(t)$	30	45	65	75	100	120	155	180	200	215

Data ini dapat pula disajikan dalam bentuk grafik seperti pada Gambar 4.1 berikut.

Berdasarkan data ini kita dapat menentukan jarak tempuh benda pada waktu-waktu tertentu, misalnya 75 meter setelah berjalan 4 menit, 180 meter setelah berjalan 8 menit. Tetapi kita tidak dapat memastikan jarak yang telah ditempuh benda setelah berjalan $4\frac{1}{2}$

/home/julan/Documents/pasted25.png

Gambar 4.1: Grafik jarak dan waktu tempuh

4.2 Metode Interpolasi Lagrange

4.2.1 Polinomial Lagrange

Secara intuitif, melalui dua titik yang berlainan selalu dapat dibentuk polinomial derajat satu, melalui tiga titik berlainan selalu dapat dibentuk polinomial derajat dua, dan seterusnya. Selanjutnya, polinomial yang melalui titik-titik yang diberikan disebut polinomial interpolasi. Sebelum kita tunjukkan eksistensi polinomial interpolasi secara formal, kita perhatikan dulu satu kasus sederhana berikut sebagai ilustrasi.

Misalkan diketahui dua titik berlainan (x_0, y_0) dan (x_1, y_1), kemudian kita bangun dua buah polinomial derajat satu sebagai berikut

$$L_0(x) := \frac{x - x_1}{x_0 - x_1} \text{ dan } L_1(x) := \tfrac{x-x_0}{x_1-x_0}. \tag{4.1}$$

Diperoleh sifat $L_0(x_0) = 1$, $L_0(x_1) = 0$ dan $L_1(x_0) = 0$, $L_1(x_1) = 1$. Sifat ini diilustrasikan pada Gambar 4.3(kiri) berikut.

/home/julan/Documents/pasted27.png

Gambar 4.3: Polinomial Lagrange derajat 1 dan interpolasinya

Selanjutnya didefinisikan polinomial $P(x)$ sebagai kombinasi linear kedua polinomial Lagrange tadi,

$$P(x) = y_0 L_0(x) + y_1 L_1(x). \tag{4.2}$$

Diperoleh bahwa $P(x)$ merupakan polinomial derajat satu dengan sifat $P_0(x) = y_0 L_0(x_0) + y_1 L_1(x) = y_0 \cdot 1 + y_1 \cdot 0 = y_0$. Dengan cara yang sama dapat ditunjukkan dengan mudah bahwa $P(x_1) = y_1$. Polinomial hasil kombinasi ini ditunjukkan pada panel kanan Gambar 4.3.

Definisi 4.1. Misal diberikan $n + 1$ bilangan berlainan $x_0, x_1, \cdots, x_n$. Untuk setiap $k =$

/home/julan/Documents/pasted28.png

Gambar 4.4: Beberapa polinomial Lagrange

sebagai berikut:

$$\begin{aligned} L_{1,0}(x) &= \frac{(x-x_1)}{(x_0-x_1)} = \frac{x-3}{1-3} = -\frac{1}{2}(x-3) \\ L_{1,1}(x) &= \frac{(x-x_0)}{x_1-x_0} = \frac{x-1}{3-1} = \frac{1}{2}(x-1). \quad \blacksquare \end{aligned}$$

Contoh 4.2. Tentukan semua polinomial Lagrange yang berkaitan dengan titik $x_0 = -1, x_1 = 2$ dan $x_2 = 4$. Kemudian gambarkan grafik ketiga polinomial ini pada satu bidang koordinat !

Penyelesaian. Berdasarkan formula (4.3) diperoleh

$$\begin{aligned} L_{2,0}(x) &= \frac{(x-2)(x-4)}{(-1-2)(-1-4)} = \frac{1}{15}(x-2)(x-4), \\ L_{2,1}(x) &= \frac{(x+1)(x-4)}{(2+1)(2-4)} = -\frac{1}{6}(x+1)(x-4), \\ L_{2,2}(x) &= \frac{(x+1)(x-2)}{(4+1)(4-2)} = \frac{1}{10}(x+1)(x-2). \end{aligned}$$

Dengan menggunakan perintah MATLAB berikut diperoleh grafik yang dimaksud seperti ditunjukkan pada Gambar 4.4. $\blacksquare$

```
>>x=-1:0.05:4;
>>y1=(x-2).*(x-4)/15;y2=-(x+1).*(x-4)/6;y3=(x+1).*(x-2)/10;
>>plot(x,y1,x,y2,x,y3);
>>plot([-1 -1 2 2 4 4],[0 1 0 1 0 1],'.');grid
```

Secara umum profil polinomial Lagrange yang berkaitan titik dengan $x_0, x_{1,} \cdots, x_n$ yaitu polinomial yang memenuhi (4.3) ditunjukkan pada Gambar 4.5. Dalam visual ini polinomial Lagrange $L_{n,k}$ hanya bernilai 1 di $x = x_k$, sedangkan di titik *node* lainnya bernilai nol. Tidak ada kondisi khusus nilai $L_{n,k}$ di luar titik *node*.

Pada teorema ini ditegaskan bahwa derajat polinomial paling tinggi n. Ini berarti ada kemungkinan polinomial interpolasi yang terbentuk mempunyai derajat kurang dari n. Misalnya, 4 buah titik secara kebetulan terletak pada kurva parabola maka polinomial interpolasi yang dimaksud hanya berderajat dua, bukan berderajat tiga. Keadaan seperti ini sangat jarang terjadi. Pada umumnya, polinomial interpolasi yang melalui $n+1$ buah titik berbeda akan berderajat n.

Contoh 4.3. Diberikan pasangan data sebagai berikut

x	1.0	1.1	1.3	1.4	1,6	1.8	2.0
y	2.7183	3.0042	3.6693	4.0552	4.9530	6.0496	7.3891

Pada tabel ini, hanya tersedia beberapa pasangan data saja. Berapa nilai y yang berkaitan dengan $x = 1.2$.

Penyelesaian. Karena pada tabel tidak tersedia data y_k untuk $x_k = 1.2$ maka kita perlu mencari nilai pendekatan dengan menggunakan teknik interpolasi. Diperhatikan bahwa ada 7 pasangan titik sehingga kita dapat membangun polinomial interpolasi sampai derajat 6. Dalam praktiknya biasa digunakan interpolasi derajat lebih rendah. Terdapat beberapa kemungkinan interpolasi derajat lebih rendah dengan menggunakan titik-titik disekitarnya.

1. **Interpolasi linear**
   Perhatikan titik $x = 1.2$ berada di antara 1.1 dan 1.3 sehingga kita dapat menggunakan kedua titik $(1.1, 3.0042)$ dan $(1.3, 3.6693)$ untuk membuat interpolasi linear. Pertama, bangun dua polinomial Lagrange, yaitu

   $$L_{1,0}(x) = \frac{x - 1.3}{1.1 - 1.3} = -5(x - 1.3) \text{ dan } L_{1,1}(x) = \tfrac{x-1.1}{1.3-1.1} = 5(x - 1.1).$$

   Kemudian definisikan polinomial interpolasi dengan menggunakan (4.6), yaitu

   $$\begin{aligned} P_1(x) &= (3.0042)\,(-5(x - 1.3)) + (3.6693)\,(5(x - 1.1)) \\ &= -15.0210(x - 1.3) + 18.3465(x - 1.1). \end{aligned}$$

   Selanjutnya substitusikan $x_t = 1.2$ ke dalam $P_1(x)$ sehingga diperoleh

   $$y_t = P_1(1.2) = -15.0210(1.2 - 1.3) + 18.3465(1.2 - 1.1) = 3.3368.$$

2. **Interpolasi kuadratik**
   Dibutuhkan tiga titik untuk membangun polinomial interpolasi derajat dua di mana

Sesungguhnya masih terdapat aproksimasi lainnya dengan menggunakan polinomial interpolasi derajat 3, 4 dan seterusnya sampai derajat 6 di mana masing-masing terdapat beberapa kemungkinan. Silakan mencoba sendiri.

Contoh 4.4. Kembali pada contoh sebelumnya. Sesungguhnya data pada tabel di atas diperoleh dari fungsi $y = e^x$. Bandingkan ketelitian masing-masing aproksimasi yang telah diperoleh dengan menggunakan beberapa interpolasi polinomial tersebut.

Penyelesaian. Nilai eksak untuk $x = 1.2$ adalah $y = e^{1.2} = 3.3201$. Diperoleh kesalahan mutlak untuk masing-masing hasil di atas adalah
Interpolasi linear : $E = |P_1(1.2) - e^{1.2}| = |3.3368 - 3.3201| = 0.0167$
Interpolasi kuadratik 1: $E = |P_2(1.2) - e^{1.2}| = |3.3212 - 3.3201| = 0.0011$
Interpolasi kuadratik 2 : $E = |P_2(1.2) - e^{1.2}| = |3.3190 - 3.3201| = 0.0011$.
Berdasarkan hasil ini, kedua interpolasi kuadratik memberikan akurasi yang sama dengan kesalahan 0.0011. Dalam kasus ini, interpolasi kuadratik memberikan hasil lebih akurat daripada interpolasi linear. ■

Contoh 4.5. Kembali ke tabel data contoh 4.3, tentukan banyak cara interpolasi yang dapat digunakan untuk mengaproksimasi nilai y yang berkaitan dengan $x = 1.5$.

Penyelesaian. Kumpulkan ke dalam suatu tabel derajat interpolasi dan kemungkinan *node* yang digunakan. Berdasarkan tabel ini kita mempunyai 12 kemungkinan yang dapat digunakan untuk mengaproksimasi nilai y yang berkaitan dengan $x = 1.5$.

TABEL 4.2: Semua kemungkinan interpolasi

Interpolasi	*Node* yang digunakan	#(interpolasi)
linear	$(1.4, 1.6)$	1
kuadrat	$(1.3, 1.4, 1.6), (1.4, 1.6, 1.8)$	2
kubik	$(1.1, 1.3, 1.4, 1.6), (1.3, 1.4, 1.6, 1.8), (1.4, 1.6, 1.8, 2.0)$	3
derajat 4	$(1.0, 1.1, 1.3, 1.4, 1.6), (1.1, 1.3, 1.4, 1.6, 1.8)$ $(1.3, 1.4, 1.6, 1.8, 2.0)$	3
derajat 5	$(1.0, 1.1, 1.3, 1.4, 1.6, 1.8), (1.1, 1.3, 1.4, 1.6, 1.8, 2.0)$	2
derajat 6	(1.0,1.1,1.3,1.4,1.6,1.8,2.0)	1
	Total	12

Jadi terdapat 12 kemungkinan interpolasi untuk mengaproksimasi $y = f(1.5)$. ■

Contoh 4.6. Tabel berikut menyajikan data populasi suatu negara dari 1940 sampai 1990 yang disajikan setiap 10 tahun.

/home/julan/Documents/pasted30.png

Gambar 4.6: Grafik perkembangan jumlah penduduk

Untuk 1995 ekuivalen dengan $x = 15$ dan 1981 ekuivalen dengan $x = 41$. Ini diperoleh berdasarkan selisih tahun 1940 terhadap tahun ke-0. Setelah disubtitusikan ke dalam polinomial interpolasi $P_5(x)$, kemudian dilakukan kalkulasi, maka diperoleh

$$P_5(15) = 165,622,062 \text{ jiwa dan } P_5(41) = 228,972,451 \text{ jiwa.} \quad \blacksquare$$

4.2.3 Komputasi dengan MATLAB

Berdasarkan penjelasan sebelumnya, untuk menghasilkan polinomial interpolasi maka terlebih dahulu harus mendefinisikan polinomial Lagrange. Diingatkan kembali bahwa jika terdapat $n + 1$ bilangan berlainan maka akan terdapat $n + 1$ polinomial Lagrange yang diformulasikan menurut (4.3). Selanjutnya kita menyusun kode MATLAB untuk mendefinisikan polinomial Lagrange ke k untuk data $x_0, x_1, \cdots, x_n$.

m-file untuk mendefinisikan fungsi Lagrange

```
function y = lagr1(x,datax,k)
%mendefinsikan fungsi Lagrange ke-k yang didasarkan pada datax.
n=length(datax);
%cek indeks
if k>n

    error('indeks k melebih banyak data')

end
xx=datax;xx(k)=[];%membuang komponen ke k datax
d=prod(datax(k)-xx); %penyebut pada definisi polinomial Lagrange
y = 1;
for k=1:n-1;

    ya=(x-xx(k));
    y=y.*ya;
```

```
>> y1 = lagr1(4,datax,1)

    y1 =

        0
```

Untuk Lagrange kedua

```
>> y2 = lagr1(-1,datax,2)

    y2 =

        0

>> y2 = lagr1(2,datax,2)

    y2 =

        1

>> y2 = lagr1(4,datax,2)

    y2 =

        0
```

Untuk Lagrange ketiga

```
>> y3 = lagr1(-1,datax,3)

    y3 =

        0

>> y3 = lagr1(2,datax,3)

    y3 =

        0

>> y3 = lagr1(4,datax,3)

    y3 =

        1
```

Polinomial interpolasi untuk pasangan titik $(x_k, y_k), k = 0, 1, \cdots, n$ dapat didefinisikan pada MATLAB dengan m-file berikut. Kode ini sebagai implementasi formula (4.6).

/home/julan/Documents/pasted33.png

Gambar 4.7: Grafik polinomial interpolasi

Hitunglah aproksimasi nilai y yang berkaitan dengan $x_t = 3.5767$ dengan menggunakan polinomial interpolasi derajat 1, derajat 3, derajat 4, dan derajat 5.

Penyelesaian. Untuk masing-masing derajat menggunakan `datax` dan `datay` yang berbeda.

Untuk derajat 1: menggunakan `datax=[3.4728, 3.9741]` sebab $x_t = 3.5767$ terletak di dalamnya. Pasangannya adalah `datay=[-1.2709,-1.4126]`. Selanjutnya pada *command window* MATLAB diberikan perintah:

```
>> datax = [3.4728 3.9741]; datay = [-1.2709, -1.4126];
>> yt = poli_inter1(3.5767,datax,datay)
yt =

    -1.3003
```

Ini berarti nilai aproksimasi yang dimaksud adalah -1.3003.

Untuk derajat 3: dibutuhkan 4 pasangan titik yang melingkupi $x_t = 3.5767$, salah satunya adalah pasangan `datax=[2.6130,3.1066,3.4728,3.9741]` dengan pasangannya `datay=[-0.3591,-0.9644,-1.2709,-1.4126]`. Mengikuti langkah sebelumnya, yaitu

```
>> datax = [ 2.6130,3.1066,3.4728,3.9741];
>> datay =[-0.3591,-0.9644,-1.2709,-1.4126];
>> yt = poli_inter1(3.5767,datax,datay)
yt =

    -1.3293
```

Untuk derajat 4: gunakan pasangan data

`datax`= $[2.6130, 3.1066, 3.4728, 3.9741, 4.4007]$ dan
`datay=[-0.3591,-09644,-1.2709,-1.4126,-1.2585]`. Seperti biasa kita lakukan perintah berikut.

Selanjutnya bila dievaluasi di $x = x_2$ maka diperoleh

$$\begin{aligned}
f(x_2) &= f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\
&\leftrightarrow \quad a_2(x_2 - x_0)(x_2 - x_1) = f(x_2) - f(x_0) - \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) \leftrightarrow \\
a_2 &= \frac{f(x_2) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)} - \frac{f(x_1) - f(x_0)}{(x_2 - x_1)(x_1 - x_0)} \\
&= \frac{\frac{f(x_2)-f(x_0)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{(x_2-x_1)(x_1-x_0)}(x_2 - x_0)}{x_2 - x_0} \\
&= \frac{\frac{f(x_2)-f(x_1)+f(x_1)-f(x_0)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{(x_2-x_1)(x_1-x_0)}(x_2 - x_0)}{x_2 - x_0} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} + \frac{f(x_1)-f(x_0)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{(x_2-x_1)(x_1-x_0)}(x_2 - x_0)}{x_2 - x_0} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} + \frac{f(x_1)-f(x_0)}{(x_1-x_0)}\left(\frac{x_1-x_0}{x_2-x_1} - \frac{x_2-x_0}{x_2-x_1}\right)}{x_2 - x_0} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{(x_1-x_0)}}{x_2 - x_0}.
\end{aligned}$$

Jadi diperoleh koefisien $a_2 = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{(x_1-x_0)}}{x_2-x_0}$.

4.3.1 Pengertian Selisih Terbagi

Sebelum diberikan bentuk umum koefisien a_k pada (4.7), diperkenalkan terlebih dahulu istilah selisih terbagi sebagai berikut.

Definisi 4.2. Misalkan ada $n + 1$ buah pasangan titik (x_i, y_i) dengan $y_i = f(x_i)$ $i = 0, 1, \cdots, n$ untuk sebuah fungsi f yang diketahui, maka diperoleh selisih terbagi sebagai berikut.

- **selisih terbagi tingkat nol**, ada $n + 1$ nilai, yaitu

$$f[x_i] := f(x_i),\ i = 0, 1, \cdots, n.$$

  Ini tidak lain adalah nilai fungsi f di titik x_i, $i = 0, 1, \cdots, n$.

TABEL 4.4: Daftar selisih terbagi

x	Tingkat nol	Tingkat satu	Tingkat dua	Tingkat tiga
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0, x_1]$		
x_2	$f[x_2]$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
x_3	$f[x_3]$	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$

Teorema 4.2. *Jika diberikan $n+1$ pasangan titik berlainan $(x_i, y_i), i = 0, 1, \cdots, n$ di mana $y_i = f(x_i)$ maka polinomial yang didefinisikan sebagai berikut*

$$P_n(x) := f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1}) \tag{4.8}$$

merupakan polinomial interpolasi, yaitu $P_n(x_k) = f(x_k)$. Selanjutnya polinomial ini disebut ***formula selisih bagi Newton****.*

Bukti. Buktinya menggunakan metode induksi matematika pada n dengan prosesnya cukup rumit sehingga tidak diberikan di sini. Bagi Anda yang tertarik dapat melihatnya beberapa buku teks analisis numerik lanjut, misalnya Kress (1998) dalam [**?**]. ■

Berdasarkan Teorema ini, nilai koefisien a_k pada (4.7) tidak lain adalah selisih terbagi ke k terhadap titik $a_0, a_1, \cdots a_k$, yaitu

$$a_k = f[a_0, a_1, \cdots, a_k] \tag{4.9}$$

Contoh 4.8. Misalkan diketahui pasangan data berikut

x	0.0	0.2	0.4	0.6	0.8
$f(x)$	1.0000	1.2214	1.4918	1.8221	2.2255

Gunakan formula selisih terbagi Newton untuk menemukan polinomial interpolasi data ini, kemudian aproksimasilah nilai $f(0.75)$.

Penyelesaian. Dalam soal ini ada 5 pasangan data, yaitu $x_0 = 0.0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$ dan $x_4 = 0.8$ dengan pasangannya $y_0 = 1.0000, y_2 = 1.4918, y_3 = 1.8221$ dan $y_4 = 2.2255$.
Selisih terbagi tingkat nol ada 5 nilai, yaitu

$$f[x_0] = 1.000, f[x_1] = 1.2214, f[x_2] = 1.4918, f[x_3] = 1.8221, f[x_4] = 2.2255.$$

Proses penghitungan selisih terbagi dapat dengan cepat dilakukan langsung menggunakan tabel ini, yaitu dengan cara menggunakan data pada kolom sebelumnya. Diperhatikan angka 1.1070 diperoleh dari

$$\frac{1.2214 - 1.0000}{0.2 - 0.0}.$$

Juga angka 0.7487 diperoleh dari

$$\frac{1.6515 - 1.3520}{0.6 - 0.2},$$

dan seterusnya. Dilihat dari formulasi (4.8) maka hanya hanya bilangan pertama pada tiap-tiap kolom saja (cetak tebal) yang berkontribusi sebagai koefisien pada polinomial interpolasi, yaitu

$$\begin{aligned} P_4(x) &= 1 + 1.1070(x-0) + 0.6125(x-0)(x-0.2) + 0.2270(x-0)(x-0.2) \\ &\quad (x-0.4) + 0.0600(x-0)(x-0.2)(x-0.4)(x-0.6). \end{aligned}$$

Aproksimasi $f(0.75)$ diperoleh dengan memasukkan nilai $x = 0.75$ ke dalam $P_4(x)$ sehingga diperoleh

$$f(0.75) \approx P_4(0.75) = 2.1152. \quad \blacksquare$$

Contoh 4.9. Dengan menggunakan Tabel selisih terbagi sebelumnya, aproksimasilah $f(0.75)$ dengan menggunakan interpolasi linear, kuadratik dan kubik.

Penyelesaian. Perhatikan kembali tabel hasil terbagi sebelumnya dan lihat bilangan menurut rute: dari kiri atas ke kanan bawah.

x_i	tingkat nol	tingkat satu	tingkat dua	tingkat tiga	tingkat empat
0.0	1.0000				
0.2	1.2214	1.1070			
0.4	1.4918	1.3520	0.6125		
0.6	1.8221	1.6515	0.7487	0.2270	
0.8	2.2255	2.0170	0.9137	0.2750	0.0600

Karena 0.75 terletak di antara 0.6 dan 0.8 maka gunakan nilai selisih bagi 1.8221 dan 2.0170, sehingga polinomial interpolasi linearnya adalah

$$P_1(x) = 1.8221 + 2.0170(x - 0.6),$$

Penyelesaian. Definisikan fungsi $f(x) := \ln(1+x)$. Akan diaproksimasi nilai $f(0.45)$. Pertama dibangun tabel selisih terbagi sebagai berikut

x_i	tingkat nol	tingkat satu	tingkat dua
0	0		
0.6	0.47	0.783333	
0.9	0.642	0.57333	-0.233333

Polinomial interpolasi yang bersesuaian adalah sebagai berikut

$$P_2(x) = 0.783333(x-0) - 0.23333(x-0)(x-0.6).$$

Substitusi $x = 0.45$ diperoleh $\ln(1.45) \approx P_2(0.45) = 0.36825$. Kesalahan nyata adalah

$$E_{nyata} = |0.36825 - 0.37156| = 0.003312.$$

Untuk estimasi kesalahan diperlukan derivatif ketiga $f'''(x)$. Dengan penjabaran sederhana diperoleh $f'''(x) = \frac{2}{(1+x)^3}$. Berdasarkan Teorema 4.3, diperoleh estimasi berikut

$$|f(0.45) - P_2(0.45)| = \frac{1}{3!}\frac{2}{(1+\xi)^3}|(0.45-0)(0.45-0.6)(0.45-0.9)| = \frac{0.010125}{(1+\xi)^3}.$$

Karena $\xi \in [0,1]$ maka $\frac{1}{(1+\xi)^3} \leq \frac{1}{(1+0)^3} = 1$. Batas atas kesalahannya adalah

$$|f(0.45) - P_2(0.45)| = \frac{0.010125}{(1+\xi)^3} \leq 0.010125.$$

Jadi estimasi kesalahan $E_{est} = 0.010125$ masuk akal karena masih di atas kesalahan nyatanya, yaitu 0.003312. ■

4.3.3 Komputasi dengan MATLAB

Masalah komputasi yang berkaitan dengan metode selisih terbagi Newton ini adalah sebagai berikut.

- bagaimana menentukan semua nilai selisih terbagi untuk semua tingkat jika diberikan pasangan titik $(x_k, f(x_k)), k = 0, 1, \cdots, n$.
- bagaimana mendefinisikan polinomial interpolasi (4.8).

```
    v=D(j-1:end,j-1);
    a=v(2:end)-v(1:end-1);
    b=datax(j:end)-datax(1:end-j+1);
    b=b' d=a./b;
    D(j:end,j)=d;

end
```

Untuk mengeksekusi program ini dibutuhkan masukan berupa **datax** dan **datay**. Baiklah, gunakan saja data yang sudah pernah dihitung semua nilai hasil terbaginya, yaitu

```
>> datax=0:0.2:0.8;
>> datay=[1 1.2214 1.4918 1.8221 2.2255];
>> D=hbagi(datax,datay)
D =
1.0000      0      0      0      0
1.2214 1.1070      0      0      0
1.4918 1.3520 0.6125      0      0
1.8221 1.6515 0.7487 0.2271      0
2.2255 2.0170 0.9137 0.2750 0.0599
```

Ternyata hasilnya sama seperti yang telah diperoleh sebelumnya, kecuali ada perbedaan sangat kecil yang disebabkan oleh kesalahan pembulatan, misalnya antara 0.2271 dan 0.2270, antara 0.0599 dan 0.0600. Ini menunjukkan kode MATLAB ini bekerja dengan sempurna.

Selanjutnya dengan menggunakan data pada diagonal utama matriks ini dapat didefinisikan polinomial interpolasi sebagai berikut.

```
function y = poli_inter2(x,datax,datay)
D = hbagi(datax,datay);
n=length(datax);
y = D(1,1);
yy=1;
for k=2:n;

    p=(x-datax(k-1));
    yy=yy*p;
    y=y+D(k,k)*yy;

end
```

4. Umumnya bila terdapat $n+1$ buah pasangan titik (x_i, y_i), $i = 0, 1, 2, \cdots, n$ dengan $x_i \neq x_j, i \neq j$ maka dapat dibangun polinomial interpolasi berderajat paling tinggi n. Artinya derajat polinomial interpolasi dapat saja kurang dari n. Diberikan 6 pasangan data berikut

x_i	-2	-1	0	1	2	3
$y_i = f(x_i)$	1	4	11	16	13	-4

   a) Buktikan bahwa polinomial interpolasi yang terbentuk hanya berderajat 3.

   b) Verifikasilah hasil yang didapat dengan cara menggambar grafik fungsi pada MATLAB.

5. Diberikan data jumlah penduduk dalam ribuan jiwa.

Tahun	1920	1930	1940	1950	1960	1970	1980	1990
Populasi	106021	123203	132165	151326	179323	203302	226542	249633

   a) Bangunlah polinomial interpolasi derajat 6 dengan menggunakan data pada 1920 sampai dengan pada 1980. Gunakan untuk mengaproksimasi jumlah penduduk tahun 1990. Berapa besar kesalahan nyatanya. Gunakan juga ukuran kesalahan relatif.

   b) Bangunlah polinomial interpolasi derajat 6 dengan menggunakan data tahun 1930 sampai dengan pada 1990. Gunakan untuk mengaproksimasi jumlah penduduk pada 1920. Berapa besar kesalahan nyatanya. Gunakan juga ukuran kesalahan relatif.

   Teknik mengaproksimasi data di luar rentang data yang ada disebut ekstrapolasi. Gunakan m-file yang sudah ditulis sebelumnya. Sajikan hasil yang diperoleh dalam bentuk visual grafik.

6. Bila diketahui formula selisih terbagi Newton untuk fungsi f pada *node* $x_0 = 0, x_1 = 0.25, x_2 = 0.5$ dan $x_3 = 0.75$ adalah

$$P_3(x) = 1 + 4x + 4x(x - 0.25) + \frac{16}{3}x(x - 0.25)(x - 0.5),$$

   tentukan nilai $f(0.75)$.

Bukti. Lihat Kress (1998) dalam [?]. ■

Dalam kasus di mana fungsi $f(x)$ diketahui terdiferensial dengan $f(x_i) = y_i'$ untuk setiap $i = 0, 1, \cdots, n$ maka formula (4.14) menjadi

$$H_{2n+1}(x) = \sum_{k=0}^{n} \left[y_k H_k^0(x) + f'(x_k) H_k^1(x) \right].$$

Walaupun formulasi interpolasi Hermite pada (4.14) kelihatannya jelas, namun dalam komputasinya akan kesulitan dalam menentukan formula untuk $L_k'(x)$, yaitu derivatif dari fungsi Lagrange ke k terutama bila derajatnya cukup tinggi. Untuk itu pendekatan selisih terbagi akan digunakan untuk menyajikan polinomial interpolasi Hermite.

Berkaitan dengan titik $x_0, x_1, \cdots, x_n$ didefinisikan barisan baru $z_0, z_1, \cdots, z_{2n+1}$ dengan

$$z_{2k} = z_{2k+1} = x_k \text{ untuk setiap } k = 0, 1, \cdots, n.$$

Jadi barisan (z_n) yang terbentuk dapat disajikan secara eksplisit sebagai berikut

$$\begin{aligned} z_0 &= x_0,\ z_1 = x_0 \\ z_2 &= x_1,\ z_3 = x_1 \\ &\vdots \\ z_{2n} &= x_n,\ z_{2n+1} = x_n. \end{aligned}$$

Karena $z_{2k} = z_{2k+1} = x_k$ maka tidaklah mungkin mendefinisikan selisih terbagi $f[z_{2k}, z_{2k+1}]$ karena akan membagi dengan bilangan nol. Berdasarkan Teorema nilai rata-rata (TNR) pada Kalkulus, selalu ada $\xi_k \in (x_k, x_{k+1})$ sehingga $f[x_k, x_{k+1}] := \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} = f'(\xi_k)$. Untuk itu cukup beralasan dengan mengambil $f[z_{2k}, z_{2k+1}] := \lim_{x_{k+1} \to x_k} f[x_k, x_{k+1}]$ sehingga digunakan definisi berikut untuk $f[z_{2k}, z_{2k+1}]$.

$$f[z_{2k}, z_{2k+1}] := f'(x_k).$$

Selanjutnya, polinomial interpolasi Hermite dibangun dalam bentuk selisih terbagi terhadap titik-titik $z_0, z_1, \cdots z_{2n+1}$ sebagai berikut

$$\begin{aligned} H_{2n+1}(x) &= a_0 + a_1(x - z_0) + a_2(x - z_0)(x - z_1) \\ &\quad + a_3(x - z_0)(x - z_1)(x - z_2) + \cdots \\ &\quad + a_{2n+1}(x - z_0)(x - z_1) \ldots (x - z_{2k}). \end{aligned} \tag{4.15}$$

z_k	$f(z)$	Tingkat 1	Tingkat 2	Tingkat 3	Tingkat 4	Tingkat 5
$z_0 = 0$	0					
$z_1 = 0$	0	22				
$z_2 = 3$	67	22.3333	0.1111			
$z_3 = 3$	67	23	0.2222	0.0370		
$z_4 = 5$	115	24	0.5	0.0556	0.0037	
$z_5 = 5$	115	24	0	-0.25	-0.0611	-0.013
$z_6 = 8$	186	23.6667	-0.1111	-0.0222	0.0456	0.0133
$z_7 = 8$	186	22	-0.5556	-0.1482	-0.0252	-0.0141
$z_8 = 13$	280	18.8	-0.64	-0.0106	0.0172	0.0042
$z_9 = 13$	280	21	0.44	0.216	0.0283	0.0014

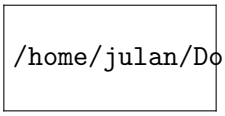
Sampai di sini masih tersisa selisih terbagi tingkat 6, tingkat 7, tingkat 8, dan tingkat 9. Angka-angka ini sangat rumit bila dikerjakan dengan cara manual. Misalnya, 22.3333 diperoleh dari $\frac{67-0}{3-0}$, 0.0556 diperoleh dari hasil $\frac{0.5-0.2222}{5-0}$, dan seterusnya.

Karena itu akan lebih baik jika komputasi dilakukan dengan menggunakan MATLAB.

4.4.2 Komputasi dengan MATLAB

Untuk itu terlebih dulu perlu disusun kode MATLAB yang dibutuhkan. Seperti sebelumnya, sajikan semua selisih terbagi dalam bentuk matriks segitiga bawah. Berikut ini adalah kode atau m-file MATLAB yang dapat digunakan untuk mendapatkan matriks selisih terbagi Hermite.

```
function D = hbagi_hermite(datax,datay,ddatay)
%menghasilkan semua hasil terbagi untuk polinomial Hermite
n=length(datax);
dataz=zeros(1,2*n);
dataz(1:2:end-1)=datax;
dataz(2:2:end)=datax;
datazy=zeros(1,2*n);
datazy(1:2:end-1)=datay;
datazy(2:2:end)=datay;
D=zeros(2*n,2*n);
D(:,1)=datazy';
D(2:2:end,2)=ddatay';
```



Gambar 4.8: Interpolasi Hermite

```
y = D(1,1); yy=1;
for k=2:2*n;

    p=(x-dataz(k-1));
    yy=yy.*p;
    y=y+D(k,k)*yy;

end
```

Untuk menjawab pertanyaan pada contoh di atas, berikan perintah berikut pada *command window.*

```
>> datax=[0 3 5 8 13];
>> datay=[0 67 115 186 280];
>> ddatay=[22 23 24 22 21];
>> y = poli_hermite(10,datax,datay,ddatay)
y =

    217.5630
```

Ini berarti pada akhir detik ke-10 mobil tersebut diprediksi telah menempuh jarak 217.5630 meter berdasarkan aproksimasi dengan menggunakan polinomial Hermite. Untuk mengetahui dan meyakinkan bahwa polinomial ini menginterpolasi pasangan titik waktu versus jarak yang diberikan sebaiknya gambarkan grafiknya dengan perintah berikut.

```
>> x=0:0.01:13;
>> y = poli_hermite(x,datax,datay,ddatay);
>> subplot(1,2,1)
>> plot(x,y,datax,datay,'*')
```

Diperoleh grafik seperti tertera pada Gambar 4.8. Ternyata pasangan waktu versus jarak, dan pasangan waktu versus kecepatan benar-benar dilewati oleh kedua polinomial ini.

Untuk mengetahui apakah mobil pernah melewati batas kecepatan maksimum, perhatikan Gambar 4.8 panel kanan. Berdasarkan grafik ini ada beberapa saat mobil tersebut berjalan

Contoh 4.12. Diberikan pasangan `datax=[1.9862,2.3743,3.265,3.4146,5.5991]` dan `datay=[0.6478,9.8833,5.8279,4.2350,5.1551]`. Akan ditentukan polinomial derajat 4 yang menginterpolasi pasangan titik ini. Kemudian hasilnya digunakan untuk mengaroksimasi nilai y_t yang berkaitan dengan $x_t = 3$.

Penyelesaian. Pada *command window* cukup dilakukan langkah-langkah berikut

```
>>datax=[1.9862,2.3743,3.265,3.4146,5.5991];
>>datay=[0.6478,9.8833,5.8279,4.2350,5.1551];
>>p=polyfit(datax,datay,4)
p=

    26.0098 -277.8787 1077.6496 -1788.9280 1075.0403
```

Sampai di sini telah didapatkan koefisien polinomial interpolasi yang untuk pasangan data tersebut. Bila ditulis secara eksplisit maka polinomial yang diperoleh sebagai berikut

$$P_4(x) = 1075.0403 - 1788.9280x + 1077.6496x^2 - 277.8787x^3 + 26.0098x^4.$$

Ingat koefisien yang dihasilkan MATLAB digunakan dalam urutan menurun sebagai koefisien dalam derajat monomial. Aproksimasi nilai y_t berkaitan dengan $x_t = 3$ cukup dengan perintah

```
>>xt=3;
>>yt=polyval(p,xt)
yt =

    11.1694
```

Untuk meyakinkan bahwa polinomial yang dihasilkan ini benar-benar menginterpolasi pasangan titik yang diberikan sebaiknya gambarkan grafiknya sebagai berikut, juga sertakan noktah untuk titik (x_t, y_t).

```
>>t=1.75:0.05:3.75;
>>y=polyval(p,t);
>>plot(t,y,'*',xt,yt,'s');grid
```

Hasilnya disajikan pada Gambar 4.9. Berdasarkan tampilan ini, ternyata memang benar bahwa polinomial yang dihasilkan menginterpolasi pasangan titik yang diberikan yang ditandai oleh $*$, sedangkan aproksimasi titik yang dimaksud ditandai oleh ★.

Walaupun MATLAB sudah menyiapkan fasilitas instant untuk urusan interpolasi, namun disarankan agar Anda memiliki pengetahuan tentang dasar teoretis yang melatarbelakanginya. Dengan cara ini sikap kritis akan tumbuh sehingga dapat meningkatkan kemampuan dalam penelitian.

5 HITUNG NUMERIK DIFERENSIAL DAN INTEGRAL

5.1 Pendahuluan

Diferensial dan integral merupakan topik utama pada kuliah kalkulus di mana keduanya dapat dipandang sebagai dua operasi yang saling berkebalikan. Khusus pada hitung integral tertentu biasanya didasarkan pada teorema fundamental kalkulus, yaitu

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

di mana F suatu fungsi yang bersifat $F'(x) = f(x)$ untuk setiap $x \in (a, b)$ dan disebut antiderivatif fungsi f. Sebagai contoh, fungsi $f(x) = x^2$ mempunyai antiderivatif $F(x) = \frac{1}{3}x^3$ sehingga diperoleh

$$\int_a^b x^2\,dx = \left[\frac{1}{3}x^3\right]_a^b = \frac{1}{3}\left(b^3 - a^3\right).$$

Jadi nilai eksak integral ini dapat diperoleh dengan mudah. Bila antiderivatif suatu fungsi tidak dapat ditemukan dalam bentuk eksplisit maka teknik integrasi ini tidak dapat digunakan. Sebagai contoh, $\int_0^1 e^{-x^2}\,dx$ atau $\int_0^{\pi/2} \sqrt{1 + \cos^2 x}\,dx$ tidak dapat diselesaikan dengan menggunakan cara ini karena antiderivatif fungsi ini tidak dapat ditemukan secara eksplisit.

Berdasarkan tampilannya, fungsi dapat dikelompokkan menjadi dua macam yaitu fungsi dengan formula eksplisit, seperti $f(x) = \sin x$, $g(x) = x^2 + 2x + 4$, $h(x) = x \ln x + e^x$ dan lain sebagainya, dan fungsi tabular. Fungsi tabular hanya berupa pasangan data diskrit dalam bentuk tabel. Biasanya diperoleh dari hasil pengukuran. Fungsi jarak $s(t)$ terhadap variabel waktu t berikut tidak menampilkan formula eksplisit, namun hanya berupa tabel hasil pengukuran untuk beberapa nilai t tertentu.

Untuk derivasi formula aproksimasi derivatif, sebagian referensi menggunakan polinomial interpolasi dan sebagian lainnya menggunakan ekspansi Taylor. Pada buku ini akan digunakan pendekatan kedua, yaitu menggunakan ekspansi Taylor.

Teorema 5.1. *Andaikan $f \in \mathcal{C}^n[a,b]$ dan $f^{(n+1)}$ ada pada $[a,b]$. Misalkan $x_0 \in [a,b]$. Untuk setiap $x \in [a,b]$ selalu terdapat $\xi(x)$ di antara x_0 dan x sehingga*

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n}_{P_n(x)} + R_n(x) \tag{5.1}$$

di mana $R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)^{n+1}$ disebut suku sisa.

Bukti. Lihat Bartle and Sherbert (1993) dalam [**?**]. ■

Teorema Taylor ini memberikan bentuk ekspansi fungsi f menjadi polinomial dalam suku-suku $(x-x_0)$. Berbeda dari polinomial interpolasi yang sudah dipelajari sebelumnya, polinomial Taylor $P_n(x)$ di sini hanya menginterpolasi satu titik x_0, tetapi ia menginterpolasi derivatif semua tingkat fungsi f di titik ini, yaitu

$$P_n(x_0) = f(x_0),\ P_n^{'}(x_0) = f''(x_0),\ P_n^{''}(x_0) = f''(x_0), \cdots$$

5.2.1 Aproksimasi Derivatif Pertama

Formula dua titik

Formula ini menggunakan dua titik x dan x_0+h, atau x_0-h dan x_0 untuk mengaproksimasi nilai $f'(x_0)$. Pada ekspansi Taylor diambil $x = x_0+h$, $h > 0$ dan $n = 1$ sehingga diperoleh

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{f''(\xi)}{2}h^2$$

untuk suatu $\xi \in (x_0, x_0+h)$. Setelah persamaan ini diselesaikan untuk $f'(x_0)$ diperoleh

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \frac{f''(\xi)}{2}h.$$

Diperoleh aproksimasi **selisih maju** (*forward difference*)

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0-h)}{2h} \tag{5.6}$$

yang memberikan order konvergensi $\mathcal{O}(h^2)$. Dikatakan formula tiga titik karena formula ini sesungguhnya melibatkan tiga titik $x_0 - h, x_0$ dan $x_0 + h$ walaupun pada formula akhir (5.6) titik x_0 tidak muncul. Formula ini disebut juga aproksimasi **selisih terpusat** (*centered difference*). Berdasarkan order konvergensi, formula selisih terpusat lebih baik dari formula selisih maju dan selisih mundur dalam hal kecepatan konvergensinya. Karena itu akurasinya juga dipastikan membaik seiring berkurangnya ukuran langkah (*step size*) h. Masih ada satu lagi formula tiga titik yang diberikan dalam [**?**], yaitu menggunakan $x_0, x_0 + h$ dan $x_0 + 2h$.

$$f'(x_0) = \frac{1}{2h}\left(-3f(x_0) + 4f(x_0+h) - f(x_0+2h)\right) \tag{5.7}$$

dengan kesalahan $E = \frac{h^2}{3}f'''(\xi)$, $\xi \in (x_0, x_0+2h)$, yakni dengan order $\mathcal{O}(h^2)$. Formula (5.7) disebut juga formula **tiga titik tepi**. Formula ini sering digunakan untuk mengaproksimasi derivatif di tepi atau ujung interval. Sebagai contoh, bila x_0 titik tepi paling kiri interval digunakan $h > 0$, sedangkan bila x_0 titik paling kanan interval digunakan $h < 0$.

Formula lima titik

Formula ini menggunakan lima titik dan memberikan kesalahan dengan order $\mathcal{O}(h^4)$. Ada 2 jenis formula lima titik ini, yaitu terpusat dan tepi. Untuk formula **lima titik terpusat** diberikan dalam [**?**] sebagai berikut.

$$f'(x_0) \approx \frac{1}{12h}\left(f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)\right) \tag{5.8}$$

dengan kesalahan $E = \frac{h^4}{30}f^{(5)}(\xi)$, $\xi \in (x_0-2h, x_0+2h)$. Sedangkan **formula lima titik tepi** diberikan sebagai

$$f'(x_0) \approx \frac{1}{12h}\left(-25f(x_0) + 48f(x_0+h) - 36f(x_0+2h) + 16f(x_0+3h) - 3f(x_0+4h)\right) \tag{5.9}$$

dengan kesalahan $E = \frac{h^4}{5}f^{(5)}(\xi), \xi \in (x_0, x_0+4h)$.

Berdasarkan uraian di atas, formula lima titik memberikan order konvergensi paling cepat. Tetapi order konvergensi ini terjamin dapat dicapai jika fungsi f terdiferensial sampai

Tiga aproksimasi lainnya dapat dihitung sendiri dengan mudah. Hanya ada satu kemungkinan untuk formula lima titik, yaitu terpusat dengan $h = 0.1$

$$f'(0.2) \approx \frac{1}{12(0.1)} \left(f(1.8) - 8f(1.9) + 8f(2.1) - f(2.2)\right) = 22.166999.$$

Untuk mengetahui estimasi kesalahan dibutuhkan batas atas nilai derivatif. Amati

$$f(x) = xe^x \rightarrow f'(x) = (1+x)e^x \rightarrow f''(x) = (2+x)e^x, \cdots, f^{(n)}(x) = (n+x)e^x.$$

Perhatikan untuk aproksimasi selisih maju dengan $h = 0.1$. Dengan menggunakan formula kesalahan $E = -\frac{f''(\xi)}{2}h, \xi \in (2.0, 2.1)$ diperoleh estimasi kesalahan

$$|E| = \left|-\frac{f''(\xi)}{2}h\right| = \left(\frac{(2+\xi)e^{\xi}}{2}\right)(0.1) \leq (2+2.1)e^{2.1}(0.05) = 1.6741.$$

Telah digunakan sifat bahwa derivatif kedua $f''(x) = (2+x)e^x$ merupakan fungsi naik sehingga nilai maksimumnya tercapai di ujung kanan interval, dalam hal ini di $\xi = 2.1$. Jadi secara teoretis, kesalahan aproksimasi dengan menggunakan formula selisih maju tidak akan melebihi 1.6741. Faktanya, karena derivatif eksaknya adalah $f'(2.0) = (1+2.0)e^2 = 22.167168$ maka kesalahan nyatanya adalah

$$E_{\text{nyata}} = 23.708446 - 22.167168 = 1.541278.$$

Ternyata kesalahan nyata masih di bawah estimasi. Berikutnya analisis kesalahan metode selisih terpusat dengan $h = 0.1$. Estimasi kesalahannya adalah

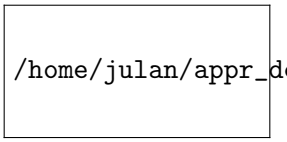
$$|E| = \frac{f'''(\xi)}{6}h^2 = \frac{1}{6}(3+\xi)e^{\xi}(0.1)^2 \leq \frac{1}{6}(3+2.1)e^{2.1}(0.1)^2 = 0.069412.$$

Faktanya, kesalahan nyatanya adalah

$$E_{\text{nyata}} = 22.228790 - 22.167168 = 0.059622.$$

Sempurna, kesalahan nyata kurang dari kesalahan teoretis. ■

Silakan mencoba untuk hasil aproksimasi lainnya.



Gambar 5.1: Grafik aproksimasi derivatif dan derivatif eksaknya

Penulisan m-file MATLAB untuk kasus 1

```
function[x,dermun,dermaj,derpus]=derfun(a,b,h)
x = a:h:b;
n = length(x);
for k = 1:n
dermun(k)=(feval('fun',x(k))-feval('fun',x(k)-h))/h;
dermaj(k)=(feval('fun',x(k)+h)-feval('fun',x(k)))/h;
derpus(k)=(feval('fun',x(k)+h)-feval('fun',x(k)-h))/(2*h);
end
%fungsi yang didieferensialkan function
y = fun(x)
y = exp(-x.^2);
```

Dengan memberikan perintah pada *command window*

```
>>[x,dermun,dermaj,derpus]=derfun(-3,3,0.2);
>>dy = -2*x.*exp(-x.^2);
>>plot(x,dy,x.dermun,':.',x,dermaj,':.',x,derpus,':.');
```

maka diperoleh grafik seperti ditunjukkan pada Gambar 5.1.

Di sini dy adalah implementasi dari $y' = -2xe^{-x^2}$, yaitu diferensial eksak fungsi $y = e^{-x^2}$. Dilihat dari polanya, penulisan kode aproksimasi derivatif dapat menggunakan operasi *array* atau vektor tanpa harus menggunakan kendali loop `for`. Penggunaan operasi *array* pada kode MATLAB lebih menguntungkan karena waktu eksekusinya jauh lebih cepat daripada jika menggunakan kendali loop. Idenya, pada *array* $x = \{a = x_0, x_1, \cdots, x_n = b\}$ dibentuk *array* baru $xx = \{a - h, a = x_0, x_1, \cdots, x_n = b, b + h\}$ yaitu dengan menambah *node* $a - h$ di kiri a dan *node* $b + h$ di kanan b. Kemudian, didefinisikan *array* $fx = \{f(x_0), f(x_1), \cdots, f(x_n)\}$ dan *array* $fxx = \{f(x_0-h), f(x_0), f(x_1), \cdots, f(x_n), f(x_n+h)\}$. Perintah

```
(fx(1:end)-fx(1:end-2))
```

Sengaja indeks titik *node* dibuat mulai dari 1 karena MATLAB tidak mengenal indeks 0, jadi harus dimulai dari 1. Diperoleh

$$\begin{aligned} f'(x_1) &\approx \frac{f(x_2)-f(x_1)}{h_1},\ f'(x_n) \approx \frac{f(x_n)-f(x_{n-1})}{h_{n-1}}, \\ f'(x_k) &\approx \frac{f(x_{k+1})-f(x_{k-1})}{h_{k-1}+h_k}, k = 2, \cdots, n-1. \end{aligned}$$

Didasarkan pada pemikiran ini, maka kode berikut disusun.

```
function [x derf] = dertab(datax,datay)
x = datax;y=datay;
n = length(x);
h = x(2:end)-x(1:end-1);
derf(1)=(y(2)-y(1))/h(1);derf(n)=(y(end)-y(end-1))/h(n-1);
for k = 2:n-1

    derf(k)=(y(k+1)-y(k-1))/(h(k-1)+h(k));

end
```

Contoh 5.2. Misalkan sebuah fungsi hanya disajikan data berikut.

```
x: 0.2   0.3   0.4   0.5   0.6   0.7   0.8
y: 1.109 1.195 1.303 1.431 1.581 1.753 1.946
```

Hitunglah aproksimasi derivatif fungsi ini pada titik-titk yang diberikan.

Penyelesaian. Dalam hal ini gunakan m-file `dertab.m` di atas dengan memasukkan data-data yang diperlukan sebagai berikut.

```
>>datax=[0.2  0.3  0.4  0.5  0.6  0.7  0.8];
>>datay=[1.109 1.195 1.303 1.431 1.581 1.753 1.946];
>>[x derf] = dertab(datax,datay)
x =
    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
derf =
    0.8600    0.9700    1.1800    1.3900    1.6100    1.8250    1.9300
```

Ini menunjukkan nilai yang diberikan oleh *array* `derf` adalah aproksimasi derivatif pada titik yang bersesuaian. Dapat ditulis dalam pasangan berikut

a) Formula dua titik (maju dan mundur)

b) Formula tiga titik (pusat dan tepi)

c) Formula lima titik (pusat dan tepi)

2. Untuk fungsi yang sama seperti soal nomor (1), lakukan ekperimen numerik untuk mengaproksimasi $f'(0.5)$ dengan ukuran langkah $h = [1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}]$ untuk ketiga metode: formula dua titik maju, formula tiga titik terpusat dan formula lima titik terpusat. Selidikilah pola peluruhan kesalahan nyata menurut penurunan h. Gambarkan pola peluruhan kesalahan ini pada bidang koordinat dengan cara membuat plot $E_{nyata}(h)$ pada sumbu Y versus $\frac{1}{h}$ pada sumbu X. Apa yang dapat Anda simpulkan dari visual ini dikaitkan dengan order konvergensi $\mathcal{O}(h^\alpha)$ seperti yang telah diberikan secara teoretis. (Petunjuk: Gunakan m-file MATLAB sebelumnya)

3. Data berikut adalah hasil pencuplikan fungsi $f(x) = 3xe^x - \cos x$ pada beberapa titik.

x	1.20	1.29	1.30	1.31	1.40
$f(x)$	11.590	13.782	14.043	14.307	16.862

a) Aproksimasilah $f''(1.3)$ dengan formula tiga titik pusat dengan $h = 0.1$. Hitunglah kesalahan nyata.

b) Aproksimasilah $f''(1.3)$ dengan formula tiga titik pusat dengan $h = 0.01$. Hitunglah kesalahan nyata.

4. Pada awal subbab ini telah disinggung bahwa aproksimasi derivatif $f'(x_0)$ dengan formula

$$f'_h(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}$$

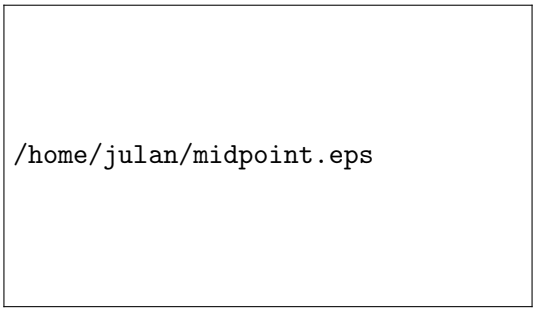
dengan pengambilan h sangat kecil dapat memengaruhi akurasi aproksimasi dikarenakan adanya kesalahan pembulatan. Lakukan ekperimen numerik pada fungsi berikut dengan memainkan $h = 10^{-n}, n = 1, 2, \cdots, 20$

a) $f(x) = \sqrt{1 + x}$, $x_0 = 1$

b) $f(x) = x \sin(x)$, $x_0 = \pi/6$

c) $f(x) = 2(\ln x)^2 + 3\cos x$, $x_0 = 2$

Bandingkan setiap hasil aproksimasi dengan nilai eksaknya. Seharusnya kesalahan yang terjadi semakin lama semakin kecil, tetapi apa yang terjadi dengan eksperimen ini?



Gambar 5.2: Ilustrasi metode midpoint

5.3.1 Formula Kuadratur Dasar

Prosedur dasar untuk aproksimasi integral tertentu suatu fungsi f pada interval $[a, b]$ adalah dengan cara memilih polinomial interpolasi yang mengaproksimasi f, kemudian mengintegralkan polinomial ini. Hasilnya digunakan sebagai aproksimasi untuk integral fungsi f semula. Ada tiga macam metode yang berkaitan dengan aturan kuadratur dasar, yaitu metode titik tengah (*midpoint*), metode trapesium dan metode Simpson.

Metode midpoint

Metode ini menggunakan polinomial interpolasi derajat nol atau fungsi konstan untuk mengaproksimasi fungsi f. Hanya menggunakan 1 absis, yaitu $x_1 = \frac{a+b}{2}$ titik tengah interval $[a, b]$. Inilah yang menyebabkan pendekatan ini disebut metode midpoint. Ilustrasinya diberikan pada Gambar 5.2. Karena itu digunakan aproksimasi

$$f(x) \approx P_0(x) = f(x_1) = f\left(\frac{a+b}{2}\right)$$

sehingga integralnya diaproksimasi oleh

$$I(f) \approx M(f) = \int_a^b P_0(x)\,dx = (b-a) f\left(\frac{a+b}{2}\right). \tag{5.13}$$

Ekspresi (5.13) merupakan bentuk (5.12) dengan $n = 1, x_1 = \frac{a+b}{2}, w_1 = b - a$. Secara geometris, metode midpoint ini mengatakan bahwa luas daerah di bawah kurva $y = f(x)$ dari $x = a$ sampai dengan $x = b$ diaproksimasi oleh luas daerah persegi dengan sisi-sisi $b - a$ dan $f\left(\frac{a+b}{2}\right)$.

Metode Simpson

Metode ini menggunakan polinomial interpolasi derajat dua atau parabola untuk mengaproksimasi fungsi yang diintegralkan. Gambar berikut memberikan ilustrasi metode Simpson.

Gambar 5.4: Ilustrasi metode Simpson

Dengan menggunakan absis $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ maka polinomial interpolasi $P_2(x)$ yang besesuaian diberikan oleh

$$P_2(x) = f(a) + f\left[a, \frac{a+b}{2}\right](x-a) + f\left[a, \frac{a+b}{2}, b\right](x-a)\left(x - \frac{a+b}{2}\right).$$

Dengan menggunakan penjabaran yang agak panjang, integral $\int_a^b P_2(x)\,dx$ dapat diselesaikan dan hasilnya digunakan untuk aproksimasi integral semula, yaitu

$$I(f) \approx S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right). \tag{5.15}$$

Ini juga merupakan bentuk (5.12) dengan ketiga absis $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ dan bobot $w_1 = \frac{b-a}{6}f(a)$,$w_2 = \frac{2}{3}(b-a), w_3 = \frac{b-a}{6}f(b)$. Secara geometris, metode Simpson mengaproksimasi integral $\int_a^b f(x)\,dx$ dengan luas daerah di bawah kurva parabola.

Contoh 5.3. Tentukan aproksimasi integral berikut dengan menggunakan metode midpoint, trapesium, dan Simpson:

$$\int_0^1 e^{-x^2}\,dx.$$

Bukti. Lihat Faires & Burden (2003) dalam [?]. ■

Teorema 5.3. *Bila $f \in \mathcal{C}^2[a,b]$ maka terdapat $\xi \in (a,b)$ sehingga metode trapesium memberikan kesalahan sebagai berikut*

$$I(f) - T(f) = -\frac{f''(\xi)}{12}(b-a)^3. \tag{5.17}$$

Bukti. Lihat Faires & Burden (2003) dalam [?]. ■

Teorema 5.4. *Bila $f \in \mathcal{C}^4[a,b]$ maka terdapat $\xi \in (a,b)$ sehingga metode Simpson memberikan kesalahan sebagai berikut*

$$I(f) - S(f) = -\frac{f^{(4)}(\xi)}{2880}(b-a)^5. \tag{5.18}$$

Bukti. Lihat Kress (1998) dalam [?]. ■

Contoh 5.4. Pada kuliah kalkulus, logaritma natural (ln) didefinisikan juga dalam bentuk integral berikut

$$\ln x := \int_1^x \frac{dt}{t} = \int_0^{x-1} \frac{dz}{z+1}.$$

Jadi nilai $\ln 2$ dapat ditentukan dengan menghitung integral berikut

$$\ln 2 = \int_0^1 \frac{dx}{x+1}.$$

Aproksimasilah nilai $\ln 2$ ini dengan ketiga metode kuadratur dasar, tentukan estimasi kesalahannya, dan bandingkan dengan kesalahan nyatanya.

Penyelesaian. Di sini dipunyai $a = 0, b = 1$ dan $f(x) = \frac{1}{x+1}$. Aproksimasinya adalah sebagai berikut. Metode trapesium memberikan hasil

$$T(f) = \frac{1-0}{2}\left(1 + \frac{1}{2}\right) = 0.75.$$

Metode Simpson memberikan hasil

$$S(f) = \frac{1}{6}\left(1 + \frac{4}{\frac{1}{2}+1} + \frac{1}{2}\right) = 0.69444.$$

5.3.3 Formula Kuadratur Bersusun

Penyusunan formula kuadratur bersusun untuk mengaproksimasi $\int_a^b f(x)\,dx$ didasarkan pada dua ide berikut, yaitu

- Semakin sempit domain integral $[a,b]$ semakin teliti hasil aproksimasi yang diperoleh,
- Bila domain integral $[a,b]$ dipartisi menjadi $x_0 := a < x_1 < x_2 < \cdots < x_n := b$ maka berlaku

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0:=a}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \cdots \int_{x_{n-1}}^{x_n:=b} f(x)\,dx \\ &= \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)\,dx\end{aligned}$$

Sebagai ilustrasi diperhatikan contoh berikut ini.

Contoh 5.5. Misal diinginkan mengaproksimasi integral $\int_0^2 e^x\,dx$ dengan metode Simpson. Untuk membandingkan hasil aproksimasinya, diberikan dulu nilai eksak integral ini, yaitu

$$\int_0^2 e^x dx = e^2 - e^0 = 6.3890561.$$

- Domain integral $[0,2]$ tidak dipecah diperoleh

  $$S(f) = \frac{1}{3}\left(e^0 + 4e^1 + e^2\right) = 6.4207278.$$

  Hasil ini memberikan kesalahan 0.0316717.

- Domain integral dipecah menjadi 2: $[0,2] = [0,1] \cup [1,2]$, sehingga $\int_0^2 e^x dx = \int_0^1 e^x dx + \int_1^2 e^x dx$. Diperoleh aproksimasi

  $$S(f) = \frac{1}{6}\left(e^0 + 4e^{0.5} + e^1\right) + \frac{1}{6}\left(e^1 + 4e^{1.5} + e^2\right) = 6.3912102.$$

  Hasil ini memberikan kesalahan 0.0021541. Wow, dengan hanya memecah menjadi dua domain integrasi, kesalahannnya dapat direduksi lebih dari 90%.

- Domain integral dipecah menjadi 4: $[0,2] = [0, \frac{1}{2}] \cup [\frac{1}{2}, 1] \cup [1, \frac{3}{2}] \cup [\frac{3}{2}, 2]$, sehingga $\int_0^2 e^x dx = \int_0^{0.5} e^x dx + \int_{0.5}^1 e^x dx + \int_1^{1.5} e^x dx + \int_{1.5}^1 e^x dx$. Diperoleh aproksimasi

  $$\begin{aligned}S(f) &= \frac{1}{12}\left(e^0 + 4e^{0.25} + e^{0.5}\right) + \frac{1}{12}\left(e^{0.5} + 4e^{0.75} + e^1\right) \\ &+ \frac{1}{12}\left(e^1 + 4e^{1.25} + e^{1.5}\right) + \frac{1}{12}\left(e^{1.5} + 4e^{1.75} + e^2\right) \\ &= 6.3891937.\end{aligned}$$

Teorema 5.5. [Teorema Nilai Antara (TNA)] *Bila f kontinu pada interval $[a,b]$ dan K bilangan di antara $f(a)$ dan $f(b)$ maka terdapat bilangan c diantara a dan b sehingga $f(c) = K$.*

Bukti. Lihat Bartle dan Sherbet (1993) dalam [?]. ■

Teorema 5.6. [Teorema Nilai Ekstrim (TNE)] *Bila f kontinu dan terbatas pada interval $[a,b]$ maka terdapat $c_{min}, c_{max} \in [a,b]$ sehingga $f(c_{min}) = \min_{x\in[a,b]} f(x) \leq f(x) \leq f(c_{max}) = \max_{x\in[a,b]} f(x)$ untuk setiap $x \in [a,b]$.*

Bukti. Lihat Bartle dan Sherbet (1993) dalam [?]. ■

Pada formula quadratur dasar, metode midpoint pada $[a,b]$ memberikan kesalahan $E = \frac{f''(\xi)}{24}(b-a)^3$ di mana $\xi \in (a,b)$. Bandingkan jika diterapkan pada interval $[x_{k-2}, x_k]$ maka kesalahannya adalah $E_k = \frac{f''(\xi_k)}{24}(x_k - x_{k-2})^3 = \frac{f''(\xi_k)}{3}h^3$. Ingat $x_k - x_{k-2} = 2h$. Diasumsikan f'' kontinu pada $[a,b]$ maka diperoleh

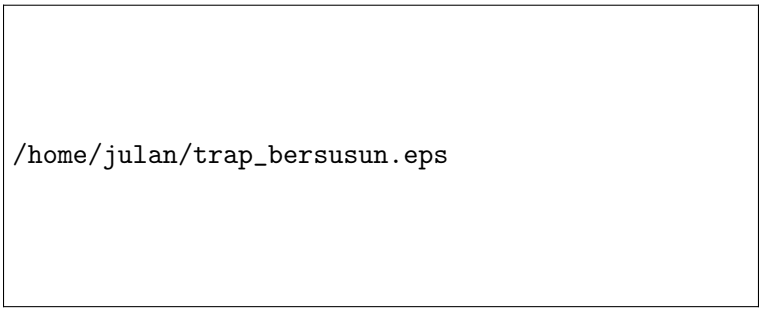
$$\min_{x\in[a,b]} f''(x) \leq f''(\xi_k) \leq \max_{x\in[a,b]} f''(x),\ k = 1, \cdots, \frac{n}{2}.$$

Dijumlahkan dari $k = 1$ sampai $k = \frac{n}{2}$,

$$\begin{aligned}
\sum_{k=1}^{n/2} \min_{x\in[a,b]} f''(x) &\leq \sum_{k=1}^{n/2} f''(\xi_k) \leq \sum_{k=1}^{n/2} \max_{x\in[a,b]} f''(x) \\
\frac{n}{2} \min_{x\in[a,b]} f''(x) &\leq \sum_{k=1}^{n/2} f''(\xi_k) \leq \frac{n}{2} \max_{x\in[a,b]} f''(x) \\
\min_{x\in[a,b]} f''(x) &\leq \frac{2}{n}\sum_{k=1}^{n/2} f''(\xi_k) \leq \max_{x\in[a,b]} f''(x).
\end{aligned}$$

Dengan asumsi f'' kontinu pada $[a,b]$ maka dengan TNA terdapat $\mu \in (a,b)$ sehingga $f''(\mu) = \frac{2}{n}\sum_{k=1}^{n/2} f''(\xi_k)$. Karena $n = \frac{b-a}{h}$ maka kesalahan dalam metode midpoint bersusun diperoleh sebagai berikut

$$\begin{aligned}
E &= \sum_{k=1}^{n/2} E_k = \frac{h^3}{3}\sum_{k=1}^{n/2} f''(\xi_k) = \frac{h^3}{3}\frac{n}{2}\left(\frac{2}{n}\sum_{k=1}^{n/2} f''(\xi_k)\right) = \frac{h^3}{6}\left(\frac{b-a}{h}\right) f''(\mu) \\
&= \frac{(b-a)h^2}{6} f''(\mu).
\end{aligned}$$



Gambar 5.6: Ilustrasi metode trapesium bersusun

Metode Simpson bersusun

Karena dibutuhkan 3 titik untuk membangun 1 aproksimasi Simpson maka dibutuhkan n genap pada pengambilan partisi seragam $a := x_0 < x_1 < x_2 < \cdots < x_{n-2} < x_{n-1} < x_n := b$, di mana $h = x_i - x_{i-1}$untuk setiap $i = 1, 2, \cdots n$. Dengan ide yang sama ketika menurunkan formula metode midpoint bersusun sebelumnya maka diperoleh tahapan aproksimasi berikut

- $S_1(f) = \frac{(x_2 - x_0)}{6}\left(f(x_0) + 4f(x_1) + f(x_2)\right) = \frac{h}{3}\left(f(x_0) + 4f(x_1) + f(x_2)\right)$,
- $S_2(f) = \frac{(x_4 - x_2)}{6}\left(f(x_2) + 4f(x_3) + f(x_4)\right) = \frac{h}{3}\left(f(x_2) + 4f(x_3) + f(x_4)\right)$,
- $S_3(f) = \frac{(x_6 - x_4)}{6}\left(f(x_4) + 4f(x_5) + f(x_6)\right) = \frac{h}{3}\left(f(x_4) + 4f(x_5) + f(x_6)\right)$,
- $\cdots$
- $S_{\frac{n}{2}}(f) = \frac{(x_n - x_{n-2})}{6}\left(f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right) = \frac{h}{3}\left(f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right)$.

Selanjutnya semua nilai ini dijumlah untuk mendapatkan

$$S(f) = \frac{h}{3}\left(f(x_0) + 4\left(f(x_1) + f(x_3) + \cdots + f(x_{n-1})\right) + 2\left(f(x_2) + f(x_4) + \cdots + f(x_{n-2})\right) + f(x_n)\right).$$

Akhirnya, metode Simpson bersusun diberikan oleh formula

$$S(f) = \frac{h}{3}\left(f(a) + f(b) + 4\sum_{k=1}^{\frac{n}{2}-1} f(x_{2k-1}) + 2\sum_{k=1}^{\frac{n}{2}-2} f(x_{2k})\right). \tag{5.23}$$

Perhatikan dengan saksama bahwa ada pengelompokan nilai fungsi f pada *node* dengan indeks genap dan *node* dengan indeks ganjil.

untuk metode midpoint. Untuk metode Trapesium diperoleh

$$\begin{aligned} T(f) &= \frac{h}{2}\left(f(a)+f(b)+2\sum_{k=2}^{n-1} f(x_k)\right) \\ &= \frac{0.5}{2}\left(f(0)+f(2)+2(f(0.5)+f(1.0)+f(1.5)\right) \\ &= 0.25\left(0+2e^{-2^2}+2(0.5e^{-0.5^2}+e^{-1^2}+1.5e^{-1.5^2})\right) = 0.4668472. \end{aligned}$$

Untuk metode Simpson diperoleh

$$\begin{aligned} S(f) &= \frac{h}{3}\left(f(a)+f(b)+4\sum_{k=1}^{\frac{n}{2}-1} f(x_{2k-1})+2\sum_{k=1}^{\frac{n}{2}-2} f(x_{2k})\right) \\ &= \frac{0.5}{3}\left(f(0)+f(2)+4(f(0.5)+f(1.5))+2(f(1))\right) \\ &= 0.4937312. \end{aligned}$$

Untuk $h = 0.25$ bersesuaian dengan $n = 8$, yaitu absis $x_0 = 0, x_1 = 0.25, x_2 = 0.5, x_3 = 0.75, x_4 = 1, x_5 = 1.25, x_6 = 1.5, x_7 = 1.75, x_8 = 2$. Diperoleh

$$M(f) = 2(0.25)\left(f(0.25)+f(0.75)+f(1.25)+f(1.75)\right) = 0.5030266$$

untuk midpoint. Untuk metode trapesium diperoleh

$$T(f) = \frac{0.25}{2}\left(f(0)+f(2)+2(f(0.25)+f(0.5)+\cdots+f(1.75)\right) = 0.4849369.$$

Sedangkan metode Simpson memberikan hasil

$$\begin{aligned} S(f) &= \frac{0.25}{3}[f(0)+f(2)+4(f(0.25)+f(0.75)+f(1.25)+f(1.75)) \\ &+ 2(f(0.5)+f(1)+f(1.5))] = 0.4909668. \end{aligned}$$

Untuk membandingkan akurasi aproksimasi integral ini, terlebih dulu dihitung dulu nilai eksak integral yang dimaksud, yaitu

Kuadratur	$h_1 = 0.5$		$h_2 = 0.25$		$\frac{E_2}{E_1}$
	Aproksimasi	Kesalahan	Aproksimasi	Kesalahan	
Midpoint	0.5474992	0.0566570	0.5030266	0.0121844	11.55%
Trapesium	0.4668472	0.0239950	0.4849369	0.0059053	4.99%
Simpson	0.4937312	0.0028890	0.4909668	0.0001246	0.59%

Aproksimasi integral untuk polinomial

Pada penurunan formula kuadratur telah digunakan polinomial sebagai fungsi aproksimasinya. Bila diperhatikan dengan saksama, metode midpoint memberikan hasil eksak untuk fungsi konstan, metode trapesium memberikan hasil eksak untuk fungsi linear dan metode Simpson memberikan hasil eksak untuk fungsi kuadratik.

Contoh 5.8. Diberikan beberapa fungsi polinomial berikut $f_0(x) = 2$, $f_1(x) = 2x + 1$, $f_2(x) = 3x^2 + 2x + 1$ dan $f_3(x) = x^3 + 2x + 1$. Hitunglah nilai $\int_0^1 f_k(x)\,dx$ dengan menggunakan ketiga formula kuadratur dasar. Selidikah ketepatan aproksimasi dengan nilai eksaknya.

Penyelesaian. Nilai eksak integral masing-masing fungsi adalah sebagai berikut.

$$\begin{aligned}
\int_0^1 f_0(x)\,dx &= \int_0^1 2\,dx = 2,\ \int_0^1 f_1(x)\,dx = \int_0^1 (2x+1)\,dx = \left[x^2+x\right]_0^1 = 2\\
\int_0^1 f_2(x)\,dx &= \int_0^1 (3x^2+2x+1)\,dx = \left[x^3+x^2+x\right]_0^1 = 3,\\
\int_0^1 f_3(x)\,dx &= \int_0^1 (x^3+2x+1)\,dx = \left[\frac{1}{4}x^4+x^2+x\right]_0^1 = \frac{9}{4}.
\end{aligned}$$

Nilai aproksimasi untuk masing-masing metode diperoleh sebagai berikut.

Midpoint: $M(f_0) = (1-0)f_0(0.5) = 2$, sebab $f_0(0.5) = 2$ sebuah fungsi konstan. Jadi $M(f_0)$ memberikan hasil **eksak** untuk integral fungsi konstan. Untuk fungsi f_1 diperoleh $M(f_1) = (1-0)f_1(0.5) = 2(0.5) + 1 = 2$, juga eksak. Untuk f_2, diperoleh $M(f_2) = (1-0)(f_2(0.5)) = 2.75$, tidak lagi eksak.

Trapesium: $T(f_0) = \frac{1}{2}(1-0)\,[f_0(0) + f_0(1)] = \frac{1}{2}\,[2+2] = 2$, ternyata hasilnya eksak. Kemudian, $T(f_1) = \frac{1}{2}(1-0)\,[f_1(0) + f_1(1)] = \frac{1}{2}\,[2(0)+1+2(1)+1] = 2$, ternyata masih eksak. Untuk f_2 diperoleh $T(f_2) = \frac{1}{2}(1-0)\,[f_2(0) + f_2(1)] = \frac{1}{2}\,[1+6] = \frac{7}{2}$, hasilnya tidak lagi eksak.

Simpson: Dengan mudah dapat ditunjukkan bahwa metode Simpson memberikan hasil eksak untuk fungsi f_0 dan f_1.

$$S(f_2) = \frac{1-0}{6}\,[f_2(0) + f_2(1) + 4f_2(0.5)] = \frac{1}{6}\left[1+6+4(\frac{3}{4}+1+1)\right] = 3,$$

ternyata eksak. Untuk f_3 diperoleh

$$S(f_3) = \frac{1-0}{6}\,[f_3(0) + f_3(1) + 4f_3(0.5)] = \frac{1}{6}\left[1+4+4(\frac{1}{8}+1+1)\right] = \frac{9}{4},$$

Integrasi Gauss order 2

Dibangun formula kuadratur $Q_2(f) = c_1 f(x_1) + c_2 f(x_2)$ yang mempunyai derajat akurasi 3 untuk integral

$$\int_{-1}^{1} f(x)\,dx.$$

Untuk ini disyaratkan ia memberikan hasil eksak bagi polinomial berderajat paling tinggi 3. Ini berarti ia memberikan hasil eksak bila $f(x)$ adalah 1, x, x^2 dan x^3. Bila $f(x) = 1$, berlaku $f(x_1) = f(x_2) = 1$ sehingga diperoleh

$$c_1 \cdot 1 + c_2 \cdot 1 = \int_{-1}^{1} 1\,dx = 2.$$

Untuk $f(x) = x$, berlaku $f(x_1) = x_1$dan $f(x_2) = x_2$ sehingga diperoleh

$$c_1 \cdot x_1 + c_2 \cdot x_2 = \int_{-1}^{1} x\,dx = 0.$$

Untuk $f(x) = x^2$, berlaku $f(x_1) = x_1^2$dan $f(x_2) = x_2^2$ sehingga diperoleh

$$c_1 \cdot x_1^2 + c_2 \cdot x_2^2 = \int_{-1}^{1} x^2\,dx = \frac{2}{3}.$$

Terakhir, untuk $f(x) = x^3$ diperoleh

$$c_1 \cdot x_1^3 + c_2 \cdot x_2^3 = \int_{-1}^{1} x^3\,dx = 0.$$

Diperoleh sistem persamaan taklinear 4 persamaan dengan 4 variabel c_1, c_2, x_1 dan x_2. Dengan menyelesaikan sistem ini diperoleh

$$c_1 = 1,\ c_2 = 1,\ x_1 = -\frac{\sqrt{3}}{3},\ x_2 = \frac{\sqrt{3}}{3}.$$

Dengan demikian diperoleh formula integrasi Gauss order 2 berikut

$$\int_{-1}^{1} f(x)\,dx \approx Q_2(f) = f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right). \tag{5.25}$$

Untuk batas integral secara umum $[a, b]$ digunakan transformasi variabel berikut

$$t = \frac{2x - a - b}{b - a},$$

Penyelesaian. Karena $f(x) = e^x$ dan $f^{(4)}(x) = e^x$ untuk setiap $x \in [0, 2]$. Berdasarkan ekspresi sebelumnya, diperoleh estimasi kesalahan berikut.

$$E = \frac{1}{135} f^{(4)}(\mu) = \frac{e^\mu}{135} \leq \frac{e^2}{135} = 0.0547337.$$

Ternyata kesalahan nyata 0.0209479 masih kurang dari estimasi. ■

Integrasi Gauss order n

Dengan asumsi formula kuadratur $Q_n(f)$ memberikan hasil eksak bagi integral $\int_{-1}^{1} f(x)\,dx$, yaitu

$$\sum_{k=1}^{n} c_k f(x_k) = \int_{-1}^{1} f(x)\,dx$$

dipenuhi untuk setiap f berupa $1, x, \cdots, x^{2n-1}$ maka terbentuk sistem persamaan taklinear (SPTL) dengan $2n$ variabel, yaitu $c_1, \cdots, c_n$ dan $x_1, \cdots, x_n$. Metode penyelesaian SPTL ini tidak sederhana sehingga tidak dibahas dalam buku ini. Cara lain menentukan absis dan bobot pada integrasi Gauss seperti diungkapkan oleh Burden dan Faires (2003) dalam [?] atau Kress (1998) dalam [?] adalah dengan menggunakan keluarga polinomial Legendre $\{P_0(x), P_1(x), \cdots, P_n(x), \cdots\}$, yaitu para polinomial dengan sifat

$$\int_{-1}^{1} P_i(x) P_j(x)\,dx = 0,\ i \neq j.$$

Beberapa polinomial Legendre awal adalah sebagai berikut

$$P_0(x) = 1,\ P_1(x) = x,\ P_2(x) = x^2 - \frac{1}{3},\ P_3(x) = x^3 - \frac{3}{5}x,\ P_4(x) = x^4 - \frac{6}{7}x^2 + \frac{3}{35}.$$

Akar-akar polinomial ini $t_1, \cdots, t_n$ selalu berbeda dan terbentang dalam interval $[-1, 1]$, dan ia diambil sebagai absis untuk formula integral Gauss. Sedangkan bobot $c_1, \cdots, c_n$ ditentukan berdasarkan formula

$$c_i = \int_{-1}^{1} \frac{(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}\,dx.$$

Untuk kebutuhan praktis, absis dan bobot tidak harus dihitung secara langsung, namun cukup menggunakan data yang sudah banyak tersedia dalam bentuk tabel integral Gauss misalnya oleh Stroud and Secrest yang dikutip dari [?], diberikan pada tabel 5.2.

Gambar 5.8: Interpretasi geometris metode trapesium dan Gauss

$k = 1, \cdots, N$ dapat dibentuk

$$\begin{aligned} G_k(f) &= \frac{x_k - x_{k-1}}{2} \sum_{j=1}^{n} c_{n,j} f\left(\frac{(x_k - x_{k-1})t_{n,j} + x_{k-1} + x_k}{2}\right) \\ &= \frac{h}{2} \sum_{j=1}^{n} c_{n,j} f\left(\frac{h t_{n,j} + x_{k-1} + x_{k-1} + h}{2}\right). \\ &= \frac{h}{2} \sum_{j=1}^{n} c_{n,j} f\left(\frac{h(t_{n,j} + 1) + 2x_{k-1}}{2}\right). \end{aligned}$$

Akhirnya, integral Gauss bersusun pada interval $[a, b]$ diperoleh dengan menjumlahkan semua $G_k(f)$.

$$\int_a^b f(x)\,dx \approx \frac{h}{2} \sum_{k=1}^{N} \sum_{j=1}^{n} c_{n,j} f\left(\frac{h(t_{n,j} + 1) + 2x_{k-1}}{2}\right). \tag{5.29}$$

Penulisan m-file untuk formula kuadratur bersusun

Interval $[a, b]$ dipartisi ke dalam n subinterval yang lebar masing-masingnya sama, yaitu

$$a =: x_0 < x_1 < x_2 < \cdots < x_n := b$$

dengan $h := x_i - x_{i-1}$, $i = 1, 2, \cdots, n$. Untuk metode midpoint dan metode Simpson disyaratkan banyak subinterval n genap, yaitu dengan formula

$$M(f) = 2h \sum_{k=1}^{\frac{n}{2}} f(x_{2k-1}),\ S(f) = \frac{h}{3}\left[f(a) + f(b) + 4 \sum_{k=1}^{\frac{n}{2}-1} f(x_{2k-1}) + 2 \sum_{k=1}^{\frac{n}{2}-2} f(x_{2k}) \right].$$

Sedangkan untuk metode Trapesium, banyaknya subinterval boleh genap atau ganjil, yaitu dengan fromula

$$T(f) = \frac{h}{2}\left[f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_{k-1}) \right].$$

Dikarenakan alasan ini maka disusun dua m-file, pertama untuk gabungan metode midpoint dan Simpson, sedangkan yang kedua untuk metode trapesium. Perlu diingat bahwa indeks pada MATLAB dimulai dari 1, jadi indeks 0 tidak dikenali. Jadi indeks yang berkaitan dengan absis digeser 1 langkah ke kanan, yaitu $a =: x_0 \to x(1)$ dan $x_n := b \to x(n+1)$. Untuk itu diperlukan adaptasi indeks formula pada penulisan m-file. Sebagai contoh, untuk absis berindeks ganjil $\{x_{2k-1} : k = 1, 2, \cdots\} = \{x_1, x_3, \cdots\}$ menjadi $\{x(2), x(3), \cdots\}$ pada m-file MATLAB. Begitu juga untuk absis dengan indeks genap. Karena banyak subinterval n diambil sebagai masukan maka $h := \frac{b-a}{n}$ perlu didefinisikan di awal kode. Kode berikut ditulis dengan memanfaatkan keunggulan operasi *array* sehingga tidak perlu kendali alur.

Metode midpoint dan Simpson bersusun

```
function [M,S]=mid_simp(a,b,n)
%masukkan n genap
h = (b-a)/n;
x = a:h:b;
if(rem(n,2)~=0)
```

```
>> [M,S]=mid_simp(0,2,15)
??? Error using ==> mid_simp
n yang dimasukkan tidak genap, coba cek ulang lagi!!!
```

Ini menunjukkan kode untuk metode trapesium masih bekerja dengan baik, tetapi untuk midpoint dan Simpson tidak jalan, tetapi muncul pesan *error* seperti terancang pada m-file. Hasil eksak integral ini dengan mudah dapat diperoleh sebagai berikut

$$\int_0^2 xe^{x^2}\,dx = \frac{1}{2}\int_0^2 d(e^{x^2}) = \left[\frac{1}{2}e^{x^2}\right]_0^2 = \frac{1}{2}(e^4-1) = 26.799075.$$

Aproksimasi dengan metode Simpson memberikan hasil paling akurat. ■

Metode integrasi Gauss

Ada 2 metode integrasi Gauss yang diberikan m-filenya, yaitu bentuk tunggal dan bentuk bersusun. Untuk bentuk tunggal, formulanya sebagai berikut

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\sum_{j=1}^{n} c_{n,j} f\left(\frac{(b-a)t_{n,j}+b+a}{2}\right).$$

Untuk ini dibutuhkan bobot dan absis seperti terdapat pada Tabel 5.2. Dalam kesempatan ini hanya disusun m-file yang besesuaian dengan $n = 4$, yaitu diperoleh

```
bobot = [0.8611363116, 0.3399810436, -0.3399810436, -0.8611363116]
absis = [0.3478548451, 0.6521451549, 0.6521451549, 0.3478548451].
```

Berikut m-file yang dimaksud dengan mengambil fungsi $f(x) = xe^{x^2}$ sebagai fungsi yang diintegralkan.

```
function q = gauss4(a,b)
%Pilih a dan b sebagai batas integral
c = [0.8611363116, 0.3399810436, -0.3399810436, -0.8611363116];%absis
t = [0.3478548451, 0.6521451549, 0.6521451549, 0.3478548451];%bobot
x = ((b-a)*t+b+a)/2; % absis hasil transformasi
q = (b-a)/2*sum(c.*feval('fun',x));
%fungsi yang diintegralkan
function y = fun(x);
y = x.*exp(x.^2);
```

5.4 Seputar Fungsi quad pada MATLAB

Sesungguhnya MATLAB telah menyediakan fasilitas berupa m-file untuk menghitung integral taktentu $\int_a^b f(x)\,dx$. Salah satunya adalah fungsi `quad`. Sintaksis penggunaan fungsi ini adalah sebagai berikut

```
q = quad('fun',a,b)
```

di mana `fun` adalah fungsi yang dintegralkan, `a` dan `b` adalah batas integrasi. Kode MATLAB ini disusun didasarkan pada metode Simpson adaptif rekursif dengan akurasi mencapai 10^{-6}, lihat [**?**]. Metode ini sebagai pengembangan metode Simpson. Karena beberapa keterbatasan, metode ini tidak dibahas secara detail karena cukup rumit dan panjang. Dapat diyakini bahwa hasilnya lebih akurat daripada metode Simpson biasa. Langkah-langkah penggunaan fasilitas ini adalah sebagai berikut.

- Tuliskan m-file untuk mendefinisikan fungsi yang akan diintegralkan, simpan dengan nama tertentu
- Melalui *command windows*, tulis perintah `q = quad('fun',a,b)` di mana `'fun'` nama fungsi pada langkah sebelumnya, `a` dan `b` adalah batas integral.

Contoh 5.12. Gunakan m-file **quad** pada MATLAB untuk mengaproksimasi integral berikut:

$$\int_0^2 xe^{x^2}\,dx \text{ dan } \int_0^2 \frac{1}{x^3-2x-5}\,dx.$$

Penyelesaian. Pertama-tama definisikan m-file berikut untuk mendefinisikan fungsi yang akan diintegralkan, yaitu

```
function y = fun21(x)
y = x.*exp(x.^2);
```

Kemudian, simpan m-file ini dengan nama **fun21.m**, biasanya nama file ini sudah ditawarkan langsung oleh MATLAB ketika Anda akan menyimpan. Jadi langsung klik **Save**. Dengan cara yang sama, definisikan m-file untuk fungsi kedua berikut

```
function y = fun22(x)
y = 1./(x.^3-2*x-5);
```

Setelah kedua m-file ini disimpan, buka *command window*. Lakukan perintah berikut

```
>> q = quad('fun21',0,2)
q =
```

b) Sesungguhnya data ini berasal dari fungsi $f(x) := x\sqrt{1+x^2}$. Hitunglah kesalahan nyata masing-masing aproksimasi di atas.

3. Gunakan metode midpoint, trapesium, dan Simpson bersusun untuk mengaproksimasi integral berikut, kemudian bandingkan akurasi masing-masing.

   a) $\int_0^2 e^{2x}\cos 3x\,dx$, $n = 4$

   b) $\int_0^{\pi/3} \tan x\,dx$, $n = 4$

   (c) $\int_1^4 \frac{x}{x^2+4}\,dx$, $n = 6$

   (d) $\int_0^{\pi} x^2\cos 2x\,dx$, $n = 8$

4. Tentukan banyak subinterval n dan ukuran langkah h yang dibutuhkan agar aproksimasi integral $\int_0^2 \frac{1}{x+2}\,dx$ kesalahannya tidak melebihi 10^{-5} untuk masing-masing metode berikut.

   a) metode midpoint bersusun

   b) metode trapesium bersusun

   c) metode Simpsoon bersusun

5. Sebuah mobil sedang berjalan pada lintasan lurus. Kecepatannya dicatat setiap 6 detik dalam waktu selama 84 detik. Kecepatan (v) dalam kilometer per jam diberikan pada tabel berikut.

t	0	6	12	18	24	30	36	42	48	54	60	66	72	78	84
v	134	145	159	168	158	143	130	118	107	92	84	96	112	125	133

   Hitunglah dengan cara aproksimasi jarak yang ditempuh oleh mobil dalam rentang waktu 84 detik tersebut. (Petunjuk: ingat derivatif fungsi jarak adalah fungsi kecepatan sehingga fungsi kecepatan dapat diperoleh dengan melakukan integral terhadap fungsi jarak).

6. Diberikan masalah menghitung integral $\int_0^2 e^{2x}\sin 3x\,dx$. Lakukan eksperimen numerik untuk melihat pengaruh penambahan subinterval pada formula kuadratur bersusun terhadap pola penurunan kesalahan. Terlebih dulu hitung nilai eksak integral ini. Gunakan kode MATLAB yang relevan dengan metode yang dimaksud. Rangkumlah hasil eksperimen numerik Anda ke dalam tabel seperti berikut.

n	4	6	8	10	12	14	16	18	20
E_M									
E_T									
E_S									

   Gambarkan grafik masing-masing kesalahan tersebut versus n. Berilah ulasan terhadap hasil ekperimen numerik ini.

c) Terapkan metode integral Gauss bersusun $n = 3$ dengan $N = 4$, kemudian bandingkan hasilnya dengan metode Simpson bersusun dengan $N = 4$. Metode mana yang lebih akurat. (Petunjuk: modifikasilah seperlunya m-file `gaus4_bersusun.m`).

Sekadar untuk mengetahui kesesuaian hasil aproksimasi yang diperoleh, ada baiknya konfirmasi dengan fungsi `quad` pada MATLAB.

6 SISTEM PERSAMAAN LINEAR

6.1 Pendahuluan

Banyak masalah pada bidang matematika terapan dan komputasi saintifik melahirkan sistem persamaan linear (SPL) yang harus diselesaikan. Bahkan dalam proses aproksimasi numerik itu sendiri masalahnya direduksi atau disederhanakan ke dalam bentuk SPL. Dalam notasi matriks-vektor, SPL mempunyai bentuk umum

$$Ax = b \tag{6.1}$$

di mana A adalah matriks koefisien berukuran $m \times n$, b vektor berukuran $m \times 1$. Vektor x berukuran $n \times 1$ merupakan variabel takdiketahui yang akan ditentukan. Pertanyaannya adalah apakah vektor b dapat disajikan sebagai kombinasi linear atas vektor-vektor kolom matriks A. Persamaan (6.1) dapat disajikan secara eksplisit sebagai

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

atau dalam bentuk kombinasi linear

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + x_3 \begin{bmatrix} a_{13} \\ a_{23} \\ \vdots \\ a_{m3} \end{bmatrix} + \cdots x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}. \tag{6.2}$$

Vektor x yang memenuhi keadaan ini disebut **penyelesaian** SPL (6.1). Jadi, komponen-komponen vektor penyelesaian merupakan koefisien kombinasi linear atas vektor kolom matriks A. Penyelesaian suatu SPL dapat ada atau tidak ada. Bila penyelesaiannya ada maka banyaknya dapat tunggal atau banyak. Matriks A secara umum tidak harus persegi,

Sedangkan untuk SPL kedua ditemukan $\det(A) = 0$ sehingga sistem ini singular. Jadi penyelesaiannya tergantung pada vektor b. Jika $b = [\ 1 \quad 3\ ]^T$ maka SPL ini tidak mempunyai penyelesaian. Fakta ini dapat dijelaskan secara sederhana sebagai berikut. Tulis SPL dalam bentuk biasa, yaitu

$$\begin{aligned} 2x_1 + 3x_2 &= 1 \\ 4x_1 + 6x_2 &= 3. \end{aligned}$$

Dengan menerapkan metode eliminasi yaitu persaamaan kedua dikurangi dengan 2 kali persamaan pertama, diperoleh pernyataan $0 = 1$, sebuah kontradiksi. Ini berarti tidak ada pasangan (x_1, x_2) yang memenuhi sistem ini. Sebaliknya, jika $b = [\ 1 \quad 2\ ]^T$ maka diperoleh

$$\begin{aligned} 2x_1 + 3x_2 &= 1 \\ 4x_1 + 6x_2 &= 2. \end{aligned}$$

Dengan prosedur eliminasi yang sama seperti sebelumnya diperoleh pernyataan $0 = 0$, sebuah kebenaran atau tautologi. Ini berarti salah satu variabel, katakan x_2 dipenuhi oleh sebarang bilangan. Bila dimisalkan $x_2 = \gamma$, γ konstanta sebarang maka para vektor $x = [\ \frac{1-3\gamma}{2} \quad \gamma\ ]^T$ adalah penyelesaian SPL yang dimaksud. Kebenarannya dapat diperiksa sebagai berikut

$$\begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \begin{bmatrix} \frac{1-3\gamma}{2} \\ \gamma \end{bmatrix} = \begin{bmatrix} 1 - 3\gamma + 3\gamma \\ 2 - 6\gamma + 6\gamma \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Jadi sistem singular ini mempunyai takberhingga banyak penyelesaian. ■

Untuk menyelesaikan SPL umumnya digunakan strategi transformasi menjadi SPL bentuk lain ekuivalen tetapi lebih mudah dihitung. Dengan menggandakan kedua ruas SPL $Ax = b$ dengan matriks taksingular M maka diperoleh SPL baru yang berbentuk $MAx = Mb$. Penyelesaian SPL ini diperoleh sebagai

$$x = (MA)^{-1}Mb = A^{-1}M^{-1}Mb = A^{-1}b,$$

yaitu sama dengan penyelesaian SPL sebelumnya. Kedua SPL $Ax = b$ dan $A'x = b'$ di mana $A' = MA$ dan $b' = Mb$ ternyata ekuivalen.

Transformasi matriks permutasi

Penyelesaian sebuah SPL tidak berubah jika posisi barisnya ditukar satu sama lainnya. Penukaran baris SPL ini sama artinya dengan menukar baris matriks A berikut komponen

Contoh 6.2. Diberikan SPL $Ax = b$ berikut

$$\begin{bmatrix} 2 & -1 & 1 \\ 3 & 2 & -2 \\ 1 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ -5 \\ 14 \end{bmatrix}.$$

Sistem ini mempunyai penyelesaian $x = \begin{bmatrix} 1 & 2 & 5 \end{bmatrix}^T$. Misalkan diambil matriks permutasi

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Bila dibandingkan dengan kolom-kolom matriks identitas, penggandaan dari kanan terhadap matriks A akan mengakibatkan pertukaran kolom $k_1 \to k_2$, $k_2 \to k_3$, $k_3 \to k_1$ di mana notasi k_p menyatakan kolom ke p. Jelasnya

$$A_1 = AP = \begin{bmatrix} 1 & 2 & -1 \\ -2 & 3 & 2 \\ 3 & 1 & -1 \end{bmatrix}.$$

Persamaan $A_1 x = b$ memberikan penyelesaian $x = \begin{bmatrix} 5 & 1 & 2 \end{bmatrix}^T$, ternyata hanya menukar posisi penyelesaian sebelumnya, yaitu $x_1 \to x_2$, $x_2 \to x_3, x_3 \to x_1$. Bila digandakan dari kiri maka akan terjadi pertukaran baris, yaitu

$$A_2 = PA = \begin{bmatrix} 3 & 2 & -2 \\ 1 & -1 & 3 \\ 2 & -1 & 1 \end{bmatrix}.$$

Persamaan $A_2 x = b$ memberikan penyelesaian $x = \begin{bmatrix} 4.71 & -10.71 & -6.14 \end{bmatrix}^T$ suatu hasil yang berbeda jauh dari penyelesaian sebelumnya. Jadi transformasi matriks permutasi dilakukan dengan penggandaan kanan. Sama halnya dengan transformasi matriks diagonal. Jika $A_1 = AD$, D sebuah matriks diagonal maka penyelesaian $A_1 y = b$ adalah $y = D^{-1}x$, yaitu penskalaan masing-masing komponen penyelesaian semula dengan elemen matriks D^{-1}. Bila

$$D = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

maka sistem $ADx = b$ memberikan penyelesaian $x = \begin{bmatrix} 2 & 6 & \frac{5}{2} \end{bmatrix}^T$.

Selanjutnya kita bahas beberapa metode umum penyelesaian SPL dan masalah penting yang terkait.

Untuk substitusi maju diperoleh iterasi berikut:

$$\begin{aligned} x_1 &= \frac{b_1}{a_{11}} \\ x_k &= \tfrac{1}{a_{kk}} \left(b_k - \sum_{j=1}^{k-1} a_{kj} x_j \right), \; k = 2, 3, \cdots, n. \end{aligned}$$

Contoh 6.3. Diperhatikan SPL segitiga atas berikut

$$\begin{bmatrix} 2 & 4 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \\ 8 \end{bmatrix}.$$

Persamaan terakhir SPL ini menghasilkan $2x_3 = 8$ diperoleh $x_3 = 4$. Substitusi x_3 ke persamaan kedua $2x_2 + x_3 = 10$ diperoleh $x_2 = 3$. Terakhir, x_2 dan x_3 yang sudah diperoleh disubstitusikan ke dalam persamaan pertama $2x_1 + 4x_2 + x_3 = 4$ untuk mendapatkan $x_1 = -6$. ■

6.3 Matriks Eliminasi Elementer

Permasalahan menentukan penyelesaian SPL menjadi sederhana jika kita dapat mentransformasinya ke dalam bentuk segitiga. Kita membutuhkan transformasi yang dapat menjadikan elemen taknol tertentu matriks koefisien A menjadi nol. Perhatikan ilustrasi berikut. Bila $a = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T$ sebuah vektor dengan $a_1 \neq 0$ maka

$$\begin{bmatrix} 1 & 0 \\ -\frac{a_2}{a_1} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ 0 \end{bmatrix}.$$

Jadi matriks $M = \begin{bmatrix} 1 & 0 \\ -\frac{a_2}{a_1} & 1 \end{bmatrix}$ menjadikan nol komponen a_2 pada vektor $a = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T$.
Secara umum jika $a = \begin{bmatrix} a_1 & \cdots & a_k & a_{k+1} & \cdots & a_n \end{bmatrix}^T$ dengan $a_k \neq 0$ dan matriks

$$M_k = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_n & 0 & \cdots & 1 \end{bmatrix} \tag{6.3}$$

Diperoleh

$$M_1 A = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix}.$$

Matriks M_2 didefinisikan melalui vektor kolom kedua $[\ 4 \quad 9 \quad -3\ ]^T$, yaitu dengan memandang $a_1 = 4$, $a_2 = 9$ dan $a_3 = -3$. Karena pivotnya $a_1 = 4$ maka diperoleh matriks eliminasi elementer berikut

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a_2}{a_2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}.$$

Jadi, dengan transformasi $T := M_1 M_2$ maka matriks A menjadi matriks segitiga atas, yakni

$$TA = M_1 M_2 A = \begin{bmatrix} 2 & 4 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 5 \end{bmatrix}. \quad \blacksquare$$

Pendefinisian matriks eliminasi elementer ini membutuhkan pembiasaan khususnya dalam menetapkan elemen selain nol dan satu. Tekniknya, mulai dari matriks identitas kemudian modifikasi elemen dibawah diagonal utama.

Permasalahan krusial dalam membangun matriks eliminasi elementer adalah memilih pivot yang tidak nol. Sebagai ilustrasi diperhatikan matriks berikut.

$$A = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 3 & 0 & 5 & 1 \\ 2 & 3 & 1 & 0 \\ 4 & 1 & 2 & 7 \end{bmatrix}.$$

Dengan mengikuti prosedur sebelumnya maka M_1 tidak dapat didefinisikan dikarenakan $a_{11} = 0$. Jadi, elemen ini tidak dapat dijadikan pivot. Dalam kasus seperti ini perlu menyusun kembali baris atau kolom matriks A sehingga menjadi matriks baru di mana semua elemen pada diagonal utamanya tidak nol. Strateginya adalah menggandakan dengan matriks permutasi yang bersesuaian. Melalui pengamatan langsung, kolom yang harus ditukar dengan kolom lainnya adalah kolom pertama (k_1) dan kolom kedua (k_2). Penukaran $k_1 \to k_3$ dan $k_3 \to k_1$ tidak dapat dilakukan karena tidak merubah keadaan. Jadi kemungkinannya adalah $k_1 \to k_2$ dan $k_2 \to k_1$. Untuk perubahan ini digunakan matriks

Contoh 6.5. Gunakan metode eliminasi Gaussian untuk menyelesaikan sistem persamaan berikut

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 2, \\ 4x_1 + 9x_2 - 3x_3 &= 8, \\ -2x_1 - 3x_2 + 7x_3 &= 10. \end{aligned}$$

Penyelesaian. Matriks koefisien dan vektor pada persamaan ini dapat ditulis sebagai

$$A = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix}, \; b = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}.$$

Seperti telah diperoleh pada contoh sebelumnya

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \text{ dan } M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

sehingga

$$M_2M_1A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix}$$

dan

$$M_2M_1b = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}.$$

Akhirnya diperoleh sistem persamaan dalam bentuk segitiga atas

$$\begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}.$$

Dengan substitusi mundur diperoleh $4x_3 = 8 \rightarrow x_3 = 2$, $x_2 + x_3 = 4 \rightarrow x_2 = 4 - 2 = 2$ dan $2x_1 + 4x_2 - 2x_3 = 2 \rightarrow x_1 = \frac{1}{2}(2 + 2(2) - 4(2)) = -1$. Jadi penyelesaiannya adalah $x = [\; -1 \quad 2 \quad 2 \;]^T$. ■

dapat diselesaikan dengan mudah, yaitu

$$\begin{aligned} x_4 &= 1, \\ x_3 &= \frac{1}{3}(13 - 13x_4) = \frac{1}{3}(13 - 13) = 0, \\ x_2 &= -(-7 + 5x_4 + x_3) = -(-7 + 5 + 0) = 2, \\ x_1 &= 4 - 3x_4 - x_2 = 4 - 3 - 2 = -1. \end{aligned}$$

Jadi penyelesaiannya adalah $x_1 = -1$, $x_2 = 2$, $x_3 = 0$. ■

6.5 Strategi Pivoting

Dalam proses pembentukan sistem segitiga atas, biasanya digunakan salah satu elemen pada suatu baris lebih atas untuk mengeliminasi elemen pada subdiagonal, yaitu elemen-elemen di bawah diagonal utama. Misalkan A matriks yang akan ditransformasikan menjadi matriks segitiga atas.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

Elemen a_{11} digunakan sebagai pembagi untuk mendefinisikan pengganda $\lambda_1 := \frac{a_{21}}{a_{11}}$ dan $\lambda_2 := \frac{a_{31}}{a_{11}}$ sehingga operasi $(E_2 - \lambda_1 E_1) \rightarrow (E_2)$ dan $(E_3 - \lambda_2 E_3) \rightarrow (E_3)$ mengeliminasi a_{21} dan a_{31}. Terbentuklah matriks hasil transformasi berupa

$$A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a^1_{22} & a^1_{23} \\ 0 & a^1_{32} & a^1_{33} \end{bmatrix}.$$

Sekarang elemen a^1_{22} digunakan sebagai pembagi untuk mendefinisikan pengganda $\lambda := \frac{a^1_{22}}{a^1_{22}}$ sehingga operasi $(E_3 - \lambda E_3) \rightarrow (E_3)$ mengeliminasi elemen a^1_{32}. Akhirnya diperoleh matriks segitiga atas seperti yang diinginkan.

$$A^{(2)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a^1_{22} & a^1_{23} \\ 0 & 0 & a^2_{33} \end{bmatrix}.$$

Elemen yang digunakan sebagai pembagi untuk pengganda seperti a_{11} dan a^1_{22} di atas sebagai **pivot,** tentunya diasumsikan tidak nol.

Sistem persamaan ini telah dirancang sedemikian rupa sehingga penyelesaiannya adalah $x_1 = 10$ dan $x_2 = 1$. Terapkan metode eliminasi Gaussian untuk menyelesaikan persamaan ini.

Penyelesaian. Pertama eliminasi x_1 pada E_2 dengan menggunakan baris pertama E_1. Diperoleh pengganda $\lambda = \frac{5.291}{0.003000} = 1763.\overline{6}$. Dengan OBE $(E_2 - \lambda E_1) \to (E_2)$ maka diperoleh bentuk baru

$$\begin{array}{lrcrcl} E_1: & 0.003000x_1 & + & 59.14x_2 & = & 59.17, \\ E_2: & & - & 104300x_2 & \approx & -104400. \end{array}$$

Setelah diselesaikan dengan substitusi mundur diperoleh

$$x_2 \approx \frac{-104400}{-104300} = 1.001.$$

Hasil ini cukup dekat kepada penyelesaian eksaknya, yaitu $x_2 = 1$. Tetapi kecilnya elemen pivot $a_{11} = 0.003000$ berdampak pada akurasi x_1 dikarenakan kesalahan pembulatan, yaitu

$$x_1 \approx \frac{59.17 - (59.14)(1.001)}{0.003000} = -10.00,$$

padahal eksaknya adalah $x_1 = 10$. Sekarang, sebelum dilakukan eliminasi kita lakukan operasi pertukaran baris $(E_1) \leftrightarrow (E_2)$ sehingga diperoleh sistem baru yang ekuivalen

$$\begin{array}{lrcrcl} E_1: & 5.291x_1 & - & 6.130x_2 & = & 46.78, \\ E_2: & 0.003000x_1 & + & 59.14x_2 & = & 59.17. \end{array}$$

Untuk sistem ini diperoleh $\lambda = \frac{0.00300}{5.291} = 0.0005670$. Dengan operasi $(E_2 - \lambda E_1) \to (E_2)$ diperoleh

$$\begin{array}{lrcrcl} E_1: & 5.291x_1 & - & 6.130x_2 & = & 46.78, \\ E_2: & & & 59.14x_2 & = & 59.14. \end{array}$$

Dengan substitusi mundur diperoleh penyelesaian dalam 4 digit signifikan, yaitu $x_1 = 10.00$ dan $x_2 = 1.000$ sesuai dengan hasil yang diharapkan. ■

Prinsipnya semua elemen tidak nol dapat diambil sebagai pivot, tetapi secara teknis penukaran baris ini dilakukan agar magnitud pengganda λ tidak melebihi 1. Agar keadaan ini dipenuhi maka pivot diambil sebagai elemen dengan magnitud terbesar terletak pada atau dibawah diagonal. Teknik penukaran baris seperti ini disebut **strategi pivoting parsial**. Dikatakan parsial karena hanya kolom tertentu saja yang ditentukan elemen pivotnya, tidak melibatkan elemen pada kolom lain.

Karena $\frac{|a_{31}|}{s_3}$ paling besar maka dilakukan pertukaran baris $(E_1) \leftrightarrow (E_3)$. Sistem baru yang diperoleh dalam bentuk matriks perluasan berikut.

$$\left[\begin{array}{cccc:c} 1.09 & 0.987 & 0.832 & \vdots & 4.21 \\ 4.01 & 10.2 & -1.12 & \vdots & -3.09 \\ 2.11 & -4.21 & 0.921 & \vdots & 4.21 \end{array}\right]$$

Diperoleh pengganda $\lambda_1 = \frac{4.01}{1.09} = 3.68$ dan $\lambda_2 = \frac{2.11}{1.09} = 1.94$. Dengan melakukan operasi $(E_2 - \lambda_1 E_1) \rightarrow (E_2)$ dan $(E_3 - \lambda_2 E_1) \rightarrow (E_3)$ maka diperoleh

$$\left[\begin{array}{cccc:c} 1.09 & 0.987 & 0.832 & \vdots & 4.21 \\ 0 & 6.57 & -4.18 & \vdots & -18.6 \\ 0 & -6.12 & -0.689 & \vdots & -6.16 \end{array}\right].$$

Untuk mengeliminasi x_2, perhatikan perbandingan

$$\frac{|a_{22}|}{s_2} = \frac{6.57}{10.2} = 0.644 \text{ dan } \frac{|a_{32}|}{s_3} = \frac{6.12}{4.21} = 1.45.$$

Karena $\frac{|a_{32}|}{s_3} > \frac{|a_{22}|}{s_2}$ maka dilakukan penukaran baris $(E_2) \leftrightarrow (E_3)$, sehingga diperoleh

$$\left[\begin{array}{cccc:c} 1.09 & 0.987 & 0.832 & \vdots & 4.21 \\ 0 & -6.12 & -0.689 & \vdots & -6.16 \\ 0 & 6.57 & -4.18 & \vdots & -18.6 \end{array}\right].$$

Pengganda terakhir diperoleh $\lambda = \frac{6.57}{-6.12} = -1.07$. Dengan melakukan operasi $(E_3 - \lambda E_2) \rightarrow (E_3)$ diperoleh bentuk segitiga atas berikut

$$\left[\begin{array}{cccc:c} 1.09 & 0.987 & 0.832 & \vdots & 4.21 \\ 0 & -6.12 & -0.689 & \vdots & -6.16 \\ 0 & 0 & -4.92 & \vdots & -25.2 \end{array}\right].$$

Sesungguhnya, berdasarkan hasil kalkulasi elemen $a_{32} = 0.02$ tidak persis 0. Hal ini dikarenakan adanya kesalahan pembulatan. Karena kita tahu bahwa nilai sesungguhnya adalah 0 maka kita tetapkan $a_{32} = 0$. Akhirnya, dengan substitusi mundur diperoleh

$$x_1 = -0.431,\ x_2 = 0.430,\ x_3 = 5.12. \quad \blacksquare$$

sehingga diperoleh

$$P_2M_1P_1A = \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \text{ dan } P_2M_1P_1b = \begin{bmatrix} 8 \\ 14 \\ -12 \end{bmatrix}.$$

Akhirnya elemen subdiagonal pada kolom kedua dapat dinolkan dengan menggunakan matriks eliminasi

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & 1 \end{bmatrix}.$$

Pengganda $\frac{1}{3}$ diperoleh dari $-\left(\frac{-1/2}{3/2}\right) = \frac{1}{3}$. Akhirnya diperoleh bentuk segitiga

$$M_2P_2M_1P_1A = \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & 0 & \frac{4}{3} \end{bmatrix} \text{ dan } P_2M_1P_1b = \begin{bmatrix} 8 \\ 14 \\ \frac{8}{3} \end{bmatrix}.$$

Bentuk terakhir ini diselesaikan dengan substitusi mundur untuk mendapatkan penyelesaian $x_3 = 2$, $x_2 = 2$ dan $x_1 = -1$. ■

6.5.3 Pivoting Total

Bentuk lain strategi pivoting adalah **pivoting total**. Pada tahap ke-i, strategi ini mempertimbangkan semua elemen a_{kj}, $k, j = i, i+1, \cdots, n$ untuk mendapatkan elemen dengan magnitud terbesar dan meletakkannya pada posisi pivot. Teknik ini menuntut adanya pertukaran antarbaris, dan juga pertukaran antar kolom. Penggandaan matriks permutasi dilakukan dari kiri dan kanan matriks A. Penggandaan dari kiri akan menukar baris dan penggandaan dari kanan akan melakukan pertukaran kolom. Karena pertukaran kolom akan merubah penyelesaian sistem persamaan maka diperlukan teknik untuk mendapatkan kembali penyelesaian semula. Bentuk semula sistem persamaan linear adalah $Ax = b$. Melalui pertukaran baris dan kolom dengan matriks permutasi dan matriks eliminasi elementer pada akhirnya diperoleh bentuk

$$PAQ = LU$$

di mana P matriks permutasi yang mempertukarkan baris dan Q permutasi yang mempertukarkan kolom. Misalkan y penyelesaian SPL segitiga bawah $Ly = Pb$ dan z penyelesaian

Untuk tahap kedua, pertukarkan baris kedua dan baris ketiga karena $|3| > |-2|$. Diperoleh

$$\left[\begin{array}{ccc:c} -4 & 2 & -6 & 14 \\ 0 & 3 & 1 & 15 \\ 0 & -2 & -1 & 12 \end{array}\right].$$

Selanjutnya 3 diambil sebagai pivot, diperoleh pengganda $\lambda = -\frac{2}{3}$. Dengan operasi $(E_3 - \lambda E_2) \rightarrow (E_3)$ diperoleh

$$\left[\begin{array}{ccc:c} -4 & 2 & -6 & 14 \\ 0 & 3 & 1 & 15 \\ 0 & 0 & -\frac{1}{3} & 22 \end{array}\right].$$

Akhirnya dengan substitusi mundur diperoleh $-\frac{1}{3}x_3 = 22 \rightarrow x_3 = -66$, $3x_2 = 15 - x_3 \rightarrow x_2 = 27$ dan $-4x_1 = 14 + 6x_3 - 2x_2 \rightarrow x_1 = 109$.
2) Untuk skala ditentukan dulu faktor skala sebagai elemen dengan magnitud terbesar pada masing-masing baris, yaitu $s_1 = 3$, $s_2 = 6$, $s_3 = 4$. Jadi

$$\frac{|a_{11}|}{s_1} = \frac{2}{3}, \frac{|a_{21}|}{s_2} = \frac{4}{6} = \frac{2}{3}, \frac{|a_{31}|}{s_3} = \frac{2}{4} = \frac{1}{2}.$$

Karena ada dua nilai perbandingan yang sama maka ada dua kemungkinan cara yang dapat dilakukan, yaitu tidak mengadakan pertukaran baris sama sekali atau melakukan pertukaran antara baris kedua dan ketiga. Kita pilih yang pertama agar ada perbedaan dengan hasil yang telah diperoleh pada strategi pivoting parsial sebelumnya, yaitu tetap menulis

$$\left[\begin{array}{ccc:c} 2 & -3 & 2 & 5 \\ -4 & 2 & -6 & 14 \\ 2 & 2 & 4 & 8 \end{array}\right].$$

Dengan proses yang sama seperti sebelumnya diperoleh

$$\left[\begin{array}{ccc:c} -4 & 2 & -6 & 14 \\ 0 & 3 & 1 & 15 \\ 0 & -2 & -1 & 12 \end{array}\right].$$

Akhirnya gunakan matriks eliminasi elementer M_2 berikut

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{7}{10} & 1 \end{bmatrix}}_{M_2} \begin{bmatrix} -6 & 2 & -4 \\ 0 & \frac{10}{3} & -\frac{2}{3} \\ 0 & -\frac{7}{3} & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} -6 & 2 & -4 \\ 0 & \frac{10}{3} & -\frac{2}{3} \\ 0 & 0 & \frac{1}{5} \end{bmatrix} =: A_3.$$

Dengan menulis $A_3 := U$ maka diperoleh bentuk

$$M_2 \underbrace{P_2 M_1 P_1}_{P} A \underbrace{Q_1}_{Q} = U \rightarrow PAQ = \underbrace{M_2^{-1}}_{L} U \rightarrow PAQ = LU.$$

di mana

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{7}{10} & 1 \end{bmatrix},\ P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{2}{3} & 1 \\ 1 & \frac{1}{3} & 0 \end{bmatrix},\ Q = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, U = \begin{bmatrix} -6 & 2 & -4 \\ 0 & \frac{10}{3} & -\frac{2}{3} \\ 0 & 0 & \frac{1}{5} \end{bmatrix}.$$

Dengan substitusi mundur diperoleh penyelesaian sistem segitiga bawah $Ly = Pb$, yaitu

$$y = \begin{bmatrix} 14 \\ 17\frac{1}{3} \\ 9\frac{2}{3} \end{bmatrix}.$$

Dengan substitusi maju diperoleh penyelesaian sistem segitiga atas $Uz = y$, yaitu

$$z = \begin{bmatrix} -66 \\ 27 \\ 109 \end{bmatrix}.$$

Akhirnya penyelesaian sistem semula $Ax = b$ diperoleh sebagai

$$x = Qz = \begin{bmatrix} 109 \\ 27 \\ -66 \end{bmatrix}. \quad \blacksquare$$

Berdasarkan contoh ini ternyata strategi pivoting total sangat rumit. Walaupun secara numerik strategi ini sangat stabil namun ia membutuhkan lebih banyak tahap dalam memilih pivot daripada strategi pivoting parsial. Oleh karena itu, strategi pivoting parsial sudah mencukupi untuk diterapkan dalam metode eliminasi Gaussian.

Penyelesaian. Pertama definisikan matriks koefisien A dan vektor b pada ruas kanan, sebagai berikut

```
>>A=[2.11 -4.21 0.921;4.01 10.2 -1.12; 1.09 0.987 0.832];
>>b=[2.01;-3.09;4.21];
```

Penyelesaian diperoleh dengan cara sangat sederhana, yaitu cukup dengan perintah

```
>> x=A\b
x =

    -0.4280
    0.4269
    5.1144
```

Salah satu cara untuk mengetahui kualitas penyelesaian yang diperoleh kita gunakan residu, yaitu $r := Ax - b$. Semakin residu mendekati vektor nol maka semakin baik penyelesaian x. Untuk hasil keluaran MATLAB diperoleh residu sebagai berikut.

```
>>r = Ax-b
r =

    1.0e-014 *
    -0.1776
          0
    -0.0888
```

Ini berarti residu yang diperoleh adalah $r = \begin{bmatrix} -0.1776 \\ 0 \\ -0.0888 \end{bmatrix} \times 10^{-14}$, suatu hasil yang sangat kecil. Selanjutnya residu yang diberikan oleh penyelesaian sebelumnya adalah sebagai berikut.

```
>> x1=[-0.431;0.430;5.12];
>> r1=A*x1-b
r1 =
   -0.0142
    0.0133
    0.0045
```

Jadi residu untuk penyelesaian ini adalah $r_1 = \begin{bmatrix} -0.0142 \\ 0.0133 \\ 0.0045 \end{bmatrix}$, masih cukup besar jika dibandingkan dengan residu penyelesaian yang dihasilkan oleh MATLAB. ■

perkalian/pembagian. Terbentuklah sistem baru sebagai berikut

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^1 & a_{23}^1 \\ 0 & a_{32}^1 & a_{33}^1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^1 \\ b_3^1 \end{bmatrix}.$$

2. Selanjutnya eliminasi x_2 pada baris E_3 menggunakan E_2, yaitu dengan melakukan operasi $(E_3 - \frac{a_{32}^1}{a_{22}^1}E_2) \rightarrow (E_3)$, yaitu

$$a_{32}^1 - \frac{a_{32}^1}{a_{22}^1}a_{22}^1,\ a_{33}^1 - \frac{a_{32}^1}{a_{22}^1}a_{23}^1,\ b_3^1 - \frac{a_{32}^1}{a_{22}^1}b_2^1$$

sehingga dibutuhkan 3 operasi pengurangan dan 4 operasi perkalian/pembagian.

3. Diperoleh bentuk segitiga

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^1 & a_{23}^1 \\ 0 & 0 & a_{33}^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^1 \\ b_3^2 \end{bmatrix}.$$

Selanjutnya substitusi mundur dilakukan sebagai berikut

$$x_3 = \frac{b_3^2}{a_{33}^2},\ x_2 = \frac{1}{a_{22}^1}(b_2 - a_{23}^1 x_3) = \frac{b_2}{a_{22}^1} - \frac{a_{23}^1}{a_{22}^1}x_3$$

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3) = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3.$$

Dibutuhkan 9 operasi perkalian/pembagian dan 3 operasi pengurangan/penjumlahan. Jadi total operasi aritmatika yang dibutuhkan adalah $10 + 4 + 9 = 23$ untuk operasi perkalian/pembagian dan $8 + 3 + 3 = 14$ untuk operasi penjumlahan/pengurangan.

Terdapat sedikit perbedaan banyaknya operasi aritmatika dengan hasil yang diberikan oleh Faires and Burden (2003) dalam [**?**] seperti terdapat pada tabel berikut.

TABEL 6.1: Kompleksitas metode eliminasi Gaussian

n	Perkalian/pembagian	Penjumlahan/pengurangan
3	17	11
10	430	375
50	44150	42875
100	343300	338250

```
3.00000000000093
3.99999999999938
```

Berdasarkan ilustrasi ini maka jelas bahwa penggunaan operator *backslash* lebih teliti daripada penggunaan invers matriks.

Metode lain yang sering digunakan adalah metode Cramer, di mana setiap komponen penyelesaian dihitung sebagai rasio dua determinan. Ternyata metode Cramer mempunyai kompleksitasnya yang sangat tinggi (Heath, 1997). Metode lain yang cukup populer adalah metode eliminasi Gauss-Jordan. Mirip eliminasi Gaussian standar, tetapi di sini matriks akhir berupa matriks diagonal. Menurut Heath (1997), biaya komputasi metode Gauss-Jordan ini 50% lebih mahal daripada metode eliminasi Gaussian standar. Derivasi formula kompleksitas untuk metode eliminasi Gaussian standar dapat dilihat pada (Kress, 1998).

Soal-soal untuk Latihan

1. Diberikan matriks A sebagai berikut

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 3 & 2 \end{bmatrix}.$$

   Bila $b = [\ 2 \quad 4 \quad 6\ ]^T$, berapa banyak penyelesaian yang dimiliki oleh $Ax = b$.

2. Selesaikan sistem persamaan linear dengan menggunakan metode eliminasi Gaussian

$$\begin{aligned} x_1 + x_2 - x_3 &= 1 \\ x_1 + x_2 + 4x_3 &= 2 \\ 2x_1 - x_2 + 2x_3 &= 3 \end{aligned}$$

   dengan cara biasa dan dengan ketiga strategi pivoting: parsial, skala, dan total.

3. Sistem persamaan linear berikut sudah dirancang sedemikian rupa sehingga penyelesaian eksaknya adalah $x_1 = 1$, $x_2 = 0.5$, $x_3 = -1$.

$$\begin{aligned} 3.3330x_1 + 15920x_2 + 10.333x_3 &= 7953 \\ 2.2220x_1 + 16.710x_2 + 9.6120x_3 &= 0.965 \\ -1.5611x_1 + 5.1792x_2 - 1.6855x_3 &= 2.714 \end{aligned}$$

   Selesaikan persamaan linear ini dengan metode eliminasi Gaussian standar, dengan pivoting parsial, pivoting skala, dan pivoting total. Periksalah kualitas penyelesaian

DAFTAR PUSTAKA

Indeks

Indeks