

# Bab 1

## Metode Numerik Secara Umum

---

Pengetahuan dimulai dari rasa ingin tahu, kepastian dimulai dengan rasa ragu-ragu, dan filsafat dimulai dengan kedua-duanya.  
(Jujun S. Suriasumantri)

Rasa ingin tahu adalah ibu dari semua ilmu pengetahuan  
(Anonim)

Persoalan yang melibatkan model matematika banyak muncul dalam berbagai disiplin ilmu pengetahuan, seperti dalam bidang fisika, kimia, ekonomi, atau pada persoalan rekayasa (*engineering*), seperti Teknik Sipil, Teknik Mesin, Elektro, dan sebagainya. Seringkali model matematika tersebut muncul dalam bentuk yang tidak ideal alias rumit. Model matematika yang rumit ini adakalanya tidak dapat diselesaikan dengan metode analitik yang sudah umum untuk mendapatkan **solusi sejatinya** (*exact solution*). Yang dimaksud dengan metode analitik adalah metode penyelesaian model matematika dengan rumus-rumus aljabar yang sudah baku (lazim).

Sebagai contoh ilustrasi, tinjau sekumpulan persoalan matematik di bawah ini. Bagaimana cara anda menyelesaikannya?

- (i) Tentukan akar-akar persamaan polinom:

$$23.4x^7 - 1.25x^6 + 120x^4 + 15x^3 - 120x^2 - x + 100 = 0$$

- (ii) Tentukan harga  $x$  yang memenuhi persamaan:

$$\sqrt{27.8e^{5x} - \frac{1}{x}} = \cos^{-1} \frac{(120x^2 + \sqrt{2x})}{17x - 65}$$

- (iii) Selesaikan sistem persamaan linier (*linear*):

$$\begin{aligned} 1.2a - 3b - 12c + 12d + 4.8e - 5.5f + 100g &= 18 \\ 0.9a + 3b - c + 16d + 8e - 5f - 10g &= 17 \\ 4.6a + 3b - 6c - 2d + 4e + 6.5f - 13g &= 19 \\ 3.7a - 3b + 8c - 7d + 14e + 8.4f + 16g &= 6 \\ 2.2a + 3b + 17c + 6d + 12e - 7.5f + 18g &= 9 \\ 5.9a + 3b + 11c + 9d - 5e - 25f - 10g &= 0 \\ 1.6a + 3b + 1.8c + 12d - 7e + 2.5f + g &= -5 \end{aligned}$$

- (iv) Tentukan nilai maksimum fungsi tiga matra (*dimension*):

$$f(x,y) = \cos \frac{x - \sqrt{\sin(x)} + 3}{4 + (xy)^2} + \sin(3xy - 1) - \tan\left(\frac{x(0.08 + \cos(x))}{y}\right)$$

- (v) Bila diperoleh tabulasi titik-titik (x,y) sebagai berikut (yang dalam hal ini rumus fungsi  $y = f(x)$  tidak diketahui secara eksplisit):

$x$	$y = f(x)$
2.5	1.4256
3.0	1.7652
3.5	2.0005
4.4	2.8976
6.8	3.8765

Hitung taksiran nilai  $y$  untuk  $x = 3.8$ !

- (vi) Berdasarkan titik-titik data pada tabel persoalan (v) di atas, berapa nilai  $f'(3.5)$  dan nilai  $f''(3.5)$  ?

- (vii) Hitung nilai integral-tentu berikut:

$$\int_{1.2}^{2.5} \left( \sqrt{\left(45.3e^{7x} + \frac{100}{x}\right)^4 + \frac{4}{(x^2 + 1)}} \right) dx$$

- (viii) Diberikan persamaan differensial biasa (PDB) dengan nilai awal:

$$150y'' + 2y't = \frac{\sqrt{\ln(21t + 40)}y}{t^2} + 120 \quad ; y'(0) = 0, y(0) = 1.2$$

Hitung nilai  $y$  pada  $t = 1.8!$

Menghadapi soal-soal seperti di atas, anda mungkin menyerah, atau mungkin mengatakan bahwa soal-soal tersebut tidak dapat diselesaikan dengan metode analitik yang biasa anda kenal. Soal (i) misalnya, biasanya untuk polinom derajat 2 orang masih dapat mencari akar-akar polinom dengan rumus *abc* yang terkenal itu yaitu

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (\text{P.1.1})$$

namun, untuk polinom derajat  $> 2$ , seperti pada soal (i), tidak terdapat rumus aljabar untuk menghitung akar polinom. Yang mungkin kita lakukan adalah dengan memanipulasi polinom, misalnya dengan memfaktorkan (atau menguraikan) polinom tersebut menjadi perkalian beberapa suku. Semakin tinggi derajat polinom, jelas semakin sukar memfaktorkannya. Ada juga beberapa alternatif lain. Yang pertama dengan cara coba-coba seperti **metode pembagian sintetis Horner**. Dengan metode ini, polinom dibagi dengan sebuah bilangan. Jika sisa pembagiannya nol, maka bilangan tersebut adalah akar polinom. Cara kedua adalah secara grafik, yaitu dengan merajah kurva fungsi di atas kertas grafik, kemudian berdasarkan gambar kurva, kita mengambil tarikan akar secara kasar, yaitu titik poyong kurva dengan sumbu- $x$ . Cara ini, selain kaku dan tidak praktis, ketelitian akar yang diperoleh sangat bergantung pada ketelitian penggambaran kurva.. Lagipula, merajah kurva pada kertas grafik hanya terbatas pada fungsi yang dapat digambarkan pada bidang dua matra atau tiga matra. Untuk fungsi dengan peubah lebih besar dari 3 jelas tidak dapat (malah tidak mungkin) kita gambar kurvanya. Soal nomor (ii) masih sejenis dengan soal (i), yaitu menentukan nilai  $x$  yang memenuhi kedua persamaan.

Untuk soal nomor (iii), juga tidak ada rumus yang baku untuk menemukan solusi sistem persamaan linier. Apabila sistem persamaannya hanya berupa dua garis lurus dengan dua peubah, kita masih dapat menemukan solusinya (dalam hal ini titik potong kedua garis) dengan menggunakan rumus titik potong dua buah garis atau dengan aturan Cramer. Kita juga dapat menemukan titik potong tersebut dengan menggambar kedua garis pada kertas grafik. Untuk sistem yang terdiri dari tiga buah persamaan linier dengan tiga peubah, aturan Cramer masih dapat digunakan untuk memecahkan sistem. Tetapi untuk sistem dengan jumlah persamaan dan jumlah peubah lebih besar dari tiga, tidak ada rumus yang dapat dipakai untuk memecahkannya.

Pada soal nomor (iv), relatif sukar mencari titik optimum fungsi yang memiliki banyak peubah. Untuk menentukan titik optimum (titik ekstrim fungsi), pertamanya orang harus menentukan turunan fungsi, menjadikan ruas kanannya sama dengan nol, lalu memeriksa jenis titik ekstrimnya. Bila fungsinya cukup rumit dan disusun oleh banyak peubah, menghitung turunan fungsi menjadi pekerjaan yang sukar atau bahkan tidak mungkin dilakukan.

Pertanyaan yang agak klasik sering muncul pada soal nomor (v): bagaimana menghitung nilai sebuah fungsi bila rumus fungsinya sendiri tidak diketahui? Kita semua tahu bahwa nilai fungsi diperoleh dengan cara menyulihkan (*substitute*) harga dari peubahnya ke dalam rumus fungsi. Masalahnya, bagaimana kalau persamaan fungsi tersebut tidak diketahui. Yang tersedia hanyalah beberapa buah data diskrit (*discrete*) dalam bentuk tabel. Persoalan semacam nomor (v) ini acapkali muncul pada pengamatan fenomena alam, baik berupa eksperimen di laboratorium maupun penelitian di lapangan yang melibatkan beberapa parameter (misalnya suhu, tekanan, waktu, dan sebagainya). Pengamat tidak mengetahui relasi yang menghubungkan parameter-parameter itu. Pengamat hanya dapat mengukur nilai-nilai parameter tersebut dengan menggunakan alat ukur seperti sensor, termometer, barometer, dan sebagainya. Tidak satupun metode analitik yang tersedia untuk menyelesaikan persoalan jenis ini. Begitu juga soal nomor (vi) melahirkan pertanyaan yang sama, bagaimana menghitung nilai turunan fungsi bila fungsinya sendiri tidak diketahui?.

Pada soal nomor (vii), tidak ada teknik integrasi yang dapat digunakan untuk fungsi yang bentuknya rumit itu. Begitu juga pada soal nomor (viii), tidak terdapat metode persamaan diferensial untuk menyelesaikannya. Dengan kata lain, persoalan (vii) dan (viii) tidak mempunyai solusi analitik.

## 1.1 Metode Analitik versus Metode Numerik

Contoh-contoh yang dikemukakan di atas memperlihatkan bahwa kebanyakan persoalan matematika tidak dapat diselesaikan dengan metode analitik. Metode analitik disebut juga **metode sejati** karena ia memberi kita **solusi sejati** (*exact solution*) atau solusi yang sesungguhnya, yaitu solusi yang memiliki **galat** (*error*) sama dengan nol! Sayangnya, metode analitik hanya unggul untuk sejumlah persoalan yang terbatas, yaitu persoalan yang memiliki tafsiran geometri sederhana serta bermatra rendah [CHA88]. Padahal persoalan yang muncul dalam dunia nyata seringkali nirlanjar serta melibatkan bentuk dan proses yang rumit. Akibatnya nilai praktis penyelesaian metode analitik menjadi terbatas.

Bila metode analitik tidak dapat lagi diterapkan, maka solusi persoalan sebenarnya masih dapat dicari dengan menggunakan **metode numerik**. Metode numerik adalah teknik yang digunakan untuk memformulasikan persoalan matematik sehingga dapat dipecahkan dengan operasi perhitungan/aritmetika biasa (tambah, kurang, kali, dan bagi). Metode artinya cara, sedangkan numerik artinya angka. Jadi metode numerik secara harafiah berarti cara berhitung dengan menggunakan angka-angka.

Perbedaan utama antara metode numerik dengan metode analitik terletak pada dua hal. Pertama, solusi dengan menggunakan metode numerik selalu berbentuk angka. Bandingkan dengan metode analitik yang biasanya menghasilkan solusi dalam bentuk fungsi matematik yang selanjutnya fungsi matematik tersebut dapat dievaluasi untuk menghasilkan nilai dalam bentuk angka.

Kedua, dengan metode numerik, kita hanya memperoleh solusi yang menghampiri atau mendekati solusi sejati sehingga solusi numerik dinamakan juga **solusi hampiran** (*approxomation*) atau **solusi pendekatan**, namun solusi hampiran dapat dibuat seteliti yang kita inginkan. Solusi hampiran jelas tidak tepat sama dengan solusi sejati, sehingga ada selisih antara keduanya. Selisih inilah yang disebut dengan galat (*error*).

Sebagai contoh ilustrasi penyelesaian dengan metode numerik, pandanglah sebuah persoalan integrasi-tentu berikut

$$I = \int_{-1}^1 (4 - x^2) dx \quad (\text{P.1.2})$$

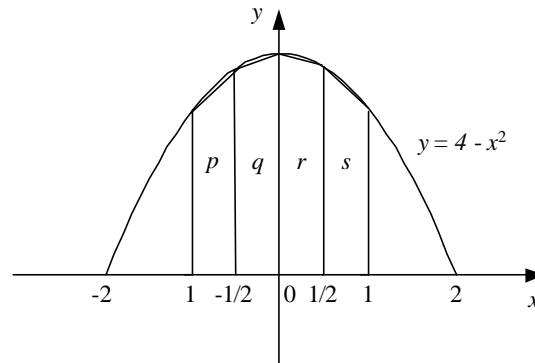
Dengan metode analitik, kita dapat menemukan solusi sejatinya dengan mudah. Di dalam kalkulus integral tentu kita mengetahui teknik pengintegralan untuk fungsi sederhana:

$$\int ax^n dx = \frac{1}{n+1} ax^{n+1} + C \quad (\text{P.1.3})$$

Maka, berdasarkan (P.1.3), kita dapat melakukan pengintegralan suku-suku dari fungsi integralnya lalu menghitung nilai integral-tentunya sebagai berikut:

$$I = \int_{-1}^1 (4 - x^2) dx = [4x - x^3/3]_{x=-1}^{x=1} = \{4(1) - (1)/3\} - \{4(-1) - (-1)/3\} = 22/3$$

Perhatikanlah bahwa  $4x - x^3/3$  adalah solusi analitik dalam bentuk fungsi matematik, sedangkan  $22/3$  adalah nilai numerik integral-tentu yang diperoleh dengan cara mengevaluasi fungsi matematik tersebut untuk batas-batas integrasi  $x = 1$  dan  $x = -1$ .



**Gambar 1.1** Integrasi  $f(x) = 4 - x^2$  secara numerik

Bandingkan penyelesaian di atas bila persoalan integrasi tersebut diselesaikan dengan metode numerik sebagai berikut. Sekali lagi, di dalam kalkulus integral kita tentu masih ingat bahwa interpretasi geometri integral  $f(x)$  dari  $x = a$  sampai  $x = b$  adalah luas daerah yang dibatasi oleh kurva  $f(x)$ , sumbu- $x$ , dan garis  $x = a$  dan  $x = b$ . Luas daerah tersebut dapat *dihampiri* dengan cara sebagai berikut. Bagilah daerah integrasi  $[-1, 1]$  atas sejumlah trapesium dengan lebar 0.5 (Gambar 1.1). Maka, luas daerah integrasi dihamperi dengan luas keempat buah trapesium, atau

$$\begin{aligned}
 I &\approx p + q + r + s & (P.1.4) \\
 &\approx \{[f(-1) + f(-1/2)] \times 0.5/2\} + \{[f(-1/2) + f(0)] \times 0.5/2\} + \\
 &\quad \{[f(0) + f(1/2)] \times 0.5/2\} + \{[f(1/2) + f(1)] \times 0.5/2\} \\
 &\approx 0.5/2 \{f(-1) + 2f(-1/2) + 2f(0) + 2f(1/2) + f(1)\} \\
 &\approx 0.5/2 \{3 + 7.5 + 8 + 7.5 + 3\} \\
 &\approx 7.25
 \end{aligned}$$

yang merupakan solusi hampiran (tanda “ $\approx$ ” artinya “kira-kira”) terhadap solusi sejati ( $22/3$ ). Galat solusi hampiran terhadap solusi sejati adalah

$$\text{galat} = |7.25 - 22/3| = |7.25 - 7.333\dots| = 0.08333\dots$$

Tentu saja kita dapat memperkecil galat ini dengan membuat lebar trapesium yang lebih kecil (yang artinya jumlah trapesium semakin banyak, yang berarti jumlah komputasi semakin banyak). Contoh ini juga memperlihatkan bahwa meskipun solusi dengan metode numerik merupakan hampiran, tetapi hasilnya

dapat dibuat seteliti mungkin dengan mengubah parameter komputasi (pada contoh perhitungan integral di atas, lebar trapesium yang dikurangi).

## 1.2 Metode Numerik dalam Bidang Rekayasa

Dalam bidang rekayasa, kebutuhan untuk menemukan solusi persoalan secara praktis adalah jelas. Dari kacamata rekayasawan, masih tampak banyak cara penyelesaian persoalan matematik yang dirasa terlalu sulit atau dalam bentuk yang kurang kongkrit. Penyelesaian analitik yang sering diberikan oleh kaum matematika kurang berguna bagi rekayasawan, karena ia harus dapat mentransformasikan solusi matematika yang sejati ke dalam bentuk berwujud yang biasanya meninggalkan kaidah kesejatiannya [BES97]. Solusi hampiran biasanya sudah memenuhi persyaratan rekayasa dan dapat diterima sebagai solusi. Lagipula, banyak persoalan matematika dalam bidang rekayasa yang hanya dapat dipecahkan secara hampiran. Kadang-kadang dapat pula terjadi bahwa metode analitik hanya menjamin keberadaan (atau hanya mengkarakteristikan beberapa properti umum) solusi, tetapi tidak memberikan cara menemukan solusi tersebut[KRE88].

Bagi rekayasawan, solusi yang diperoleh secara analitik kurang kurang berguna untuk tujuan numerik. Persoalan rekayasa dalam prakteknya tidak selalu membutuhkan solusi dalam bentuk fungsi matematika menerus (*continuous*). Rekayasawan seringkali menginginkan solusi dalam bentuk numerik, misalnya persoalan integral tentu dan persamaan diferensial. Sebuah contoh dalam termodinamika dikemukakan di bawah ini untuk memperjelas pernyataan ini [KRE88].

Sebuah bola logam dipanaskan sampai pada suhu 100°C. Kemudian, pada saat  $t = 0$ , bola itu dimasukkan ke dalam air yang bersuhu 30°C. Setelah 3 menit, suhu bola berkurang menjadi 70°C. Tentukan suhu bola setelah 22.78 menit. Diketahui tetapan pendinginan bola logam itu adalah 0.1865.

Dengan menggunakan hukum pendinginan Newton, laju pendinginan bola setiap detiknya adalah

$$dT/dt = -k(T - 30)$$

yang dalam hal ini  $k$  adalah tetapan pendinginan bola logam yang harganya 0.1865. Bagi matematikawan, untuk menentukan suhu bola pada  $t = 22.78$  menit, persamaan diferensial tersebut harus diselesaikan terlebih dahulu agar suhu  $T$  sebagai fungsi dari waktu  $t$  ditemukan. Persamaan diferensial ini dapat diselesaikan dengan metode kalkulus diferensial. Solusi umumnya adalah

$$T(t) = ce^{-kt} + 30$$

Nilai awal yang diberikan adalah  $T(0)=100$ . Dengan menggunakan nilai awal ini, solusi khusus persamaan diferensial adalah

$$T(t) = 70e^{-0.1865 t} + 30$$

Dengan menyulihkan  $t = 22.78$  ke dalam persamaan  $T$ , diperoleh

$$T(22.78) = 70e^{-0.1865 \times 22.78} + 30 = 31^\circ\text{C}.$$

Jadi, suhu bola setelah 22.78 menit adalah  $31^\circ\text{C}$ .

Bagi rekayasawan, solusi persamaan diferensial yang berbentuk fungsi menerus ini tidak terlalu penting (bahkan beberapa persamaan diferensial tidak dapat dicari solusi khususnya karena memang tidak ada teknik yang baku untuk menyelesaikannya). Dalam praktek di lapangan, seringkali para rekayasawan hanya ingin mengetahui berapa suhu bola logam setelah  $t$  tertentu misalnya setelah 30 menit tanpa perlu mencari solusi khususnya dalam bentuk fungsi terlebih dahulu. Rekayasawan cukup memodelkan sistem ke dalam persamaan diferensial, lalu solusi untuk  $t$  tertentu dicari secara numerik.

## 1.3 Apakah Metode Numerik Hanya untuk Persoalan Matematika yang Rumit Saja?

Tentu saja tidak! Anda jangan berpikiran bahwa metode numerik hanya dapat menyelesaikan persoalan rumit saja. Metode numerik berlaku umum, yakni ia dapat diterapkan untuk menyelesaikan persoalan matematika sederhana (yang juga dapat diselesaikan dengan metode analitik) maupun persoalan matematika yang tergolong rumit (yang metode analitik pun belum tentu dapat menyelesaikannya). Sebagai contoh, dengan metode numerik kita dapat menghitung integral

$$\int_0^p \sqrt{1 + \cos^2(x)} dx$$

sama mudahnya menghitung

$$\int_0^1 2x^2 dx$$



## 1.4 Peranan Komputer dalam Metode Numerik

Komputer berperan besar dalam perkembangan bidang metode numerik. Hal ini mudah dimengerti karena perhitungan dengan metode numerik adalah berupa operasi aritmetika seperti penjumlahan, perkalian, pembagian, plus membuat perbandingan. Sayangnya, jumlah operasi aritmetika ini umumnya sangat banyak dan berulang, sehingga perhitungan secara manual sering menjemukan. Manusia (yang melakukan perhitungan manual ini) dapat membuat kesalahan dalam melakukannya. Dalam hal ini, komputer berperanan mempercepat proses perhitungan tanpa membuat kesalahan.

Penggunaan komputer dalam metode numerik antara lain untuk memprogram. Langkah-langkah metode numerik diformulasikan menjadi program komputer. Program ditulis dengan bahasa pemrograman tertentu, seperti FORTRAN, PASCAL, C, C++, BASIC, dan sebagainya.

Sebenarnya, menulis program numerik tidak selalu diperlukan. Di pasaran terdapat banyak program aplikasi komersil yang langsung dapat digunakan. Beberapa contoh aplikasi yang ada saat ini adalah *MathLab*, *MathCad*, *Maple*, *Mathematica*, *Eureka*, dan sebagainya. Selain itu, terdapat juga *library* yang berisi rutin-rutin yang siap digabung dengan program utama yang ditulis pengguna, misalnya *IMSL (International Mathematical and Statistical Library)* *Math/Library* yang berisi ratusan rutin-rutin metode numerik.

Selain mempercepat perhitungan numerik, dengan komputer kita dapat mencoba berbagai kemungkinan solusi yang terjadi akibat perubahan beberapa parameter. Solusi yang diperoleh juga dapat ditingkatkan ketelitiannya dengan mengubah-ubah nilai parameter.

Kemajuan komputer digital telah membuat bidang metode numerik berkembang secara dramatis. Tidak ada bidang matematika lain yang mengalami kemajuan penting secepat metode numerik. Tentu saja alasan utama penyebab kemajuan ini adalah perkembangan komputer itu sendiri, dari komputer mikro sampai komputer *Cray*, dan kita melihat perkembangan teknologi komputer tidak pernah berakhir. Tiap generasi baru komputer menghadirkan keunggulan seperti waktu, memori, ketelitian, dan kestabilan perhitungan. Hal ini membuat ruang penelitian semakin terbuka luas. Tujuan utama penelitian itu adalah pengembangan algoritma numerik yang lebih baik dengan memanfaatkan keunggulan komputer semaksimal

mungkin. Banyak algoritma baru lahir atau perbaikan algoritma yang lama didukung oleh komputer.

Bagian mendasar dari perhitungan rekayasa yang dilakukan saat ini adalah perhitungan "waktu nyata" (*real time computing*), yaitu perhitungan keluaran (hasil) dari data yang diberikan dilakukan secara simultan dengan *event* pembangkitan data tersebut, sebagaimana yang dibutuhkan dalam mengendalikan proses kimia atau reaksi nuklir, memandu pesawat udara atau roket dan sebagainya [KRE88]. Karena itu, kecepatan perhitungan dan kebutuhan memori komputer adalah pertimbangan yang sangat penting.

Jelaslah bahwa kecepatan tinggi, keandalan, dan fleksibilitas komputer memberikan akses untuk penyelesaian masalah praktek. Sebagai contoh, solusi sistem persamaan linier yang besar menjadi lebih mudah dan lebih cepat diselesaikan dengan komputer. Perkembangan yang cepat dalam metode numerik antara lain ialah penemuan metode baru, modifikasi metode yang sudah ada agar lebih mangkus, analisis teoritis dan praktis algoritma untuk proses perhitungan baku, pengkajian galat, dan penghilangan jebakan yang ada pada metode [KRE88].

## 1.5 Mengapa Kita Harus Mempelajari Metode Numerik?

Seperti sudah disebutkan pada bagian awal bab ini, para rekayasawan dan para ahli ilmu alam, dalam pekerjaannya sering berhadapan dengan persamaan matematik. Persoalan yang muncul di lapangan diformulasikan ke dalam model yang berbentuk persamaan matematika. Persamaan tersebut mungkin sangat kompleks atau jumlahnya lebih dari satu. Metode numerik, dengan bantuan komputer, memberikan cara penyelesaian persoalan matematika dengan cepat dan akurat.

Terdapat beberapa alasan tambahan mengapa kita harus mempelajari metode numerik [CHA91]:

1. Metode numerik merupakan alat bantu pemecahan masalah matematika yang sangat ampuh. Metode numerik mampu menangani sistem persamaan besar, kenirlanjaran, dan geometri yang rumit yang dalam praktek rekayasa seringkali tidak mungkin dipecahkan secara analitik.
2. Seperti sudah disebutkan pada upapab 1.4, di pasaran banyak tersedia program aplikasi numerik komersil. Penggunaan aplikasi tersebut menjadi lebih berarti

bila kita memiliki pengetahuan metode numerik agar kita dapat memahami cara paket tersebut menyelesaikan persoalan.

3. Kita dapat membuat sendiri program komputer tanpa harus membeli paket programnya. Seringkali beberapa persoalan matematika yang tidak selalu dapat diselesaikan oleh program aplikasi. Sebagai contoh, misalkan ada program aplikasi tertentu yang tidak dapat dipakai untuk menghitung integrasi lipat dua,  $\iint$ , atau lipat tiga,  $\iiint$ . Mau tidak mau, kita harus menulis sendiri programnya. Untuk itu, kita harus mempelajari cara pemecahan integral lipat dua atau lebih dengan metode numerik.
4. Metode numerik menyediakan sarana untuk memperkuat kembali pemahaman matematika. Karena, metode numerik ditemukan dengan menyederhanakan matematika yang lebih tinggi menjadi operasi matematika yang mendasar.

## 1.6 Tahap-Tahap Memecahkan Persoalan Secara Numerik

Ada enam tahap yang dilakukan dalam pemecahan persoalan dunia nyata dengan metode numerik, yaitu

### 1. **Pemodelan**

Ini adalah tahap pertama. Persoalan dunia nyata dimodelkan ke dalam persamaan matematika (lihat contoh ilustrasi pada bab 1.2)

### 2. **Penyederhanaan model**

Model matematika yang dihasilkan dari tahap 1 mungkin saja terlalu kompleks, yaitu memasukkan banyak peubah (variable) atau parameter. Semakin kompleks model matematikanya, semakin rumit penyelesaiannya. Mungkin beberapa andaian dibuat sehingga beberapa parameter dapat diabaikan. Contohnya, faktor gesekan udara diabaikan sehingga koefisien gesekan di dalam model dapat dibuang. Model matematika yang diperoleh dari penyederhanaan menjadi lebih sederhana sehingga solusinya akan lebih mudah diperoleh.

### 3. **Formulasi numerik**

Setelah model matematika yang sederhana diperoleh, tahap selanjutnya adalah memformulasikannya secara numerik, antara lain:

- a. menentukan metode numerik yang akan dipakai bersama-sama dengan analisis galat awal (yaitu taksiran galat, penentuan ukuran langkah, dan sebagainya).

Pemilihan metode didasari pada pertimbangan:

- apakah metode tersebut teliti?
- apakah metode tersebut mudah diprogram dan waktu pelaksanaannya cepat?
- apakah metode tersebut tidak peka terhadap perubahan data yang cukup kecil?

b. menyusun algoritma dari metode numerik yang dipilih.

#### 4. **Pemrograman**

Tahap selanjutnya adalah menerjemahkan algoritma ke dalam program komputer dengan menggunakan salah satu bahasa pemrograman yang dikuasai.

#### 5. **Operasional**

Pada tahap ini, program komputer dijalankan dengan data uji coba sebelum data yang sesungguhnya.

#### 6. **Evaluasi**

Bila program sudah selesai dijalankan dengan data yang sesungguhnya, maka hasil yang diperoleh diinterpretasi. Interpretasi meliputi analisis hasil *run* dan membandingkannya dengan prinsip dasar dan hasil-hasil empirik untuk menaksir kualitas solusi numerik, dan keputusan untuk menjalankan kembali program dengan untuk memperoleh hasil yang lebih baik.

## 1.7 Peran Ahli Informatika dalam Metode Numerik

Dari tahap-tahap pemecahan yang dikemukakan di atas, tahap 1 dan 2 melibatkan para pakar di bidang persoalan yang bersangkutan. Kalau persoalannya dalam bidang teknik Sipil, maka orang dari bidang Sipil-lah yang menurunkan model matematikanya. Kalau persoalannya menyangkut bidang Teknik Kimia (TK), maka ahli Teknik Kimia-lah yang mempunyai kemampuan membentuk model matematikanya.

Dimanakah peran orang Informatika? Orang Informatika baru berperan pada tahap 3 dan 4, dan 5. Tetapi, agar lebih memahami dan menghayati persoalan, sebaiknya orang Informatika juga ikut dilibatkan dalam memodelkan, namun perannya hanyalah sebagai pendengar.

Tahap 6 memerlukan kerjasama informatikawan dengan pakar bidang bersangkutan. Bersama-sama dengan pakar, informatikawan mendiskusikan hasil numerik yang diperoleh, apakah hasil tersebut sudah dapat diterima, apakah perlu dilakukan perubahan parameter, dsb.

## 1.8 Perbedaan Metode Numerik dengan Analisis Numerik

Untuk persoalan tertentu tidaklah cukup kita hanya menggunakan metode untuk memperoleh hasil yang diinginkan; kita juga perlu mengetahui apakah metode tersebut memang memberikan solusi hampiran, dan seberapa bagus hampiran itu [BUC92]. Hal ini melahirkan kajian baru, yaitu **analisis numerik**.

Metode numerik dan analisis numerik adalah dua hal yang berbeda. Metode adalah algoritma, menyangkut langkah-langkah penyelesaian persoalan secara numerik, sedangkan analisis numerik adalah terapan matematika untuk menganalisis metode [NOB72]. Dalam analisis numerik, hal utama yang ditekankan adalah analisis galat dan kecepatan konvergensi sebuah metode. Teorema-teorema matematika banyak dipakai dalam menganalisis suatu metode. Di dalam buku ini, kita akan memasukkan beberapa materi analisis numerik seperti galat metode dan kekonvergenan metode.

Tugas para analis numerik ialah mengembangkan dan menganalisis metode numerik. Termasuk di dalamnya pembuktian apakah suatu metode konvergen, dan menganalisis batas-batas galat solusi numerik. Terdapat banyak sumber galat, diantaranya tingkat ketelitian model matematika, sistem aritmetik komputer, dan kondisi yang digunakan untuk menghentikan proses pencarian solusi. Semua ini harus dipertimbangkan untuk menjamin ketelitian solusi akhir yang dihitung.

## 1.9 Materi Apa yang Terdapat di dalam Buku Ini?

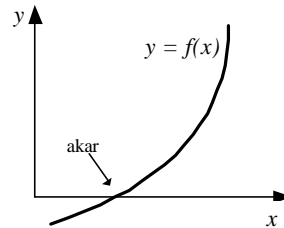
Ada enam pokok bahasan yang ditulis di dalam buku ini:

1. Solusi persamaan nirlanjar.
2. Solusi sistem persamaan lanjar.
3. Interpolasi polinom.
4. Turunan numerik.
5. Integrasi numerik.
6. Solusi persamaan diferensial biasa dengan nilai awal.

Ringkasan masing-masing pokok bahasan 1 sampai 6 adalah sebagai berikut:

### 1. Solusi persamaan nirlanjar

Selesaikan  $f(x) = 0$  untuk  $x$ .



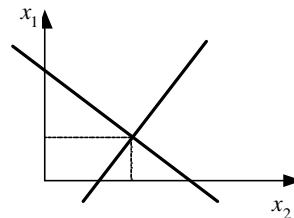
### 2. Solusi sistem persamaan lanjar

Selesaikan sistem persamaan lanjar

$$a_{11}x_1 + a_{12}x_2 = c_1$$

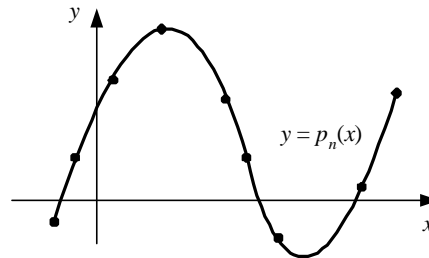
$$a_{21}x_1 + a_{22}x_2 = c_2$$

untuk harga-harga  $x_1$  dan  $x_2$ .



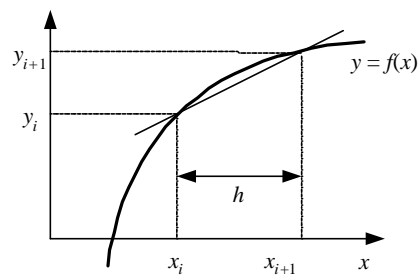
### 3. Interpolasi polinom

Diberikan titik-titik  $(x_0, y_0)$ ,  $(x_1, y_1)$ , ...,  $(x_n, y_n)$ . Tentukan polinom  $p_n(x)$  yang melalui semua titik tersebut



### 4. Turunan numerik

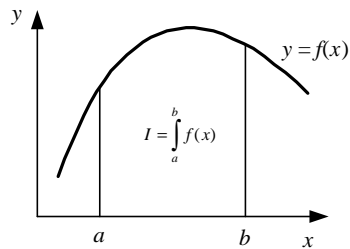
Diberikan titik  $(x_i, y_i)$  dan titik  $(x_{i+1}, y_{i+1})$ . Tentukan  $f'(x_i)$ .



## 5. Integrasi numerik

Hitung integral

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

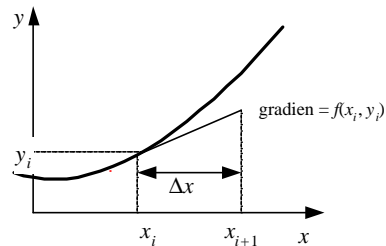


## 6. Solusi persamaan diferensial biasa dengan nilai awal

Diberikan  $dy/dx = f(x, y)$  dan

dengan nilai awal  $y_0 = y(x_0)$

Tentukan nilai  $y(x_i)$  untuk  $x_i \in R$



Sebelum menjelaskan keenam pokok bahasan tersebut, kita perlu terlebih dahulu mengerti konsep galat dalam metode numerik. Konsep galat diberikan sebagai topik tersendiri.

Perjalanan seribu mil dimulai dari satu langkah  
(pepatah)

## Bab 2

# Deret Taylor dan Analisis Galat

---

Matematik selalu memperlihatkan rasa ingin tahu untuk dapat diterapkan di alam, dan ini dapat mengungkapkan kaitan yang dalam antara pikiran kita dan alam. Kita membicarakan semesta, bagian dari alam. Jadi, tidak mengherankan bahwa sistem logik dan matematika kita bernyanyi seirama dengan alam.  
(George Zebrowski)

Columbus menemukan Amerika melalui kesalahan.  
(Majalah Intisari)

Prasyarat yang diperlukan untuk mempelajari metode numerik adalah matematika. Matematika adalah ilmu dasar, jadi anda diharapkan sudah memiliki pengetahuan mengenai konsep fungsi, geometri, konsep kalkulus seperti turunan dan integral, dan sebagainya. Tidak paham terlalu dalam tidak apa, yang penting anda mengerti.

Banyak teorema matematika yang dipakai di sini. Dari sekian banyak teorema tersebut, ada satu teorema yang menjadi **kakas** (*tools*) yang sangat penting dalam metode numerik, yaitu teorema **deret Taylor**. Deret Taylor adalah kakas yang utama untuk menurunkan suatu metode numerik.

Pada bagian yang lain, kita akan membahas konsep galat. Seperti sudah dijelaskan di dalam Bab 1, solusi yang diperoleh secara numerik adalah nilai hampiran dari solusi sejati. Ini berarti terdapat galat (*error*) pada solusi hampiran tersebut. Bab 2 ini akan menjelaskan konsep galat, cara mengukur galat, penyebab galat, perambatan galat, dan ketidakstabilan perhitungan akibat galat.



## 2.1 Deret Taylor

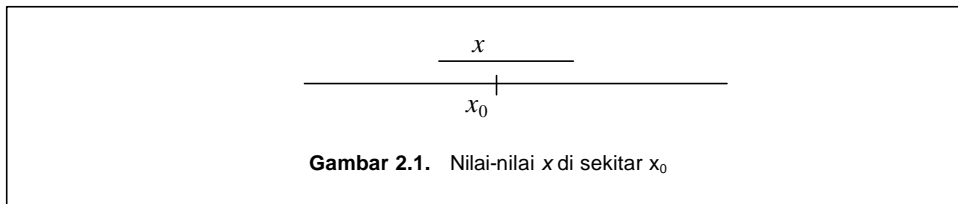
Kebanyakan dari metode-metode numerik yang diturunkan didasarkan pada penghampiran fungsi ke dalam bentuk polinom. Fungsi yang bentuknya kompleks menjadi lebih sederhana bila dihamperi dengan polinom, karena polinom merupakan bentuk fungsi yang paling mudah dipahami kelakuannya.

Kalau perhitungan dengan fungsi yang sesungguhnya menghasilkan solusi sejati, maka perhitungan dengan fungsi hampiran menghasilkan solusi hampiran. Pada bab 1 sudah dikatakan bahwa solusi numerik merupakan pendekatan (hampiran) terhadap solusi sejati, sehingga terdapat galat sebesar selisih antara solusi sejati dengan solusi hampiran. Galat pada solusi numerik harus dihubungkan dengan seberapa teliti polinom menghampiri fungsi sebenarnya. Kakas yang digunakan untuk membuat polinom hampiran adalah **deret Taylor**.

### Definisi Deret Taylor

Andaikan  $f$  dan semua turunannya,  $f', f'', f''', \dots$ , menerus di dalam selang  $[a, b]$ . Misalkan  $x_0 \in [a, b]$ , maka untuk nilai-nilai  $x$  di sekitar  $x_0$  (Gambar 2.1) dan  $x \in [a, b]$ ,  $f(x)$  dapat diperluas (diekspansi) ke dalam deret Taylor:

$$f(x) = f(x_0) + \frac{(x-x_0)}{1!} f'(x_0) + \frac{(x-x_0)^2}{2!} f''(x_0) + \dots + \frac{(x-x_0)^m}{m!} f^{(m)}(x_0) + \dots \quad (\text{P.2.1})$$



Persamaan (P.2.1) merupakan penjumlahan dari suku-suku (*term*), yang disebut **deret**. Perhatikanlah bahwa deret Taylor ini panjangnya tidak berhingga sehingga untuk memudahkan penulisan suku-suku selanjutnya kita menggunakan tanda elipsis (...). Jika dimisalkan  $x - x_0 = h$ , maka  $f(x)$  dapat juga ditulis sebagai

$$f(x) = f(x_0) + \frac{h}{1!} f'(x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^3}{3!} f'''(x_0) + \dots + \frac{h}{m!} f^{(m)}(x_0) + \dots \quad (\text{P.2.2})$$

**Contoh 2.1**

Hampiri fungsi  $f(x) = \sin(x)$  ke dalam deret Taylor di sekitar  $x_0 = 1$ .

**Penyelesaian:**

Kita harus menentukan turunan  $\sin(x)$  terlebih dahulu sebagai berikut

$$\begin{aligned} f(x) &= \sin(x), \\ f'(x) &= \cos(x), \\ f''(x) &= -\sin(x), \\ f'''(x) &= -\cos(x), \\ f^{(4)}(x) &= \sin(x), \\ &\text{dan seterusnya.} \end{aligned}$$

Maka, berdasarkan (P.2.1),  $\sin(x)$  dihampiri dengan deret Taylor sebagai berikut:

$$\sin(x) = \sin(1) + \frac{(x-1)}{1!} \cos(1) + \frac{(x-1)^2}{2!} (-\sin(1)) + \frac{(x-1)^3}{3!} (-\cos(1)) + \frac{(x-1)^4}{4!} \sin(1) + \dots$$

Bila dimisalkan  $x - 1 = h$ , maka, berdasarkan (P.2.2),

$$\begin{aligned} \sin(x) &= \sin(1) + h \cos(1) - \frac{h^2}{2} \sin(1) - \frac{h^3}{6} \cos(1) + \frac{h^4}{24} \sin(1) + \dots \\ &= 0.8415 + 0.5403h - 0.4208h^2 - 0.0901h^3 + 0.0351h^4 + \dots \end{aligned}$$

■

Kasus khusus adalah bila fungsi diperluas di sekitar  $x_0 = 0$ , maka deretnya dinamakan **deret Maclaurin**, yang merupakan deret Taylor baku. Kasus  $x_0 = 0$  paling sering muncul dalam praktek.

**Contoh 2.2**

Uraikan  $\sin(x)$ ,  $e^x$ ,  $\cos(x)$ , dan  $\ln(x+1)$  masing-masing ke dalam deret Maclaurin.

**Penyelesaian:**

Beberapa turunan  $\sin(x)$  sudah dihitung pada Contoh 2.1. Deret Maclaurin dari  $\sin(x)$  adalah

$$\begin{aligned} \sin(x) &= \sin(0) + \frac{(x-0)}{1!} \cos(0) + \frac{(x-0)^2}{2!} (-\sin(0)) + \frac{(x-0)^3}{3!} (-\cos(0)) + \frac{(x-0)^4}{4!} \sin(0) + \dots \\ &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \end{aligned}$$

Untuk menentukan deret Maclaurin dari  $e^x$ , kita harus menentukan turunan  $e^x$  terlebih dahulu sebagai berikut:  $f(x) = e^x$ ,  $f'(x) = e^x$ ,  $f''(x) = e^x$ ,  $f'''(x) = e^x$ ,  $f^{(4)}(x) = e^x$ , dan seterusnya. Deret Maclaurin dari  $e^x$  adalah

$$\begin{aligned} e^x &= e^{(0)} + \frac{(x-0)}{1!} e^{(0)} + \frac{(x-0)^2}{2!} e^{(0)} + \frac{(x-0)^3}{3!} e^{(0)} + \frac{(x-0)^4}{4!} e^{(0)} + \dots \\ &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots \end{aligned}$$

Untuk menentukan deret Maclaurin dari  $\cos(x)$ , kita harus menentukan turunan  $\cos(x)$  terlebih dahulu sebagai berikut:  $f(x) = \cos(x)$ ,  $f'(x) = -\sin(x)$ ,  $f''(x) = -\cos(x)$ ,  $f'''(x) = \sin(x)$ ,  $f^{(4)}(x) = \cos(x)$ , dan seterusnya. Deret Maclaurin dari  $\cos(x)$  adalah

$$= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Untuk menentukan deret Maclaurin dari  $\ln(x+1)$ , kita harus menentukan turunan  $\ln(x+1)$  terlebih dahulu sebagai berikut:  $f(x) = \ln(x+1)$ ,  $f'(x) = (x+1)^{-1}$ ,  $f''(x) = -(x+1)^{-2}$ ,  $f'''(x) = 2(x+1)^{-3}$ ,  $f^{(4)}(x) = -6(x+1)^{-4}$ , dan seterusnya. Deret Maclaurin dari  $\ln(x+1)$  adalah

$$\begin{aligned} \ln(x+1) &= \ln(0+1) + \frac{(x-0)}{1!} (0+1)^{-1} + \frac{(x-0)^2}{2!} (-(0+1)^{-2}) + \frac{(x-0)^3}{3!} 2(0+1)^{-3} + \frac{(x-0)^4}{4!} (-6(0+1)^{-4}) + \dots \\ &= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots \end{aligned} \quad \blacksquare$$

Karena suku-suku deret Taylor tidak berhingga banyaknya, maka -untuk alasan praktis- deret Taylor dipotong sampai suku orde tertentu. Deret Taylor yang dipotong sampai suku orde ke- $n$  dinamakan **deret Taylor terpotong** dan dinyatakan oleh:

$$f(x) \approx f(x_0) + \frac{(x-x_0)}{1!} f'(x_0) + \frac{(x-x_0)^2}{2!} f''(x_0) + \dots + \frac{(x-x_0)^n}{n!} f^{(n)}(x_0) + R_n(x) \quad (\text{P.2.3})$$

yang dalam hal ini,

$$R_n(x) = \frac{(x-x_0)^{(n+1)}}{(n+1)!} f^{(n+1)}(c) \quad , \quad x_0 < c < x \quad (\text{P.2.4})$$

disebut **galat** atau **sis**a (residu).

Dengan demikian deret Taylor yang dipotong sampai suku orde ke- $n$  dapat ditulis sebagai

$$\tilde{f}(x) = P_n(x) + R_n(x) \quad (\text{P.2.5})$$

yang dalam hal ini,

$$P_n(x) = \sum_{k=1}^n \frac{(x-x_0)^k}{k!} f^{(k)}(x_0) \quad (\text{P.2.6})$$

$$R_n(x) = \frac{(x-x_0)^{(n+1)}}{(n+1)!} f^{(n+1)}(c) \quad , \quad x_0 < c < x \quad (\text{P.2.7})$$

Sebagai contoh,  $\sin(x)$  pada Contoh 2.1 jika dihampiri dengan deret Taylor orde 4 di sekitar  $x_0 = 1$  adalah:

$$\sin(x) = \sin(1) + \frac{(x-1)}{1!} \cos(1) - \frac{(x-1)^2}{2!} \sin(1) - \frac{(x-1)^3}{3!} \cos(1) + \frac{(x-1)^4}{4!} \sin(1) + R_4(x)$$

yang dalam hal ini,

$$R_4(x) = \frac{(x-1)^5}{5!} \cos(c) \quad , \quad 1 < c < x$$

Deret Taylor terpotong di sekitar  $x_0 = 0$  disebut deret Maclaurin terpotong. Berdasarkan Contoh 2.2, deret MacLaurin terpotong untuk  $\sin(x)$ ,  $e^x$ ,  $\cos(x)$ , dan  $\ln(x+1)$  adalah

$$\sin(x) = x - x^3/3! + x^5/5! + R_5(x) ; R_5(x) = \frac{x^6}{6!} \cos(c) \quad (\text{sampai suku orde 5})$$

$$e^x = 1 + x + x^2/2! + x^3/3! + x^4/4! + R_4(x) ; R_4(x) = \frac{x^5}{5!} e^c \quad (\text{sampai suku orde 4})$$

$$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + R_6(x) ; R_6(x) = \frac{x^7}{7!} \cos(c) \quad (\text{sampai suku orde 6})$$

$$\ln(x+1) = x - x^2/2 + x^3/3 - x^4/4 ; R_4(x) = 24 \frac{x^5}{5!} (c+1)^{-5} \quad (\text{sampai suku orde 4})$$

yang dalam hal ini,  $0 < c < x$ .

Fungsi pustaka matematika di dalam kalkulator dan *compiler* bahasa pemrograman dapat menggunakan deret Taylor untuk mengevaluasi nilai-nilai fungsi trigonometri dan fungsi transenden yang tidak dapat dihitung secara langsung.

### Contoh 2.3

Hitunglah hampiran nilai  $\cos(0.2)$ , sudut dinyatakan dalam radian, dengan deret Maclaurin sampai suku orde  $n = 6$ .

#### Penyelesaian:

Dari hasil pada Contoh 2.2,

$$\cos(0.2) \approx 1 - 0.2^2/2 + 0.2^4/24 - 0.2^6/720 = 0.9800667$$

(sampai 7 angka di belakang koma) ■

Program 2.1 di bawah ini menghitung nilai hampiran  $\cos(x)$ . Setiap suku kita hitung nilainya, jika nilainya lebih kecil dari toleransi (epsilon) yang kita spesifikasikan, maka perhitungan suku berikutnya dapat dihentikan.

**Program 2.1** Program menghitung hampiran nilai  $\cos(x)$

```
function cos(x:real):real;
{mengembalikan nilai cosinus sampai nilai suatu suku < e }
const
  epsilon = 0.00000001; { toleransi nilai tiap suku }
  { jika nilai suku sudah lebih kecil dari epsilon, perhitungan cos dapat
    dihentikan }
var
  tanda, n      : integer;
  suku, jumlah : real;

  (* function pangkat(x:real; n:integer):real;
    { Menghitung nilai x^n } *)

  (* function faktorial(n:integer):integer;
    { Menghitung n! } *)

begin
  suku:=1;      {suku pertama deret cosinus}
  tanda:=1;     {tanda (+/-) suku pertama}
  n:=0;         {orde suku pertama}
  jumlah:=0;    {jumlah deret cosinus, inisialisasi dengan 0}
  while abs(suku) >= epsilon do
    begin
      jumlah:=jumlah + suku; {jumlah deret cosinus}
      n:=n+2;               {orde suku berikutnya}
      tanda:=-tanda;        {tanda suku berikutnya}
```

```

        suku:=tanda*pangkat(x,n)/faktorial(n);
    end;
    { abs(suku) < epsilon }
    cos:=jumlah;
end;

```

Deret Taylor banyak digunakan untuk menurunkan metode-metode numerik. Deret Taylor yang terpotong digunakan sebagai titik awal dalam menurunkan metode. Anda sebaiknya dapat menguasai deret Taylor terlebih dahulu sebagai alat bantu yang penting dalam metode numerik.

## 2.2 Analisis Galat

Menganalisis galat sangat penting di dalam perhitungan yang menggunakan metode numerik. Galat berasosiasi dengan seberapa dekat solusi hampiran terhadap solusi sejatinya. Semakin kecil galatnya, semakin teliti solusi numerik yang didapatkan. Kita harus memahami dua hal: (a) bagaimana menghitung galat, dan (b) bagaimana galat timbul.

Misalkan  $\hat{a}$  adalah nilai hampiran terhadap nilai sejati  $a$ , maka selisih

$$\mathbf{e} = a - \hat{a} \quad (\text{P.2.8})$$

disebut **galat**. Sebagai contoh, jika  $\hat{a} = 10.5$  adalah nilai hampiran dari  $a = 10.45$ , maka galatnya adalah  $\mathbf{e} = -0.01$ . Jika tanda galat (positif atau negatif) tidak dipertimbangkan, maka **galat mutlak** dapat didefinisikan sebagai

$$|\mathbf{e}| = |a - \hat{a}| \quad (\text{P.2.9})$$

Sayangnya, ukuran galat  $\mathbf{e}$  kurang bermakna sebab ia tidak menceritakan seberapa besar galat itu dibandingkan dengan nilai sejatinya. Sebagai contoh, seorang anak melaporkan panjang sebatang kawat 99 cm, padahal panjang sebenarnya 100 cm. Galatnya adalah  $100 - 99 = 1$  cm. Anak yang lain melaporkan panjang sebatang pensil 9 cm, padahal panjang sebenarnya 10 cm, sehingga galatnya juga 1 cm. Kedua galat pengukuran sama-sama bernilai 1 cm, namun galat 1 cm pada pengukuran panjang pensil lebih berarti daripada galat 1 cm pada pengukuran panjang kawat. Jika tidak ada informasi mengenai panjang sesungguhnya, kita mungkin menganggap kedua galat tersebut sama saja. Untuk mengatasi interpretasi nilai galat ini, maka galat harus *dinormalkan* terhadap nilai sejatinya. Gagasan ini melahirkan apa yang dinamakan **galat relatif**.

Galat relatif didefinisikan sebagai

$$e_R = \frac{e}{a} \quad (\text{P.2.10})$$

atau dalam persentase

$$e_R = \frac{e}{a} \times 100\% \quad (\text{P.2.11})$$

Karena galat dinormalkan terhadap nilai sejati, maka galat relatif tersebut dinamakan juga **galat relatif sejati**. Dengan demikian, pengukuran panjang kawat mempunyai galat relatif sejati =  $1/100 = 0.01$ , sedangkan pengukuran panjang pensil mempunyai galat relatif sejati =  $1/10 = 0.1$ .

Dalam praktek kita tidak mengetahui nilai sejati  $a$ , karena itu galat  $e$  seringkali dinormalkan terhadap solusi hampirannya, sehingga galat relatifnya dinamakan **galat relatif hampiran**:

$$e_{RA} = \frac{e}{\hat{a}} \quad (\text{P.2.12})$$

#### Contoh 2.4

Misalkan nilai sejati =  $10/3$  dan nilai hampiran =  $3.333$ . Hitunglah galat, galat mutlak, galat relatif, dan galat relatif hampiran.

#### Penyelesaian:

$$\begin{aligned} \text{galat} &= 10/3 - 3.333 = 10/3 - 3333/1000 = 1/3000 = 0.000333... \\ \text{galat mutlak} &= |0.000333...| = 0.000333... \\ \text{galat relatif} &= (1/3000)/(10/3) = 1/1000 = 0.0001 \\ \text{galat relatif hampiran} &= (1/3000)/3.333 = 1/9999 \end{aligned} \quad \blacksquare$$

Galat relatif hampiran yang dihitung dengan persamaan (P.2.12) masih mengandung kelemahan sebab nilai  $e$  tetap membutuhkan pengetahuan nilai  $a$  (dalam praktek kita jarang sekali mengetahui nilai sejati  $a$ ). Oleh karena itu, perhitungan galat relatif hampiran menggunakan pendekatan lain. Pada perhitungan numerik yang menggunakan pendekatan lelaran (*iteration*),  $e_{RA}$  dihitung dengan cara

$$e_{RA} = \frac{a_{r+1} - a_r}{a_{r+1}} \quad (\text{P.2.13})$$

yang dalam hal ini  $a_{r+1}$  adalah nilai hampiran lelaran sekarang dan  $a_r$  adalah nilai hampiran lelaran sebelumnya. Proses lelaran dihentikan bila

$$|e_{RA}| < e_S$$

yang dalam hal ini  $e_S$  adalah toleransi galat yang dispesifikasikan. Nilai  $e_S$  menentukan ketelitian solusi numerik. Semakin kecil nilai  $e_S$ , semakin teliti solusinya, namun semakin banyak proses lelarannya. Contoh 2.5 mengilustrasikan hal ini.

### Contoh 2.5

Misalkan ada prosedur lelaran sebagai berikut

$$x_{r+1} = (-x_r^3 + 3)/6, \quad r = 0, 1, 2, 3, \dots$$

Lelaran dihentikan bila kondisi  $|e_{RA}| < e_S$ , dalam hal ini  $e_S$  adalah toleransi galat yang diinginkan. Misalkan dengan memberikan  $x_0 = 0.5$ , dan  $e_S = 0.00001$  kita memperoleh runtunan:

$$\begin{aligned} x_0 &= 0.5 \\ x_1 &= 0.4791667 \quad ; \quad |e_{RA} = (x_1 - x_0)/x_1| = 0.043478 > e_S \\ x_2 &= 0.4816638 \quad ; \quad |e_{RA} = (x_2 - x_1)/x_2| = 0.0051843 > e_S \\ x_3 &= 0.4813757 \quad ; \quad |e_{RA} = (x_3 - x_2)/x_3| = 0.0005984 > e_S \\ x_4 &= 0.4814091 \quad ; \quad |e_{RA} = (x_4 - x_3)/x_4| = 0.0000693 > e_S \\ x_5 &= 0.4814052. \quad ; \quad |e_{RA} = (x_5 - x_4)/x_5| = 0.0000081 < e_S, \text{ berhenti!} \end{aligned}$$

Pada lelaran ke-5,  $|e_{RA}| < e_S$  sudah terpenuhi sehingga lelaran dapat dihentikan. ■

## 2.3 Sumber Utama Galat Numerik

Secara umum terdapat dua sumber utama penyebab galat dalam perhitungan numerik:

1. **Galat pemotongan** (*truncation error*)
2. **Galat pembulatan** (*round-off error*)

Selain kedua galat ini, masih ada sumber galat lain, antara lain [KRE88]:

- a. **Galat eksperimental**, yaitu galat yang timbul dari data yang diberikan, misalnya karena kesalahan pengukuran, ketidakteelitian alat ukur, dan sebagainya
- b. **Galat pemrograman**. Galat yang terdapat di dalam program sering dinamakan dengan kutu (*bug*), dan proses penghilangan galat ini dinamakan penirkutan (*debugging*).



Kita tidak akan membicarakan kedua galat terakhir ini karena kontribusinya terhadap galat keseluruhan tidak selalu ada. Akan halnya galat utama, galat pemotongan dan galat pembulatan, keduanya selalu muncul pada solusi numerik. Terhadap kedua galat inilah perhatian kita fokuskan.

### 2.3.1 Galat Pemotongan

Galat pemotongan mengacu pada galat yang ditimbulkan akibat penggunaan hampiran sebagai pengganti formula eksak. Maksudnya, ekspresi matematik yang lebih kompleks “diganti” dengan formula yang lebih sederhana. Tipe galat pemotongan bergantung pada metode komputasi yang digunakan untuk penghampiran sehingga kadang-kadang ia disebut juga **galat metode**. Misalnya, turunan pertama fungsi  $f$  di  $x_i$  dihampiri dengan formula

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

yang dalam hal ini  $h$  adalah lebar absis  $x_{i+1}$  dengan  $x_i$ . Galat yang ditimbulkan dari penghampiran turunan tersebut merupakan galat pemotongan.

Istilah “pemotongan” muncul karena banyak metode numerik yang diperoleh dengan penghampiran fungsi menggunakan deret Taylor. Karena deret Taylor merupakan deret yang tak-berhingga, maka untuk penghampiran tersebut deret Taylor kita hentikan/potong sampai suku orde tertentu saja. Penghentian suatu deret atau runtunan langkah-langkah komputasi yang tidak berhingga menjadi runtunan langkah yang berhingga itulah yang menimbulkan galat pemotongan.

Contohnya, hampiran fungsi  $\cos(x)$  dengan bantuan deret Taylor di sekitar  $x = 0$ :

$$\cos(x) \approx 1 - \underbrace{\frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}}_{\text{nilai hampiran}} \left| \underbrace{\frac{x^8}{8!} - \frac{x^{10}}{10!} + \dots}_{\text{galat pemotongan}} \right.$$

pemotongan

Deret Taylor fungsi  $\cos(x)$  sebenarnya tidak berhingga, namun untuk keperluan praktis, deret tersebut kita potong sampai suku orde tertentu, misalnya sampai suku orde  $n = 6$  seperti pada contoh di atas. Kita melihat bahwa menghampiri  $\cos(x)$  dengan deret Taylor sampai suku berderajat enam tidak memberikan hasil yang tepat. Galat pada nilai hampiran diakibatkan oleh pemotongan suku-suku deret. Jumlah suku-suku selanjutnya setelah pemotongan merupakan galat pemotongan untuk  $\cos(x)$ . Kita tidak dapat menghitung berapa persisnya galat pemotongan ini karena jumlah seluruh suku-suku setelah pemotongan tidak

mungkin dapat dihitung. Namun, kita dapat menghampiri galat pemotongan ini dengan rumus suku sisa:

$$R_n(x) = \frac{(x - x_0)^{(n+1)}}{(n+1)!} f^{(n+1)}(c) \quad , \quad x_0 < c < x$$

Pada contoh  $\cos(x)$  di atas,

$$R_6(x) = \frac{x^7}{7!} \cos(c) \quad , \quad 0 < c < x$$

Nilai  $R_n$  yang tepat hampir tidak pernah dapat kita peroleh, karena kita tidak mengetahui nilai  $c$  sebenarnya terkecuali informasi bahwa  $c$  terletak pada suatu selang tertentu. Karenanya tugas kita adalah mencari nilai maksimum yang mungkin dari  $|R_n|$  untuk  $c$  dalam selang yang diberikan itu [PUR84], yaitu:

$$|R_n(x)| < \text{Max}_{x_0 < c < x} |f^{(n+1)}(c)| \times \frac{(x - x_0)^{n+1}}{n+1!}$$

Contoh komputasi lain yang menghasilkan galat pemotongan adalah perhitungan dengan menggunakan skema lelaran (lihat Contoh 2.5). Sayangnya, tidak seperti deret Taylor, galat pemotongan pada perhitungan dengan skema lelaran tidak ada rumusnya.

Galat pemotongan pada deret Taylor dapat dikurangi dengan meningkatkan orde suku-sukunya, namun jumlah komputasinya menjadi lebih banyak. Pada metode yang menerapkan skema lelaran, galat pemotongan dapat dikurangi dengan memperbanyak lelaran. Hal ini ditunjukkan pada Contoh 2.5 dengan memberikan nilai  $e_s$  yang sekecil mungkin

### Contoh 2.6

[PUR89] Gunakan deret Taylor orde 4 di sekitar  $x_0 = 1$  untuk menghampiri  $\ln(0.9)$  dan berikan taksiran untuk galat pemotongan maksimum yang dibuat.

#### Penyelesaian:

Tentukan turunan fungsi  $f(x) = \ln(x)$  terlebih dahulu

$$\begin{aligned} f(x) &= \ln(x) && \rightarrow f(1)=0 \\ f'(x) &= 1/x && \rightarrow f'(1)=1 \\ f''(x) &= -1/x^2 && \rightarrow f''(1)=-1 \\ f'''(x) &= 2/x^3 && \rightarrow f'''(1)=2 \end{aligned}$$

$$\begin{aligned} f^{(4)}(x) &= -6/x^4 & \rightarrow f^{(4)}(1) &= -6 \\ f^{(5)}(x) &= 24/x^5 & \rightarrow f^{(5)}(c) &= 24/c^5 \end{aligned}$$

Deret Taylornya adalah

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + R_4(x)$$

dan

$$\ln(0.9) = -0.1 - (-0.1)^2/2 + (-0.1)^3/3 - (-0.1)^4/4 + R_4(x) = -0.1053583 + R_4(x)$$

juga

$$|R_5(0.9)| < \max_{0.9 < c < 1} \left| \frac{24}{c^5} \right| \times \frac{(-0.1)^5}{5!}$$

dan nilai  $\max |24/c^5|$  di dalam selang  $0.9 < c < 1$  adalah pada  $c = 0.9$  (dengan mendasari pada fakta bahwa suatu pecahan nilainya semakin membesar bilamana penyebut dibuat lebih kecil), sehingga

$$|R_4(0.9)| < \max_{0.9 < c < 1} \left| \frac{24}{0.9^5} \right| \times \frac{(-0.1)^5}{5!} \approx 0.0000034$$

Jadi  $\ln(0.9) = -0.1053583$  dengan galat pemotongan lebih kecil dari 0.0000034. ■

### Contoh 2.7

Deret Taylor dapat digunakan untuk menghitung integral fungsi yang sulit diintegrasikan secara analitik (bahkan, adakalanya tidak mungkin dihitung secara analitik). Hitunglah hampiran nilai  $\int_0^1 e^{x^2} dx$  secara numerik, yaitu fungsi  $f(x) = e^{x^2}$  dihampiri dengan deret Maclaurin orde 8.

#### Penyelesaian:

Deret Maclaurin orde 8 dari fungsi  $f(x) = e^{x^2}$  adalah

$$e^{x^2} = 1 + x^2 + x^4/2! + x^6/3! + x^8/4! \quad (\text{silakan memeriksanya kembali sebagai latihan})$$

Dengan demikian, maka

$$\begin{aligned} \int_0^1 e^{x^2} dx &\approx \int_0^1 \left( 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \frac{x^8}{4!} \right) dx = x + \frac{x^3}{3} + \frac{x^5}{10} + \frac{x^7}{42} + \frac{x^9}{216} \Big|_{x=0}^{x=1} \\ &= 1 + \frac{1}{3} + \frac{1}{10} + \frac{1}{42} + \frac{1}{216} = \frac{397836}{272160} = 1.4617724 \end{aligned}$$

■

### 2.3.2 Galat Pembulatan

Perhitungan dengan metode numerik hampir selalu menggunakan bilangan riil. Masalah timbul bila komputasi numerik dikerjakan oleh mesin (dalam hal ini komputer) karena semua bilangan riil tidak dapat disajikan secara tepat di dalam komputer. Keterbatasan komputer dalam menyajikan bilangan riil menghasilkan galat yang disebut **galat pembulatan**. Sebagai contoh  $1/6 = 0.166666666\dots$  tidak dapat dinyatakan secara tepat oleh komputer karena digit 6 panjangnya tidak terbatas. Komputer hanya mampu merepresentasikan sejumlah digit (atau bit dalam sistem biner) saja. Bilangan riil yang panjangnya melebihi jumlah digit (bit) yang dapat direpresentasikan oleh komputer dibulatkan ke bilangan terdekat.

Misalnya sebuah komputer hanya dapat merepresentasikan bilangan riil dalam 6 digit angka berarti, maka representasi bilangan  $1/6 = 0.166666666\dots$  di dalam komputer 6-digit tersebut adalah 0.166667. Galat pembulatannya adalah  $1/6 - 0.166667 = -0.000000333$ . Contoh dalam sistem biner misalnya  $1/10 = 0.0001100110011001100110011\dots_2$  direpresentasikan di dalam komputer dalam jumlah bit yang terbatas. Teknik yang digunakan untuk pembulatan bilangan riil dijelaskan di dalam upabab 2.5.3.

Kebanyakan komputer digital mempunyai dua buah cara penyajian bilangan riil, yaitu **bilangan titik-tetap** (*fixed point*) dan bilangan **titik-kambang** (*floating point*) [KRE88]. Dalam format bilangan titik-tetap setiap bilangan disajikan dengan jumlah tempat desimal yang tetap, misalnya 62.358, 0.013, 1.000. Sedangkan dalam format bilangan titik-kambang setiap bilangan disajikan dengan jumlah digit *berarti* yang sudah tetap, misalnya

$$0.6238 \times 10^3 \qquad 0.1714 \times 10^{-13}$$

atau ditulis juga

$$0.6238\text{E}+03 \qquad 0.1714\text{E}-13$$

Digit-digit berarti di dalam format bilangan titik-kambang disebut juga **angka bena** (*significant figure*). Konsep angka bena dijelaskan berikut ini.

#### Angka Bena

Konsep angka bena (*significant figure*) atau angka berarti telah dikembangkan secara formal untuk menandakan keandalan suatu nilai numerik. Angka bena adalah angka bermakna, angka penting, atau angka yang dapat digunakan dengan pasti [CHA91].

Contohnya,

43.123	memiliki 5 angka bena (yaitu 4, 3, 1, 2, 3)
0.1764	memiliki 4 angka bena (yaitu 1, 7, 6, 4)
0.0000012	memiliki 2 angka bena (yaitu 1, 2)
278.300	memiliki 6 angka bena (yaitu 2, 7, 8, 3, 0, 0)
270.0090	memiliki 7 angka bena (yaitu 2, 7, 0, 0, 0, 9, 0)
0.0090	memiliki 2 angka bena (yaitu 9, 0)
1360, 1.360, 0.001360 semuanya memiliki 4 angka bena	

Perhatikanlah bahwa angka 0 bisa menjadi angka bena atau bukan. Pada contoh 0.001360, tiga buah angka nol pertama tidak berarti, sedangkan 0 yang terakhir angka berarti karena pengukuran dilakukan sampai ketelitian 4 digit. Jumlah angka bena akan terlihat dengan pasti bila bilangan riil itu ditulis dalam penulisan ilmiah (*scientific notation*), misalnya tetapan dalam kimia dan fisika atau ukuran jarak dalam astronomi. Jumlah angka bena terletak pada jumlah digit *mantis*-nya (tentang mantis ini akan dibahas belakangan):

$4.3123 \times 10^1$	memiliki 5 angka bena
$1.764 \times 10^{-1}$	memiliki 4 angka bena
$1.2 \times 10^{-6}$	memiliki 2 angka bena
$2.78300 \times 10^2$	memiliki 6 angka bena
$0.2700090 \times 10^3$	memiliki 7 angka bena
$9.0 \times 10^{-3}$	memiliki 2 angka bena
$13.60 \times 10^2$ , $0.1360 \times 10^1$ , $1.360 \times 10^{-3}$	memiliki 4 angka bena
$6.02 \times 10^{23}$	memiliki 24 angka bena (bilangan <i>Avogadro</i> )
$1.5 \times 10^7$	memiliki 8 angka bena (jarak bumi-matahari)

Komputer hanya menyimpan sejumlah tertentu angka bena. Bilangan riil yang jumlah angka benanya melebihi jumlah angka bena komputer akan disimpan dalam sejumlah angka bena komputer itu. Pengabaian angka bena sisanya itulah yang menimbulkan galat pembulatan.

### 2.3.3 Galat Total

Galat akhir atau galat total atau pada solusi numerik merupakan jumlah galat pemotongan dan galat pembulatan. Misalnya pada Contoh 2.3 kita menggunakan deret Maclaurin orde-4 untuk menghampiri  $\cos(0.2)$  sebagai berikut:

$$\cos(0.2) \approx 1 - 0.2^2/2 + 0.2^4/24 \approx 0.9800667$$

$\uparrow$   
galat  
pemotongan

$\uparrow$   
galat  
pembulatan

Galat pemotongan timbul karena kita menghampiri  $\cos(0.2)$  sampai suku orde empat, sedangkan galat pembulatan timbul karena kita membulatkan nilai hampiran ke dalam 7 digit bena.

## 2.4 Orde Penghampiran

Di dalam metode numerik, fungsi  $f(x)$  sering diganti dengan fungsi hampiran yang lebih sederhana. Satu cara mengungkapkan tingkat ketelitian penghampiran itu adalah dengan menggunakan notasi **O-Besar** (*Big-Oh*).

Misalkan  $f(h)$  dihampiri dengan fungsi  $p(h)$ . Jika  $|f(h) - p(h)| \leq M|h^n|$ , yang dalam hal ini  $M$  adalah konstanta riil  $> 0$ , maka kita katakan bahwa  $p(h)$  menghampiri  $f(h)$  dengan orde penghampiran  $O(h^n)$  dan kita tulis

$$f(h) = p(h) + O(h^n) \quad (\text{P.2.14})$$

$O(h^n)$  juga dapat diartikan sebagai orde galat dari penghampiran fungsi. Karena  $h$  umumnya cukup kecil yaitu lebih kurang dari 1, maka semakin tinggi nilai  $n$ , semakin kecil galat, yang berarti semakin teliti penghampiran fungsinya. Metode yang berorde  $O(h^2)$ , misalnya, lebih teliti hasilnya daripada metode yang berorde  $O(h)$ . Juga, pada metode yang berorde  $O(h^2)$ , jika ukuran  $h$  dijadikan setengah kali semula, maka galatnya menjadi seperempat kali galat semula.

Umumnya deret Taylor digunakan untuk penghampiran fungsi. Misalkan,

$$x_{i+1} = x_i + h, \quad i = 0, 1, 2, \dots$$

adalah titik-titik selebar  $h$ , maka hampiran  $f(x_{i+1})$  dengan deret Taylor di sekitar  $x_i$  adalah

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

yang dalam hal ini

$$R_n(x_{i+1}) = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(t) = O(h^{n+1}) \quad , \quad x_i < t < x_{i+1} \quad (\text{P.2.15})$$

Jadi, kita dapat menuliskan

$$f(x_{i+1}) = \sum_{k=0}^n \frac{h^k}{k!} f^{(k)}(x_i) + O(h^{n+1}) \quad (\text{P.2.16})$$

Persamaan (P.2.16) menyatakan bahwa jika fungsi  $f(x)$  dihampiri dengan deret Taylor derajat  $n$ , maka suku sisanya cukup dinyatakan dengan lambang  $O(h^{n+1})$ . Sebagai catatan, suku sisa yang digunakan di dalam notasi  $O$ -Besar adalah suku yang dimulai dengan perpangkatan  $h^{n+1}$ .

Sebagai contoh,

$$e^h = 1 + h + h^2/2! + h^3/3! + h^4/4! + O(h^5)$$

$$\ln(x+1) = x - x^2/2 + x^3/3 - x^4/4 + x^5/5 + O(h^5)$$

$$\sin(h) = h - h^3/3! + h^5/5! + O(h^7) \quad (\text{bukan } O(h^6), \text{ karena suku orde } 6 = 0)$$

$$\cos(h) = 1 - h^2/2! + h^4/4! - h^6/6! + O(h^8) \quad (\text{bukan } O(h^7), \text{ karena suku orde } 7 = 0)$$

## 2.5 Bilangan Titik-Kambang

Untuk memahami galat pembulatan lebih rinci, kita perlu mengerti cara penyimpanan bilangan riil di dalam komputer. Format bilangan riil di dalam komputer berbeda-beda bergantung pada piranti keras dan compiler bahasa pemrogramannya. Bilangan riil di dalam komputer umumnya disajikan dalam format *bilangan titik-kambang*. Bilangan titik-kambang  $a$  ditulis sebagai

$$a = \pm m \times B^p = \pm 0.d_1d_2d_3d_4d_5d_6 \dots d_n \times B^p \quad (\text{P.2.17})$$

yang dalam hal ini,

$m$  = mantisa (riil),  $d_1d_2d_3d_4d_5d_6 \dots d_n$  adalah digit atau bit mantisa yang nilainya dari 0 sampai  $B - 1$ ,  $n$  adalah panjang digit (bit) mantisa.

$B$  = basis sistem bilangan yang dipakai (2, 8, 10, 16, dan sebagainya)

$p$  = pangkat (berupa bilangan bulat), nilainya dari  $-P_{\min}$  sampai  $+P_{\max}$

Sebagai contoh, bilangan riil 245.7654 dinyatakan sebagai  $0.2457654 \times 10^3$  dalam format bilangan titik kambang dengan basis 10. Cara penyajian seperti itu serupa dengan cara penulisan ilmiah. Penulisan ilmiah termasuk ke dalam sistem bilangan titik-kambang.

Sistem bilangan yang kita gunakan setiap hari menggunakan basis sepuluh (disebut juga sistem desimal),  $B = 10$ . Umumnya komputer menggunakan sistem biner ( $B = 2$ ), tapi beberapa komputer menggunakan basis 8 dan 16. Untuk memudahkan pemahaman –juga karena kita lebih terbiasa sehari-hari dengan bilangan desimal– kebanyakan contoh-contoh bilangan titik-kambang di dalam bab ini disajikan dalam sistem desimal.

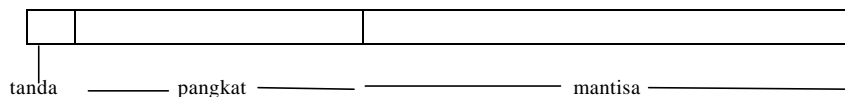
Bilangan titik-kambang di dalam sistem biner biner direpresentasikan oleh komputer dalam bentuk *word* seperti ditunjukkan pada Gambar 2.2. Bit pertama menyatakan tanda (+/-), deretan bit berikutnya menyatakan pangkat bertanda, dan deretan bit terakhir untuk mantisa.

Setiap komputer memiliki panjang *word* yang berbeda-beda. Pada komputer IBM PC, bilangan titik-kambang berketelitian tunggal (*single precission*) disajikan dalam 32 bit yang terdiri atas 1 bit sebagai tanda, 8 bit untuk pangkat dan 23 bit untuk mantisa. Jika dalam bentuk ternormalisasi (akan dijelaskan kemudian), maka bit pertama pada mantisa harus 1, sehingga jumlah bit mantisa efektif adalah 24:

$$a = \pm 0.1b_1b_2b_3b_4b_5b_6 \dots b_{23} \times B^p$$

yang dalam hal ini  $b$  menyatakan bit biner (0 atau 1).

Sedangkan pada komputer IBM 370, bilangan titik-kambang berketelitian tunggal disajikan dalam 32 bit yang terdiri dari 1 bit tanda, 7 bit pangkat (basis 16), dan 24 bit mantis (setara dengan 6 sampai 7 digit desimal) [KRE88].



**Gambar 2.2** Format bilangan titik-kambang biner (1 *word*)



## 2.5.1 Bilangan Titik-Kambang Ternormalisasi

Representasi bilangan titik-kambang (P.2.17) jauh dari unik, karena, sebagai contoh, kita juga dapat menuliskannya sebagai

$$a = \pm (mb) \times B^{p-1} \quad (\text{P.2.18})$$

Misalnya, 245.7654 dapat ditulis sebagai

$$\begin{aligned} &0.2457654 \times 10^3 \text{ atau} \\ &2.457654 \times 10^2 \text{ atau} \\ &0.02457654 \times 10^4, \text{ dan sebagainya} \end{aligned}$$

Agar bilangan titik-kambang dapat disajikan secara seragam, kebanyakan sistem komputer menormalisasikan formatnya sehingga semua digit mantisa selalu angka bena. Karena alasan itu, maka digit pertama mantisa tidak boleh nol. Bilangan titik-kambang yang dinormalisasi ditulis sebagai

$$a = \pm m \times B^p = \pm 0.d_1d_2d_3d_4d_5d_6 \dots d_n \times B^p \quad (\text{P.2.19})$$

yang dalam hal ini,  $d_1d_2d_3d_4d_5d_6 \dots d_n$  adalah digit (atau bit) mantisa dengan syarat  $1 \leq d_1 \leq b-1$  dan  $0 \leq d_k \leq B-1$  untuk  $k > 1$ . Pada sistem desimal,

$$1 \leq d_1 \leq 9 \text{ dan } 0 \leq d_k \leq 9,$$

sedangkan pada sistem biner,

$$d_1 = 1 \text{ dan } 0 \leq d_k \leq 1$$

Sebagai contoh,  $0.0563 \times 10^{-3}$  dinormalisasi menjadi  $0.563 \times 10^{-4}$ ,  $0.00023270 \times 10^6$  dinormalisasi menjadi  $0.23270 \times 10^3$ . Sebagai konsekuensi penormalan, nilai  $m$  adalah

$$1/B \leq m < 1$$

Pada sistem desimal ( $B = 10$ ),  $m$  akan berkisar dari 0.1 sampai 1, dan pada sistem biner ( $B = 2$ ), antara 0.5 dan 1 [CHA91].

Sebagai catatan, nol adalah kasus khusus. Nol disajikan dengan bagian mantisa seluruhnya nol dan pangkatnya nol. Nol semacam ini tidak dinormalisasi.

### Contoh 2.8

[BUC92] Tulislah bilangan  $e$  dalam format bilangan titik-kambang ternormalisasi dengan basis 10, basis 2, dan basis 16.

#### Penyelesaian:

Dalam basis 10 (menggunakan 8 angka bena),

$$e \approx 2.7182818 = 0.27182818 \times 10^1$$

(bilangan titik-kambang desimal ternormalisasi)

Dalam basis 2 (menggunakan 30 bit bena),

$$e \approx 0.101011011111100001010100010110_2 \times 2^2$$

(bilangan titik-kambang biner ternormalisasi)

Dalam basis 16 (gunakan fakta bahwa  $16 = 2^4$ , sehingga  $2^2 = \frac{1}{4} \times 16^1$ )

$$\begin{aligned} e &\approx 0.101011011111100001010100010110_2 \times 2^2 \\ &= \frac{1}{4} \times 0.101011011111100001010100010110_2 \times 16^1 \\ &= 0.00101011011111100001010100010110_2 \times 16^1 \\ &= 0.2B7E1516_{16} \times 16^1 \end{aligned}$$

(bilangan titik-kambang heksadesimal ternormalisasi) ■

## 2.5.2 Epsilon Mesin

Karena jumlah bit yang digunakan untuk representasi bilangan titik-kambang terbatas, maka jumlah bilangan riil yang dapat direpresentasikan juga terbatas. Untuk ilustrasi, tinjau kasus bilangan titik-kambang biner 6-bit *word* (1 bit tanda, 3 bit untuk pangkat bertanda, dan 2 bit mantisa) dengan  $B = 2$ , dan nilai pangkat dari  $-2$  sampai 3 [GER94]. Karena semua bilangan dinormalisasi, maka bit pertama harus 1, sehingga semua bilangan yang mungkin adalah berbentuk:

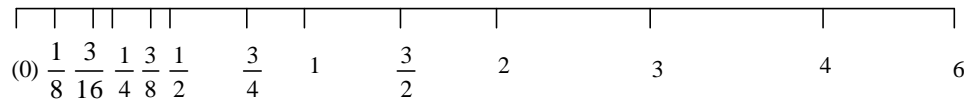
$$\pm 0.10_2 \times 2^p \quad \text{atau} \quad \pm 0.11_2 \times 2^p, \quad -2 \leq p \leq 3$$

Daftar bilangan riil positif yang dapat direpresentasikan adalah

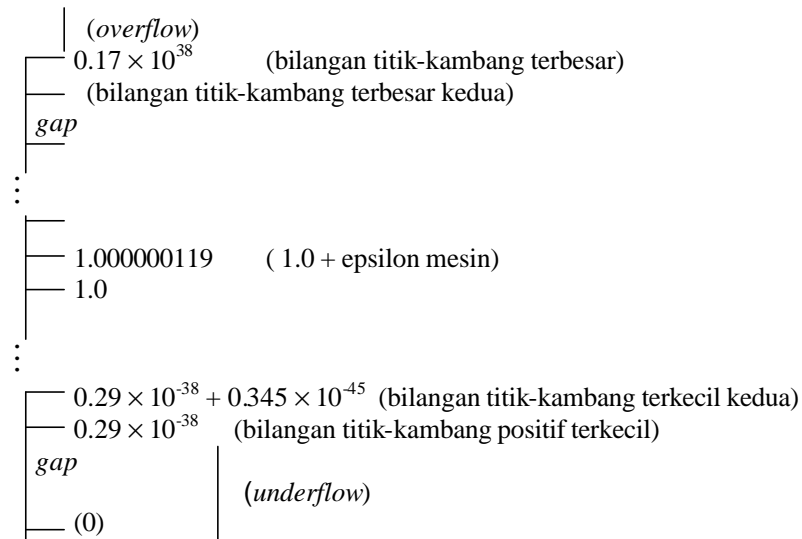
$$\begin{aligned} 0.10_2 \times 2^{-2} &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{-2} = 1/8 = 0.125_{10} \\ 0.10_2 \times 2^{-1} &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{-1} = 1/4 = 0.25_{10} \\ 0.10_2 \times 2^0 &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^0 = 1/2 = 0.5_{10} \\ 0.10_2 \times 2^1 &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^1 = 1 = 1.0_{10} \end{aligned}$$

$$\begin{aligned}
0.10_2 \times 2^2 &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^2 = 2 = 2.0_{10} \\
0.10_2 \times 2^3 &= (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^3 = 4 = 4.0_{10} \\
0.11_2 \times 2^{-2} &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{-2} = 3/16 = 0.1875_{10} \\
0.11_2 \times 2^{-1} &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{-1} = 3/8 = 0.375_{10} \\
0.11_2 \times 2^0 &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^0 = 3/4 = 0.75_{10} \\
0.11_2 \times 2^1 &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^1 = 3/2 = 1.5_{10} \\
0.11_2 \times 2^2 &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^2 = 3 = 3.0_{10} \\
0.11_2 \times 2^3 &= (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^3 = 6 = 6.0_{10}
\end{aligned}$$

Bila kita susun dari nilai positif terkecil ke nilai terbesar, maka seluruh bilangannya digambarkan dalam diagram garis bilangan sebagai berikut:



Pada komputer IBM PC, bilangan titik-kambang berketelitian-tunggal dinyatakan dalam 32-bit *word* (1 bit tanda, 8 bit pangkat, dan 24 bit mantisa). Rentang nilai-nilai positifnya diperlihatkan pada Gambar 2.3.



**Gambar 2.3** Rentang bilangan titik-kambang berketelitian tunggal pada *interpreter* Basic

Satu ukuran yang penting di dalam aritmetika komputer adalah seberapa kecil perbedaan antara dua buah nilai yang dapat dikenali oleh komputer. Ukuran yang digunakan untuk membedakan suatu bilangan riil dengan bilangan riil berikutnya adalah **epsilon mesin**. Epsilon mesin distandardisasi dengan menemukan bilangan titik-kambang terkecil yang bila ditambahkan dengan 1 memberikan hasil yang lebih besar dari 1. Dengan kata lain, jika epsilon mesin dilambangkan dengan **e** maka

$$1 + e > 1$$

(bilangan yang lebih kecil dari epsilon mesin didefinisikan sebagai nol di dalam komputer).

Epsilon mesin pada sistem bilangan riil yang ditunjukkan pada Gambar 2.3 adalah

$$e = 1.000000119 - 1.0 = 0.119 \times 10^{-6}$$

*Gap* ( $\Delta x$ ) atau jarak antara sebuah bilangan titik-kambang dengan bilangan titik-kambang berikutnya, yang besarnya adalah

$$\Delta x = e \times R \quad (\text{P.2.20})$$

yang dalam hal ini  $R$  adalah bilangan titik-kambang sekarang. Contohnya, *gap* antara bilangan positif terkecil pertama  $0.29 \times 10^{-38}$  dengan bilangan titik-kambang terkecil kedua pada Gambar 2.3 adalah

$$\Delta x = (0.119 \times 10^{-6}) \times (0.29 \times 10^{-38}) = 0.345 \times 10^{-45}$$

dan dengan demikian bilangan titik-kambang terkecil kedua sesudah  $0.29 \times 10^{-38}$  adalah

$$0.29 \times 10^{-38} + 0.345 \times 10^{-45}$$

Dari persamaan P.2.16 dapat dilihat bahwa *gap* akan bertambah besar dengan semakin besarnya bilangan titik-kambang.

Keadaan *underflow* terjadi bila suatu bilangan titik-kambang tidak dapat dinyatakan di antara 0 dan bilangan positif terkecil (atau antara 0 dan bilangan negatif terbesar). Keadaan *overflow* terjadi bila suatu bilangan titik-kambang lebih besar dari bilangan positif terbesar (atau lebih kecil dari bilangan negatif terkecil)

Jika kita mengetahui jumlah bit mantisa dari suatu bilangan titik-kambang, kita dapat menghitung epsilon mesinnya dengan rumus

$$e = B^{1-n} \quad (\text{P.2.21})$$

yang dalam hal ini  $B$  adalah basis bilangan dan  $n$  adalah banyaknya digit (atau bit) bena di dalam mantisa. Pada contoh pertama di atas ( $B = 2$  dan  $n = 2$ ),

$$e = 2^{1-2} = 0.5$$

dan pada contoh bilangan titik-kambang berketelitian tunggal pada komputer IBM PC,

$$e = 2^{1-24} = 0.00000011920928955078125 = 0.119 \times 10^{-6}$$

Kita juga dapat menemukan perkiraan nilai epsilon mesin dengan prosedur yang sederhana. Gagasannya ialah dengan membagi dua secara terus menerus nilai 1 dan memeriksa apakah 1 ditambah hasil bagi itu lebih besar dari 1. Potongan programnya dituliskan di dalam Program 2.2.

**Program 2.2** Program menghitung hampiran nilai epsilon mesin

```

procedure HitungEpsilonMesin1(var eps : real);
{ Prosedur untuk menemukan epsilon mesin
  Keadaan Awal : sembarang
  Keadaan Akhir: eps berisi harga epsilon mesin
}
begin
  eps:=1;
  while eps + 1 > 1 do
    eps:=eps/2;
    {eps + 1 < 1}

  eps:=2*eps;      {nilai epsilon mesin}
end;

```

Hasil pelaksanaan program dengan *compiler* Turbo Pascal dan komputer dengan *processor* 486DX adalah

$$e = 0.90949470177 \times 10^{-12}$$

Jika yang diinginkan adalah epsilon mesin dalam bentuk perpangkatan dari 2, prosedur untuk menghitungnya dituliskan di dalam Program 2.3.

**Program 2.3** Program menghitung hampiran nilai epsilon mesin dalam bentuk  $2^k$

```
procedure HitungEpsilon_Mesin2(var n : integer);
{ Prosedur untuk menemukan epsilon mesin dalam bentuk perpangkatan 2
  Keadaan Awal : sembarang
  Keadaan Akhir :  $2^{-(n-1)}$  adalah epsilon mesin
}
var
  eps, ne : real;
begin
  eps:=1; ne:=2; n:=0;
  while ne > 1 do
    begin
      eps:=eps/2;
      ne:=1 + eps;
      n:=n + 1;
    end;
  {eps + 1 < 1}
  { Epsilon mesin ialah  $2^{-(n-1)}$  }
end;
```

Hasil pelaksanaan program dengan *compiler* Turbo Pascal dan komputer dengan *processor* 486 adalah

$$e = 2^{-40} \quad (\text{Keterangan: } 2^{-40} = 0.90949470177 \times 10^{-12})$$

Nilai epsilon mesin yang diperoleh dapat berbeda-beda bergantung pada bahasa pemrograman dan komputer yang digunakan karena beberapa bahasa menggunakan bilangan berketelitian ganda (*double precision*) untuk representasi bilangan titik-kambangnya.

Epsilon dapat digunakan sebagai kriteria berhenti kekonvergenan pada prosedur lelaran yang konvergen. Nilai lelaran sekarang dibandingkan dengan nilai lelaran sebelumnya. Jika selisih keduanya sudah kecil dari epsilon mesin, lelaran dihentikan, tetapi jika tidak, lelaran diteruskan.

### 2.5.3 Pembulatan pada Bilangan Titik-Kambang

Dari ilustrasi pada upabab 2.5.2 jelaslah bahwa jumlah bilangan riil yang dapat dinyatakan sebagai bilangan titik-kambang terbatas banyaknya (bergantung pada banyaknya bit *word*). Bilangan titik-kambang yang tidak dapat mencocoki satu dari nilai-nilai di dalam rentang pasti terletak di dalam *gap*. Karena itu, bilangan tersebut dibulatkan (atau dikuantisasi) ke salah satu nilai di dalam rentang. Galat yang timbul akibat penghampiran tersebut diacu sebagai galat pembulatan (atau galat kuantisasi). Misalnya, 0.3748 dibulatkan ke 0.375, 3.2 dibulatkan ke 3.0, dan sebagainya.

Ada dua teknik pembulatan yang lazim dipakai oleh komputer, yaitu **pemenggalan** (*chopping*) dan **pembulatan ke digit terdekat** (*in-rounding*). Kedua teknik pembulatan tersebut diilustrasikan di bawah ini.

### 1. Pemenggalan (*chopping*)

Misalkan  $a$  adalah bilangan titik-kambang dalam basis 10:

$$a = \pm 0.d_1d_2d_3 \dots d_nd_{n+1} \dots \times 10^p$$

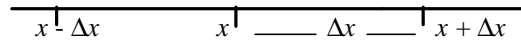
Misalkan  $n$  adalah banyak digit mantis komputer. Karena digit mantis  $a$  lebih banyak dari digit mantis komputer, maka bilangan  $a$  dipotong sampai  $n$  digit saja:

$$fl_{\text{chop}}(a) = \pm 0.d_1d_2d_3 \dots d_nd_n \times 10^p \quad (\text{P.2.21})$$

Sebagai contoh, bilangan  $p = 0.31459265358 \dots \times 10^0$  di dalam komputer dengan 7 digit mantis disimpan sebagai

$$fl_{\text{chop}}(p) = 0.3141592 \times 10^0 \text{ dengan galat sebesar } 0.00000065 \dots$$

Perhatikan juga bahwa pemenggalan berarti sembarang besaran yang berada pada *gap* sebesar  $\Delta x$  akan disimpan sebagai besaran pada ujung selang yang lebih kecil, sehingga batas atas galat untuk pemenggalan adalah  $\Delta x$  (Gambar 2.4) [CHA91].



**Gambar 2.4** Batas atas pemenggalan. Bilangan yang akan dipenggal pada mulanya terletak antara  $x$  dan  $x + \Delta x$ . Setelah pemenggalan, bilangan tersebut =  $x$ .

Contoh pemenggalan pada bilangan titik-kambang biner misalnya,

$$1/10 = 0.0001100110011001100110011 \dots_2$$

Bila digunakan mantis 32 bit, komputer memenggal bilangan  $1/10$  di atas menjadi (setelah dinormalkan)

$$1/10 \approx 0.1100110011001100110011001100_2 \times 2^{-3}$$

## 2. Pembulatan ke digit terdekat (*in-rounding*)

Misalkan  $a$  adalah bilangan titik-kambang dalam basis 10:

$$a = \pm 0.d_1 d_2 d_3 \dots d_n d_{n+1} \dots \times 10^p$$

Misalkan  $n$  adalah jumlah digit mantis komputer. Karena digit mantis  $a$  lebih banyak dari digit mantis komputer, maka bilangan  $a$  dibulatkan sampai  $n$  digit:

$$fl_{\text{round}}(a) = \pm 0.d_1 d_2 d_3 \dots \hat{d}_n \times 10^p \quad (\text{P.2.22})$$

yang dalam hal ini

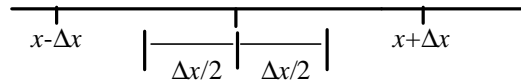
$$\hat{d}_n = \begin{cases} d_n & , \text{jika } d_{n+1} < 5 \\ d_n + 1 & , \text{jika } d_{n+1} > 5 \\ d_n & , \text{jika } d_{n+1} = 5 \text{ dan } n \text{ genap} \\ d_n + 1 & , \text{jika } d_{n+1} = 5 \text{ dan } n \text{ ganjil} \end{cases} \quad (\text{P.2.23})$$

Contohnya, bilangan  $p = 0.31459265358 \dots \times 10^0$  di dalam komputer hipotetis dengan 7 digit mantis dibulatkan menjadi  $fl(\pi) = 0.3141593 \times 10^0$  dengan galat sebesar 0.00000035.... Contoh ini memperlihatkan bahwa pembulatan ke digit terdekat menghasilkan galat yang lebih rendah daripada pemenggalan.

Contoh lainnya, nilai  $a = 0.5682785715287 \times 10^{-4}$  :

- di dalam komputer 7 digit dibulatkan menjadi  $fl_{\text{round}}(a) = 0.5682786 \times 10^{-4}$
- di dalam komputer 8 digit dibulatkan menjadi  $fl_{\text{round}}(a) = 0.56827857 \times 10^{-4}$
- di dalam komputer 6 digit dibulatkan menjadi  $fl_{\text{round}}(a) = 0.568278 \times 10^{-4}$
- di dalam komputer 9 digit dibulatkan menjadi  $fl_{\text{round}}(a) = 0.568278572 \times 10^{-4}$

Perhatikan juga bahwa pembulatan ke digit terdekat berarti sembarang besaran yang berada pada *gap* sebesar  $\Delta x$  akan disimpan sebagai bilangan terdekat yang diperbolehkan, sehingga batas atas galat untuk pembulatan adalah  $\Delta x/2$  (Gambar 2.5):



**Gambar 2.5** Batas atas pembulatan ke digit terdekat



Contoh pembulatan  $p$  yang diberikan di atas memperlihatkan bahwa galat pembulatan ke digit terdekat lebih rendah daripada galat pemenggalan, karena itu cara pemenggalan biasanya tidak direkomendasikan untuk dipakai. Namun yang mengherankan, kebanyakan komputer menggunakan cara pemenggalan! Alasannya adalah bahwa algoritma pembulatan ke digit terdekat lebih sukar sehingga membutuhkan waktu lebih lama dari pada waktu untuk pemenggalan. Sedangkan algoritma pemenggalan lebih sederhana sehingga mudah direalisasikan. Pendekatan ini dapat diterima dengan anggapan bahwa jumlah angka bena cukup besar sehingga galat pemenggalan yang dihasilkan biasanya dapat diabaikan [CHA91]. Algoritma pembulatan/pemenggalan dapat diimplementasikan baik di dalam piranti keras atau di dalam rutin perangkat lunak.

Secara umum dapat dinyatakan bahwa satu-satunya cara meminimumkan galat pembulatan adalah menggunakan jumlah angka bena yang lebih banyak. Di dalam program komputer itu artinya kita menggunakan bilangan riil berketelitian ganda (*double precision*) ketimbang bilangan berketelitian tunggal (*single precision*).

## 2.5.4 Aritmetika Bilangan Titik-Kambang

Selain mengandung galat pembulatan pada representasinya, operasi aritmetika pada bilangan titik-kambang juga menghasilkan galat pembulatan yang lain. Operasi aritmetika pada bilangan titik-kambang meliputi operasi penambahan dan pengurangan, operasi perkalian, dan operasi pembagian.

### 2.5.4.1 Operasi Penambahan dan Pengurangan

Terdapat dua buah kasus serius yang menyebabkan timbulnya galat pembulatan pada operasi penjumlahan dua buah bilangan titik-kambang:

Kasus 1: Penjumlahan (termasuk pengurangan) bilangan yang sangat kecil ke (atau dari) bilangan yang lebih besar menyebabkan timbulnya galat pembulatan.

Galat pembulatan pada Kasus 1 ini terjadi karena untuk menjumlahkan dua buah bilangan yang berbeda relatif besar, pangkatnya harus disamakan terlebih dahulu (disamakan dengan pangkat bilangan yang lebih besar). Caranya adalah dengan menggeser digit-digit (atau bit) bilangan yang pangkatnya lebih kecil. Pergeseran digit (atau bit) ini mengakibatkan adanya digit (atau bit) yang hilang. Perhatikan contoh berikut.

**Contoh 2.9**

Misalkan digunakan komputer dengan mantis 4 digit (basis 10). Hitunglah

$$1.557 + 0.04381 = 0.1557 \times 10^1 + 0.4381 \times 10^{-1}$$

**Penyelesaian:**

$$\begin{aligned} 0.1557 \times 10^1 &= 0.1557 \times 10^1 \\ 0.4381 \times 10^{-1} &= 0.004381 \times 10^1 + (\text{pergeseran digit untuk menyamakan pangkat}) \\ &= 0.160081 \times 10^1 \\ &\quad \text{in-rounding} \rightarrow 0.1601 \times 10^1 \\ &\quad \text{chopping} \rightarrow 0.1600 \times 10^1 \end{aligned}$$

Perhatikanlah bahwa dua digit terakhir dari bilangan yang digeser ke kanan pada dasarnya telah hilang dari perhitungan.

$$\text{Galat mutlak pembulatan} = |(0.160081 \times 10^1) - (0.1601 \times 10^1)| = 0.000019$$

$$\text{Galat mutlak pemenggalan} = |(0.160081 \times 10^1) - (0.1600 \times 10^1)| = 0.000081 \quad \blacksquare$$

Galat perhitungan semacam Kasus 1 ini dapat terjadi dalam perhitungan deret tak berhingga yang suku awalnya relatif lebih besar dibandingkan suku berikutnya. Jadi, setelah beberapa suku ditambahkan, kita berada dalam situasi penambahan besaran yang kecil terhadap besaran yang besar. Suatu cara mengurangi galat jenis ini adalah menjumlahkan deret dalam urutan terbalik -yakni dalam urutan yang menaik ketimbang menurun. Dengan cara ini, setiap suku baru akan sebanding besarnya dengan jumlah deret yang terakumulasi [CHA91].

**Contoh 2.10**

Misalkan digunakan komputer dengan mantis 4 digit (basis 10). Hitunglah

$$3677 - 0.3283 = 0.3677 \times 10^4 - 0.3283 \times 10^0$$

**Penyelesaian:**

$$\begin{aligned} 0.3677 \times 10^4 &= 0.3677 \times 10^4 \\ 0.3283 \times 10^0 &= 0.00003283 \times 10^4 - (\text{pergeseran digit untuk menyamakan pangkat}) \\ &= 0.36766717 \times 10^4 \\ &\quad \text{in-rounding} \rightarrow 0.3677 \times 10^4 \\ &\quad \text{chopping} \rightarrow 0.3676 \times 10^4 \end{aligned}$$

$$\text{Galat mutlak pembulatan} = |(0.36766717 \times 10^4) - (0.3677 \times 10^4)| = 0.00003283$$

$$\text{Galat mutlak pemenggalan} = |(0.36766717 \times 10^4) - (0.3676 \times 10^4)| = 0.00006717 \quad \blacksquare$$

### Contoh 2.11

[NAK93] Misalkan digunakan komputer dengan mantis 24 bit (setara dengan 7 tempat desimal). Nilai

$$x = 1.0 + \sum_{i=1}^{10000} 0.00001 = 1.0 + \underbrace{0.00001 + 0.00001 + \dots + 0.00001}_{10000 \text{ kali}}$$

akan dihitung dengan program Pascal yang menggunakan bilangan riil berketelitian-tunggal sebagai berikut:

```
program hitung;  
var  
  x, jumlah : real;  
  i: integer;  
begin  
  jumlah:=1.0;  
  for i:=1 to 10000 do  
    jumlah:=jumlah + 0.00001;  
  writeln('x=', jumlah);  
end.
```

Bila digunakan komputer dengan 7 angka bena, hasil program di atas adalah  $x = 1.100136$ , padahal seharusnya  $x = 1.0 + 10000(0.00001) = 1.0 + 0.1 = 1.1$ .

$$\text{Galat mutlak} = |1.1 - 1.100136| = 0.000136$$

$$\text{Galat relatif} = \frac{1.1 - 1.100136}{1.1} \times 100\% = -0.0124$$

Bagaimana kita menjelaskan hasil perhitungan ini? Sumber penyimpangan ini adalah galat pembulatan. Karena bilangan 1 dapat dinyatakan secara persis dalam komputer, maka galatnya nol. Sebaliknya 0.00001 tidak dapat dinyatakan secara tepat dan dikuantisasi oleh nilai yang sedikit berbeda dibandingkan nilai sejatinya. Penyimpangan yang kecil itu dapat diabaikan untuk komputasi yang sedikit, namun terakumulasi setelah penjumlahan yang berulang kali [CHA91]. Untuk jelasnya perhatikan proses perhitungannya dalam sistem biner di bawah ini.

$$\begin{array}{rcl} 1_{10} & = & 0.100000000000000000000000_2 \times 2^1 \\ (0.00001)_{10} & = & 0.101001111100010110101100_2 \times 2^{-16} \\ & = & 0.0000000000000000101001111100010110101100_2 \times 2^1 + \\ \hline (1 + 0.00001)_{10} & = & 0.10000000000000000101001111100010110101100_2 \times 2^1 \\ & & \text{dibulatkan ke mantis 24 bit} \\ (1.00001)_{10} & & \\ (\text{nilai sejati}) & & \\ & & 0.100000000000000001010100_2 \times 2 = (1.0000100136)_{10} \\ & & (\text{nilai hampiran}) \end{array}$$

$$\begin{aligned}\text{galat mutlak} &= |1.00001 - 1.0000100136| = 0.136 \times 10^{-7} \\ \text{galat total} &\approx 10000 \times 0.136 \cdot 10^{-7} = 0.136 \times 10^{-3} = 0.000136\end{aligned}$$

Satu cara untuk mengurangi galat total ini adalah menjumlahkan suku-suku dalam urutan terbalik, yaitu 0.00001 dijumlahkan terlebih dahulu sebanyak seribu kali, baru kemudian hasilnya dijumlahkan dengan 1.0. Jadi cara perhitungannya adalah

$$x = \left( \sum_{i=1}^{10000} 0.00001 \right) + 1.0 = \underbrace{0.00001 + 0.00001 + \dots + 0.00001}_{10000 \text{ kali}} + 1.0$$

Selain itu, gunakan bilangan berketelitian ganda, sebab galat pembulatannya jauh lebih kecil. ■

**Kasus 2:** Pengurangan dua buah bilangan yang hampir sama besar (*nearly equal*). Bila dua bilangan titik-kambang dikurangkan, hasilnya mungkin mengandung nol pada posisi digit mantis yang paling berarti (posisi digit paling kiri). Keadaan ini dinamakan **kehilangan angka bena** (*loss of significance*). Baik pemenggalan maupun pembulatan ke digit terdekat menghasilkan jawaban yang sama.

### Contoh 2.12

Kurangi  $0.56780 \times 10^5$  dengan  $0.56430 \times 10^5$  (5 angka bena)

**Penyelesaian:**

$$\begin{array}{r} 0.56780 \times 10^5 \\ 0.56430 \times 10^5 - \\ \hline 0.00350 \times 10^5 \end{array} \rightarrow \text{dinormalisasi menjadi } 0.350 \times 10^3 \text{ (3 angka bena)}$$

$$\begin{array}{ll} \text{in-rounding} & \rightarrow 0.350 \times 10^3 \\ \text{chopping} & \rightarrow 0.350 \times 10^3 \end{array}$$

Hasil yang diperoleh hanya mempunyai 3 angka bena. Jadi kita kehilangan 2 buah angka bena. Meskipun kita dapat menuliskan hasilnya sebagai  $0.35000 \times 10^3$ , namun dua nol yang terakhir bukan angka bena tetapi sengaja ditambahkan untuk mengisi kekosongan digit yang hilang. ■

Hasil yang lebih dramatis diperlihatkan pada Contoh 2.13 dan Contoh 2.14 di bawah ini.

**Contoh 2.13**

Kurangi  $3.1415926536$  dengan  $3.1415957341$  (11 angka bena).

**Penyelesaian:**

$$\begin{array}{r} 3.1415926536 = 0.31415926536 \times 10^1 \\ 3.1415957341 = 0.31415957341 \times 10^1 - \\ \hline -0.30805 \times 10^{-5} \quad (5 \text{ angka bena}) \end{array}$$

$$\begin{array}{ll} \text{in-rounding} & \rightarrow -0.30805 \times 10^{-5} \\ \text{chopping} & \rightarrow -0.30805 \times 10^{-5} \end{array}$$

Jadi, kita kehilangan 6 angka bena!.

■

**Contoh 2.14**

Kurangi  $0.7642 \times 10^3$  dengan  $0.7641 \times 10^3$  (4 angka bena).

**Penyelesaian:**

$$\begin{array}{r} 0.7642 \times 10^3 \\ 0.7641 \times 10^3 - \\ \hline 0.0001 \times 10^3 = 0.1 \times 10^0 \quad (1 \text{ angka bena}) \end{array}$$

$$\begin{array}{ll} \text{in-rounding} & \rightarrow 0.1 \times 10^0 \\ \text{chopping} & \rightarrow 0.1 \times 10^0 \end{array}$$

Jadi kita kehilangan 3 buah angka bena.

■

Kehilangan angka bena bila mengurangkan dua buah bilangan yang hampir sama besar merupakan sumber galat utama pada operasi bilangan titik-kambang. Kehilangan angka bena dapat dihindari dengan mengubah metode komputasi yang digunakan. Tujuan dari pengubahan metode komputasi adalah menghilangkan operasi pengurangan dua buah bilangan yang hampir sama besar, misalnya dengan pengelompokan suku-suku, perkalian dengan bentuk sekawan, menggunakan deret Taylor, atau manipulasi aljabar lainnya. Contoh 2.15 sampai 2.17 berikut mengilustrasikan cara pengubahan ini.

**Contoh 2.15**

[MAT92] Diberikan  $f(x) = x(\sqrt{x+1} - \sqrt{x})$ . Hitunglah  $f(500)$  dengan menggunakan 6 angka bena dan pembulatan ke digit terdekat.

**Penyelesaian:**

$$\begin{aligned}
 f(500) &= 500(\sqrt{501} - \sqrt{500}) \\
 &= 500(22.3830 - 22.3607) \\
 &= 500(0.0223) \\
 &= 11.15 \text{ (empat angka bena)} \\
 &\text{(solusi sejatinya adalah 11.174755300747198..)}
 \end{aligned}$$

Hasil yang tidak akurat ini disebabkan adanya operasi pengurangan dua bilangan yang hampir sama besar, yaitu  $22.3830 - 22.3607$ . Ketelitian hasil dapat kita tingkatkan bila kita dapat menghilangkan pengurangan tersebut. Caranya adalah mengubah metode komputasi sedemikian sehingga pengurangan dua bilangan yang hampir sama besar menjadi hilang.

Susunlah kembali fungsi  $f(x)$  menjadi bentuk yang lebih baik:

$$\begin{aligned}
 f(x) &= x(\sqrt{x+1} - \sqrt{x}) \\
 &= x(\sqrt{x+1} - \sqrt{x}) \frac{(\sqrt{x+1} + \sqrt{x})}{(\sqrt{x+1} + \sqrt{x})} \\
 &= \frac{x[(\sqrt{x+1})^2 - (\sqrt{x})^2]}{(\sqrt{x+1} + \sqrt{x})} \\
 &= \frac{x}{\sqrt{x+1} + \sqrt{x}} = p(x)
 \end{aligned}$$

sehingga

$$p(500) = \frac{500}{\sqrt{501} + \sqrt{500}} = \frac{500}{22.3830 + 22.3607} = 11.1748$$

Hasil ini jauh lebih baik dibandingkan yang pertama. Solusi sejatinya, 11.174755300747198..., jika dibulatkan sampai 6 angka bena adalah 11.1747, yang lebih dekat ke  $p(500)$  daripada ke  $f(500)$ . ■

### Contoh 2.16

Hitunglah akar-akar polinom  $x^2 - 40x + 2 = 0$  sampai 4 angka bena.

**Penyelesaian:**

$$x_{1,2} = \frac{40 \pm \sqrt{(-40)^2 - 8}}{2} = 20 \pm \sqrt{398} = 20.00 \pm 19.95$$

$$x_1 = 20 + 19.95 = 39.95 \text{ (4 angka bena)}$$

$x_2 = 20 - 19.95 = 0.05$  (1 angka bena)  
 (kehilangan tiga buah angka bena akibat pengurangan dua buah bilangan yang hampir sama, yaitu  $20 - 19.95$ )

Nilai  $x_2$  yang lebih akurat dapat diperoleh dengan mengingat lagi pelajaran matematika di sekolah lanjutan bahwa:

jika  $x_1$  dan  $x_2$  adalah akar-akar persamaan  $ax^2 + bx + c = 0$  maka  $x_1x_2 = c/a$

Dengan demikian,  $x_2$  dihitung sebagai berikut:

$$39.95x_2 = 2/1 \Rightarrow x_2 = 2.000/39.95 = 0.05006 \quad (4 \text{ angka bena, lebih akurat}) \quad \blacksquare$$

### Contoh 2.17

[MAT92] Diberikan  $f(x) = \frac{e^x - 1 - x}{x^2}$ . Hitung  $f(0.01)$  sampai 6 angka bena.

**Penyelesaian:**

$$f(0.01) = \frac{e^{0.01} - 1 - 0.01}{0.01^2} = \frac{1.01005 - 1 - 0.01}{0.0001} = 0.5 \quad (1 \text{ angka bena})$$

Hasil yang tidak akurat ini karena adanya kehilangan angka bena akibat pengurangan dua buah nilai yang hampir sama besar, yaitu  $1.01005 - 1$ . Hasil yang lebih akurat dapat diperoleh dengan menguraikan  $f(x)$  ke dalam deret Maclaurin sampai suku orde 2 (Gunakan dalil *L'Hospital* bila menjumpai pembagian 0/0):

$$f(x) \approx p(x) = 1/2 + x/6 + x^2/24$$

sehingga

$$p(0.01) = 1/2 + 0.01/6 + 0.01^2/24 = 0.5 + 0.001667 + 0.00005 = 0.501671$$

Solusi sejatinya adalah 0.50167084168057542..., yang jika dibulatkan sampai 6 angka bena adalah  $0.501671 = p(0.01)$ .  $\blacksquare$

### 2.5.4.2 Operasi Perkalian dan Pembagian

Operasi perkalian dan pembagian dua buah bilangan titik-kambang tidak memerlukan penyamaan pangkat seperti halnya pada penjumlahan. Perkalian dapat dilakukan dengan mengalikan kedua mantis dan menambahkan kedua pangkatnya. Pembagian dikerjakan dengan membagi mantis dan mengurangi pangkatnya. Selain itu, register dengan panjang ganda dibutuhkan untuk menyimpan hasil antara. Hasil akhir dipotong ke dalam register tunggal.

**Contoh 2.18**

Hitung perkalian  $0.4652 \times 10^4$  dengan  $0.1456 \times 10^{-1}$  (4 angka bena).

**Penyelesaian:**

Kalikan mantis:	0.4652	Jumlahkan pangkat:	4
	$0.1456 \times$		$-1$
	<u>0.06773312</u>		<u>3</u>

Gabungkan mantis dengan pangkat:  $0.06773312 \times 10^3$

Normalisasi:  $0.6773312 \times 10^2$

*in-rounding*  $\rightarrow 0.6773 \times 10^2$

*chopping*  $\rightarrow 0.6773 \times 10^2$  ■

**Contoh 2.19**

Hitung  $(0.8675 \times 10^{-5}) / 0.2543 \times 10^{-2}$  (4 angka bena).

**Penyelesaian:**

Bagi mantis:	0.8675	Kurangi pangkat:	-4
	$0.2543 :$		$-2$
	<u>3.4113252</u>		<u>-2</u>

Gabungkan mantis dengan pangkat:  $3.4113252 \times 10^{-2}$

Normalisasi:  $0.34113252 \times 10^{-1}$

*in-rounding*  $\rightarrow 0.3411 \times 10^{-1}$

*chopping*  $\rightarrow 0.3411 \times 10^{-1}$  ■

## 2.6 Perambatan Galat

Galat yang dikandung dalam bilangan titik-kambang merambat pada hasil komputasi. Misalkan terdapat dua bilangan  $a$  dan  $b$  (nilai sejati) dan nilai hampirannya masing-masing  $\hat{a}$  dan  $\hat{b}$ , yang mengandung galat masing-masing  $e_a$  dan  $e_b$ . Jadi, kita dapat menulis

$$a = \hat{a} + e_a$$

dan

$$b = \hat{b} + e_b.$$



Di bawah ini akan diperlihatkan bagaimana galat merambat pada hasil penjumlahan dan perkalian  $a$  dan  $b$ .

Untuk penjumlahan,

$$a + b = (\hat{a} + \mathbf{e}_a) + (\hat{b} + \mathbf{e}_b) = (\hat{a} + \hat{b}) + (\mathbf{e}_a + \mathbf{e}_b) \quad (\text{P.2.24})$$

Jadi, galat hasil penjumlahan sama dengan jumlah galat masing-masing *operand*.

Untuk perkalian,

$$ab = (\hat{a} + \mathbf{e}_a)(\hat{b} + \mathbf{e}_b) = \hat{a}\hat{b} + \hat{a}\mathbf{e}_b + \hat{b}\mathbf{e}_a + \mathbf{e}_a\mathbf{e}_b$$

yang bila kita susun menjadi

$$ab - \hat{a}\hat{b} = \hat{a}\mathbf{e}_b + \hat{b}\mathbf{e}_a + \mathbf{e}_a\mathbf{e}_b$$

Dengan mengandaikan bahwa  $a \neq 0$  dan  $b \neq 0$ , maka galat relatifnya adalah

$$\begin{aligned} \frac{ab - \hat{a}\hat{b}}{ab} &= \frac{\hat{a}\mathbf{e}_b + \hat{b}\mathbf{e}_a + \mathbf{e}_a\mathbf{e}_b}{ab} \\ &= \frac{\hat{a}\mathbf{e}_b}{ab} + \frac{\hat{b}\mathbf{e}_a}{ab} + \frac{\mathbf{e}_a\mathbf{e}_b}{ab} \end{aligned}$$

Dengan mengandaikan bahwa  $a$  dan  $\hat{a}$  hampir sama besar, yaitu  $a \approx \hat{a}$ , begitu juga  $b$  dan  $\hat{b}$ , dan  $\mathbf{e}_a$  dan  $\mathbf{e}_b$  sangat kecil maka  $\hat{a}/a \approx 1$ ,  $\hat{b}/b \approx 1$ , dan  $(\mathbf{e}_a/a)(\mathbf{e}_b/b) \approx 0$ . Dengan demikian

$$\frac{ab - \hat{a}\hat{b}}{ab} = \frac{\mathbf{e}_b}{b} + \frac{\mathbf{e}_a}{a} = \mathbf{e}_{Rb} + \mathbf{e}_{Ra} \quad (\text{P.2.25})$$

Jadi, galat relatif hasil perkalian sama dengan jumlah galat relatif masing-masing *operand*.

Jika operasi aritmetika hanya dilakukan sekali saja, maka kita tidak perlu terlalu khawatir terhadap galat yang ditimbulkannya. Namun, bila operasi dilakukan terhadap seruntunan komputasi, maka galat operasi aritmetika awal akan merambat dalam seruntunan komputasi. Bila hasil perhitungan sebuah operasi aritmetika dipakai untuk operasi selanjutnya, maka akan terjadi penumpukan galat yang semakin besar, yang mungkin mengakibatkan hasil perhitungan akhir menyimpang dari hasil sebenarnya. Ketidakpastian hasil akibat galat pembulatan

yang bertambah besar itu dapat menyebabkan perhitungan menjadi **tidak stabil** (*unstable* atau *instability*), sedangkan lawannya adalah **stabil**, yang merupakan proses numerik yang diinginkan. Metode komputasi dikatakan stabil jika galat pada hasil antara (*intermediate*) hanya sedikit pengaruhnya pada hasil akhir. Jika galat pada hasil antara memberikan pengaruh yang besar pada hasil akhir maka metode komputasinya dikatakan tidak stabil [KRE88]. Ketidakstabilan ini dinamakan "ketidakstabilan numerik", yang dapat dihindari dengan memilih metode komputasi yang stabil (di dalam Bab Solusi Persamaan Diferensial Biasa masalah ini akan dikemukakan lagi). Ketidakstabilan numerik harus dibedakan dengan "ketidakstabilan matematik" dari persoalan yang diberikan. Ketidakstabilan matematik sering dinamakan **kondisi buruk** (*ill conditioned*), yaitu kondisi yang timbul karena hasil perhitungan sangat peka terhadap perubahan kecil data. Kondisi buruk didiskusikan lebih komprehensif di bawah ini.

## 2.7 Kondisi Buruk

Suatu persoalan dikatakan **berkondisi buruk** (*ill conditioned*) bila jawabannya sangat peka terhadap perubahan kecil data (misalnya perubahan kecil akibat pembulatan). Bila kita mengubah sedikit data, maka jawabannya berubah sangat besar (drastis). Lawan dari berkondisi buruk adalah **berkondisi baik** (*well conditioned*). Suatu persoalan dikatakan berkondisi baik bila perubahan kecil data hanya mengakibatkan perubahan kecil pada jawabannya.

Sebagai contoh, tinjau persoalan menghitung akar persamaan kuadrat  $ax^2 + bx + c = 0$  di bawah ini. Di sini kita hanya mengubah nilai nilai tetapan  $c$ -nya saja:

$$(i) \quad x^2 - 4x + 3.999 = 0 \Rightarrow \text{akar-akarnya } x_1 = 2.032 \text{ dan } x_2 = 1.968$$

Sekarang, ubah 3.99 menjadi 4.00:

$$(ii) \quad x^2 - 4x + 4.000 = 0 \Rightarrow \text{akar-akarnya } x_1 = x_2 = 2.000$$

Ubah 4.00 menjadi 4.001:

$$(iii) \quad x^2 - 4x + 4.001 = 0 \Rightarrow \text{akar-akarnya imajiner}$$

Kita katakan bahwa persoalan akar-akar persamaan kuadrat di atas berkondisi buruk, karena dengan pengubahan sedikit saja data masukannya (dalam hal ini nilai koefisien  $c$ ), ternyata nilai akar-akarnya berubah sangat besar.

Kapankah akar persamaan  $f(x) = 0$  berkondisi buruk? Misalkan  $f(x)$  diubah sebesar  $\epsilon$  sehingga akarnya berubah sebesar  $h$ :

$$f(x + h) + \epsilon = 0 \quad \text{P.2.26)}$$

Bila karena pengubahan  $\epsilon$  yang sangat kecil mengakibatkan  $h$  menjadi besar, dikatakan persoalan mencari akar  $f(x) = 0$  berkondisi buruk [NOB72].

**Bukti:**

Kita menggunakan teorema nilai rata-rata (TNR) di dalam kalkulus diferensial:

$$\frac{f(m) - f(n)}{m - n} = f'(t) \quad , \quad m < t < n$$

Karena  $m - n = h \Rightarrow m = n + h$ , maka

$$\frac{f(m + h) - f(n)}{h} = f'(t)$$

atau

$$f(p + h) = f(p) + h f'(t) \quad \text{(P.2.27)}$$

Terapkan (P.2.27) pada (P.2.26):

$$f(x) + h f'(t) + \epsilon = 0 \quad , \quad x < t < x + h \quad \text{(P.2.28)}$$

Pada persoalan pencarian akar,  $f(x) = 0$ , sehingga

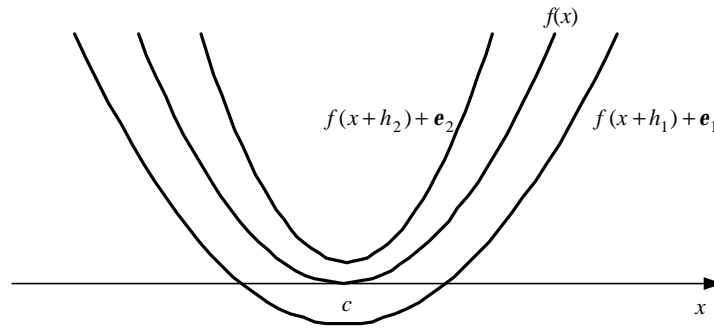
$$h f'(t) + \epsilon = 0 \quad \text{(P.2.29)}$$

atau

$$h = \frac{-\epsilon}{f'(t)} \quad \text{(P.2.30)}$$

Terlihat dari P.2.30 bahwa akar-akar  $f(x) = 0$  akan berkondisi buruk bila  $f'(t)$  bernilai sangat kecil. Bila  $f'(t)$  sangat kecil nilainya maka  $h$  menjadi sangat besar yang berarti akar  $x$  bertambah sebesar  $h$  tersebut. ■

Masalah seperti ini sering muncul pada pencarian akar kembar (seperti pada contoh di atas). Nilai  $f'(x)$  di sekitar akar kembar sangat kecil (mendekati 0), yang berakibat nilai  $h$  menjadi sangat besar (lihat Gambar 2.6).



**Gambar 2.6** Kondisi buruk pada pencarian akar kembar

Contoh persoalan yang berkondisi buruk lainnya adalah persoalan mencari mencari solusi sistem persamaan linier berupa titik potong dari dua buah garis lurus yang berbentuk  $ax + by = c$  yang diubah-ubah koefisiennya ( $b$  dan  $c$ ):

(i)  $x + y = 2$   
 $x + 0.9999y = 1.9999$   
 Solusi:  $x = y = 1.0000$

(ii)  $x + y = 2$   
 $x + 0.9999y = 2.0010$   
 Solusi:  $x = 12, y = -10$

(iii)  $x + y = 2$   
 $x + y = 1.9999$   
 Solusi: tidak ada

(iv)  $x + y = 2$   
 $x + y = 2$   
 Solusi: tidak berhingga, yaitu disepanjang garis  $x + y = 2$

Kondisi buruk yang terjadi pada perhitungan titik potong dua buah garis lurus dapat kita analisis sebagai berikut. Sistem persamaan linier di atas dapat ditulis sebagai

$$\begin{aligned}x + y &= 2 \\x + (1 + \mathbf{e})y &= 2 + \mathbf{d}\end{aligned}$$

yang dalam hal ini  $\mathbf{e}$  dan  $\mathbf{d}$  dalam orde  $10^{-4}$ .

Solusi sistem persamaan linier tersebut adalah

$$\begin{aligned}x + y &= 2 \\x + (1 + \mathbf{e})y &= 2 + \mathbf{d} -\end{aligned}$$


---


$$-\mathbf{e}y = -\mathbf{d} \Rightarrow y = \mathbf{d}/\mathbf{e} \quad \text{dan} \quad x = 2 - \mathbf{d}/\mathbf{e}, \mathbf{e} \neq 0$$

Nilai  $\mathbf{e}$  dan  $\mathbf{d}$  sangat penting. Perubahan kecil  $\mathbf{e}$  mempunyai pengaruh yang besar pada solusi, yang berarti persoalan mencari solusi sistem persamaan linier berkondisi buruk.

## 2.8 Bilangan Kondisi

Kondisi komputasi numerik dapat diukur dengan **bilangan kondisi**. Bilangan kondisi merupakan ukuran tingkat sejauh mana ketidakpastian dalam  $x$  diperbesar oleh  $f(x)$  [CHA88]. Bilangan kondisi dapat dihitung dengan bantuan deret Taylor. Fungsi  $f(x)$  diuraikan di sekitar  $\hat{x}$  sampai suku orde pertama:

$$f(x) \approx f(\hat{x}) + f'(\hat{x})(x - \hat{x}) \quad (\text{P.2.31})$$

Galat relatif hampiran dari  $f(x)$  adalah

$$\mathbf{e}_{RA}[f(x)] = \frac{f(x) - f(\hat{x})}{f(\hat{x})} \approx \frac{f'(\hat{x})(x - \hat{x})}{f(\hat{x})} \quad (\text{P.2.32})$$

dan galat relatif hampiran dari  $x$  adalah

$$\mathbf{e}_{RA}[x] = \frac{x - \hat{x}}{\hat{x}} \quad (\text{P.2.33})$$

Bilangan kondisi didefinisikan sebagai nisbah (*ratio*) antara P.2.32 dan P.2.33:

$$\text{Bilangan kondisi} = \left| \frac{\mathbf{e}_{RA}[f(x)]}{\mathbf{e}_{RA}[x]} \right| = \left| \frac{\hat{x}f'(\hat{x})}{f(\hat{x})} \right| \quad (\text{P.2.34})$$

Arti dari bilangan kondisi adalah:

- bilangan kondisi = 1 berarti galat relatif hampiran fungsi sama dengan galat relatif  $x$
- bilangan kondisi lebih besar dari 1 berarti galat relatif hampiran fungsi besar
- bilangan kondisi lebih kecil dari 1 berarti galat relatif hampiran fungsi kecil (kondisi baik)

Suatu komputasi dikatakan berkondisi buruk jika bilangan kondisinya sangat besar, sebaliknya berkondisi baik bila bilangan kondisinya sangat kecil.

### Contoh 2.20

Misalkan  $f(x) = \sqrt{x}$ . Tentukan bilangan kondisi perhitungan akar kuadrat  $x$ .

#### Penyelesaian:

Hitung  $f'(x)$  terlebih dahulu

$$f'(x) = \frac{1}{2\sqrt{x}}$$

yang akan digunakan untuk menghitung

$$\text{bilangan kondisi} = \left| \frac{\hat{x}/(2\sqrt{\hat{x}})}{\sqrt{\hat{x}}} \right| = \frac{1}{2}$$

Bilangan kondisi ini sangat kecil, yang berarti penarikan akar kuadrat  $x$  merupakan proses yang berkondisi baik. Sebagai contoh,  $\sqrt{20.999} = 4.5824665$ , dan jika 20.999 diubah sedikit (dibulatkan) menjadi 21.000 maka  $\sqrt{21.000} = 4.5825756$ . Ternyata perubahan kecil pada nilai  $x$  hanya berakibat perubahan sedikit pada  $f(x)$ . ■

### Contoh 2.21

Hitung bilangan kondisi  $f(x) = \frac{10}{1-x^2}$ .

#### Penyelesaian:

Hitung  $f'(x)$  terlebih dahulu

$$f'(x) = \frac{20x}{(1-x^2)^2}$$

yang digunakan untuk menghitung

$$\text{bilangan kondisi} = \left| \frac{\hat{x}[20\hat{x}/(1-\hat{x}^2)^2]}{10/(1-\hat{x}^2)} \right| = \left| \frac{2\hat{x}^2}{1-\hat{x}^2} \right|$$

Bilangan kondisi ini sangat besar untuk  $|x| \approx 1$ . Jadi, menghitung  $f(x)$  untuk  $x$  mendekati 1 atau -1 sangat buruk keadaannya, karena galat relatifnya besar. Sebagai contoh,  $f(1.009) = -55.306675$ , tetapi  $f(1.01) = -497.51243$ . Ternyata perubahan kecil pada nilai  $x$  di sekitar 1 (karena dibulatkan dari 4 angka bena menjadi 3 angka bena), mengakibatkan nilai  $f(x)$  berubah sangat besar. Untuk  $x$  yang jauh dari 1 atau -1,  $f(x)$  berkondisi baik. ■

### Contoh 2.22

[CHA91] Hitung bilangan kondisi untuk  $f(x) = \tan(x)$ .

#### Penyelesaian:

Hitung  $f'(x)$  terlebih dahulu

$$f'(x) = \frac{1}{\cos^2(x)}$$

yang digunakan untuk menghitung

$$\text{bilangan kondisi} = \left| \frac{\hat{x}[1/\cos^2(\hat{x})]}{\tan(\hat{x})} \right|$$

Bilangan kondisi ini sangat besar untuk  $x \approx \pi/2$ . Misalkan untuk  $x = \pi/2 + 0.1(\pi/2)$ ,

$$\text{bilangan kondisi} = 1.7279(40.86)/-6.314 = -11.2$$

dan untuk  $x = \pi/2 + 0.01(\pi/2)$ ,

$$\text{bilangan kondisi} = 1.5865(4053)/-63.66 = -101$$

■

Orang yang bijaksana belajar dari kesalahan orang lain, hanya orang yang bodohlah yang belajar dari kesalahannya sendiri.  
(Pepatah Rusia)

## Soal Latihan

1. Tentukan hampiran fungsi di bawah ini ke dalam deret Taylor:

- (a)  $\ln(x)$  sampai orde-4 di sekitar  $x_0=1$ , lalu hampiri nilai  $\ln(0.9)$ .
- (b)  $f(x) = e^x - 1$  sampai orde-3 di sekitar  $x_0=0$ , Lalu hitung nilai  $f(0.0001)$  sampai empat angka bena.
- (c)  $\sinh(x) = \frac{1}{2} (e^x - e^{-x})$  di sekitar  $x_0 = 0$ , lalu hitung nilai hampiran  $\int_0^1 \sinh(x) dx$
- (d)  $\sin(x)$  sampai orde-3, lalu tentukan batas atas galat  $\sin(x)$  jika  $0 \leq x \leq 0.5$ .

2. (a) Tentukan polinom Maclarin orde 4 untuk  $f(x)$ , kemudian gunakan polinom tersebut untuk menghampiri nilai  $f(0.23)$ , serta tentukan batas atas galatnya.

- (i)  $f(x) = \sin(2x)$
- (ii)  $f(x) = \ln(1+x)$

(b) Cari polinom Taylor orde 3 pada  $x_0=1$  untuk  $f(x) = x^3 - 2x^2 + 3x + 5$  dan perlihatkan bahwa ia mewakili  $f(x)$  secara tepat

(c) Hitunglah

$$\int_0^1 \sin(2x) dx$$

- (i) secara analitis (solusi sejati)
- (ii) secara numerik, yang dalam hal ini  $\sin(2x)$  dihamperi dengan deret Maclarin yang telah anda dapatkan pada jawaban 2(a)(i) di atas. Hitung galat mutlak dan galat relatif hasilnya. Pakailah enam angka bena untuk, baik untuk setiap hasil antara maupun hasil akhir.

3. Hitung  $\sqrt{10.1} - \sqrt{10}$  secara langsung tetapi hasil setiap perhitungan antara dan hasil akhir dibulatkan sampai empat angka bena. Kemudian, hitunglah  $\sqrt{10.1} - \sqrt{10}$  dengan cara yang lebih baik.

4. Carilah akar persamaan kuadrat  $x^2 - 10.1x + 1 = 0$  dengan rumus  $abc$ , yang setiap hasil perhitungan antara maupun hasil perhitungan akhir dibulatkan dengan teknik:

- (a) pembulatan ke dalam (*in-rounding*)



(b) pemenggalan (*chopping*)

sampai empat angka bena. Bandingkan hasilnya jika akar terbesar ( $x_1$ ) dihitung dengan rumus  $abc$  dan akar terkecil ( $x_2$ ) dengan rumus  $x_1x_2 = c/a$

5. Diberikan beberapa bilangan titik-kambang yang telah dinormalkan sebagai berikut:

$$a = 0.4523123 \times 10^{-4}$$

$$b = 0.2365401 \times 10^1$$

$$c = 0.4520156 \times 10^{-4}$$

$$d = 0.1234567 \times 10^{-3}$$

Bila mesin yang digunakan untuk operasi aritmetika mempunyai tujuh angka bena, hitung hasil komputasi yang diberikan oleh mesin tersebut (dalam bentuk bilangan titik-kambang ternormalisasi):

(i)  $a + b + c + d$

(ii)  $a + c + d + b$

(iii)  $a - c$

(iv)  $ab - c$

6. Misalkan digunakan mesin hipotetik dengan mantis empat angka bena. Lakukan operasi aritmetika untuk bilangan titik-kambang ternormalisasi berikut. Normalkan hasilnya.

(a)  $0.3796 \times 10^2 + 0.9643 \times 10^{-2}$

(b)  $0.4561 \times 10^{-2} - 0.6732 \times 10^{-2}$

(c)  $0.1234 \times 10^3 \times 0.4321 \times 10^{-1}$

7. Carilah cara yang lebih baik untuk menghitung:

- |  |  |
|--|--|
| (i) $f(x) = (x - \sin(x))/\tan(x)$               | untuk $x$ mendekati nol                  |
| (ii) $f(x) = x - \sqrt{x^2 - a}$                 | untuk $x$ yang jauh lebih besar dari $a$ |
| (iii) $f(x) = \cos^2(x) - \sin^2(x)$             | untuk $x$ di sekitar $\pi/4$             |
| (iv) $f(x) = \log(x+1) - \log(x)$                | untuk $x$ yang besar                     |
| (v) $(1 + \alpha)^{1/2} - 1,  \alpha  \leq 0.01$ | sampai enam angka bena                   |
| (vi) $\sin(\alpha + x) - \sin(\alpha)$           | untuk $x$ yang kecil                     |
| (vii) $(a+x)^n - a^n$                            | untuk $x$ yang kecil                     |
| (viii) $((x^3 - 3x^2) + 3x) - 1$                 | untuk $x = 2.72$                         |
| (ix) $\sqrt{(1 + \cos x)}/2$                     | untuk $x \approx \pi/4$                  |

8. Bagaimana cara menghitung

$$\frac{3a}{4} - \sin(a) + \frac{\sin(2a)}{8}$$

sampai 6 angka bena untuk  $|a| \leq 0.2$ ? (Petunjuk : gunakan deret Maclaurin)

9. Diketahui  $f(x) = \cos(x)$ . Tentukan  $f'(1)$  dengan teorema dasar turunan:

$$\lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} = f'(x)$$

Hitung  $f'(1)$  dengan bermacam-macam  $h = 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001$ . Untuk memperbaiki hasil perhitungan, hitunglah  $f'(1)$  dengan cara yang lebih baik.

10. (a) Hitunglah dengan lima angka bena nilai  $f(13.400)$  bila

$$f(x) = x - 1000(\sqrt{x+0.1} - \sqrt{x})$$

(b) Perbaiki hasil perhitungan anda dengan mengubah metode komputasi.

11. Tentukan bilangan kondisi fungsi-fungsi berikut ini dan tentukan apakah fungsi tersebut berkondisi baik atau berkondisi buruk. Jika berkondisi baik, tentukan di  $x$  berapakah fungsi ini berkondisi buruk (berikan contoh nilai  $x$  untuk memperjelas jawaban anda)

- (a)  $f(x) = 1/(1 - x)$
- (b)  $f(x) = x^2$
- (c)  $f(x) = x^n$
- (d)  $f(x) = \sin(x)$
- (e)  $f(x) = \log(x)$

12. Uraikan  $f(x) = \cos(x)$  di sekitar  $x = \pi/4$ . Hitunglah  $f(\pi/3)$  sampai galat relatif hampiran kurang dari 0.5%. (Petunjuk: hitung suku demi suku, setiap kali menambahkan dengan jumlah suku yang lama, hitung galat relatif hampiran)

Esensi dari matematika adalah kebebasannya  
(George Cantor)

# Bab 3

## Solusi Persamaan Nirlanjar

---

Saya tidak tahu bagaimana saya tampak pada dunia; tetapi bagi saya sendiri saya nampaknya hanyalah seperti seorang anak laki-laki yang bermain-main di pantai, dan mengalihkan diri sendiri sekarang dan kemudian menemukan koral yang lebih halus atau kerang yang lebih indah daripada yang biasa, sementara samudera besar kebenaran semuanya terbentang di hadapan saya tak terungkap.  
(Isaac Newton)

Dalam bidang sains dan rekayasa, para ahli ilmu alam dan rekayasawan sering berhadapan dengan persoalan mencari solusi persamaan –lazim disebut **akar persamaan** (*roots of equation*) atau **nilai-nilai nol**– yang berbentuk  $f(x) = 0$ . Beberapa persamaan sederhana mudah ditemukan akarnya. Misalnya  $2x - 3 = 0$ , pemecahannya adalah dengan memindahkan  $-3$  ke ruas kanan sehingga menjadi  $2x = 3$ , dengan demikian solusi atau akarnya adalah  $x = 3/2$ . Begitu juga persamaan kuadrat seperti  $x^2 - 4x - 5 = 0$ , akar-akarnya mudah ditemukan dengan cara pemfaktoran menjadi  $(x - 5)(x + 1) = 0$  sehingga  $x_1 = 5$  dan  $x_2 = -1$ .

Umumnya persamaan yang akan dipecahkan muncul dalam bentuk nirlanjar (*non linear*) yang melibatkan bentuk sinus, cosinus, eksponensial, logaritma, dan fungsi transenden lainnya. Misalnya,

1. Tentukan akar riil terkecil dari

$$9.34 - 21.97x + 16.3x^3 - 3.704x^5 = 0$$

2. Kecepatan ke atas sebuah roket dapat dihitung dengan memakai rumus berikut:

$$v = u \ln \left| \frac{m_0}{m_0 - qt} \right| - gt$$

yang dalam hal ini  $v$  adalah kecepatan ke atas,  $u$  adalah kecepatan pada saat bahan bakar dikeluarkan relatif terhadap roket,  $m_0$  massa awal roket pada saat  $t = 0$ ,  $q$  laju pemakaian bahan bakar, dan  $g$  percepatan gravitasi ( $= 9.8 \text{ m/det}^2$ ). Jika  $u = 2200 \text{ m/det}$ ,  $m_0 = 160000 \text{ kg}$ , dan  $q = 2680 \text{ kg/det}$ , hitunglah waktu saat  $v = 1000 \text{ m/det}$ . (Nyatakan persamaan dengan ruas kanan sama dengan 0:

$$v - u \ln[m_0/(m_0 - qt)] + gt = 0 \quad )$$

3. Dalam teknik kelautan, persamaan gelombang berdiri yang dipantulkan oleh dermaga pelabuhan diberikan oleh

$$h = h_0 \{ \sin(2px/I) \cos(2ptv/I) + e^{-x} \}$$

Tentukan  $x$  jika  $h = 0.5h_0$ ,  $I = 20$ ,  $t = 10$  dan  $v = 50$ !

(Nyatakan persamaan dengan ruas kanan sama dengan 0:

$$h - h_0 \{ \sin(2px/I) \cos(2ptv/I) + e^{-x} \} = 0 \quad )$$

4. Suatu arus osilasi dalam rangkaian listrik diberikan oleh

$$I = 10e^{-t} \sin(2pt)$$

yang dalam hal ini  $t$  dalam detik. Tentukan semua nilai  $t$  sedemikian sehingga  $I = 2$  ampere.

(Nyatakan persamaan dengan ruas kanan sama dengan 0:

$$I - 10e^{-t} \sin(2pt) = 0 \quad )$$

5. Dalam bidang teknik lingkungan, persamaan berikut ini dapat digunakan untuk menghitung tingkat oksigen pada hilir sungai dari tempat pembuangan limbah:

$$c = 10 - 15(e^{-0.1x} - e^{-0.5x})$$

yang dalam hal ini  $x$  adalah jarak hilir sungai ke tempat pembuangan limbah. Tentukan jarak hilir sungai tersebut bila pembacaan pertama pada alat pengukur tingkat oksigen adalah 4 bila pengukur berada 5 mil dari pembuangan.

(Nyatakan persamaan dengan ruas kanan sama dengan 0:

$$c - 10 - 15(e^{-0.1x} - e^{-0.5x}) = 0 \quad )$$

## 6. Reaksi kesetimbangan



dapat dicirikan oleh hubungan setimbang

$$K = \frac{[C]}{[A]^2 [B]}$$

yang dalam hal ini  $[.]$  menyatakan konsentrasi zat kimia. Andaikan bahwa kita mendefinisikan peubah  $x$  sebagai jumlah mol  $C$  yang dihasilkan. Hukum kekekalan massa dapat dipakai untuk merumuskan ulang hubungan keseimbangan itu sebagai

$$K = \frac{[C_0] + x}{([A_0] - 2x)([B_0] - x)}$$

yang dalam hal ini indeks 0 menunjukkan konsentrasi awal tiap unsur. Jika diketahui tetapan kesetimbangan  $K = 1.25 \times 10^{-2}$ , dan konsentrasi larutan  $[A_0] = 50$ ,  $[B_0] = 40$ , dan  $[C_0] = 5$ , hitunglah  $x$ .

(Nyatakan persamaan dengan ruas kanan sama dengan 0:

$$K - \frac{[C_0] + x}{([A_0] - 2x)([B_0] - x)} = 0 \quad )$$

Keenam contoh di atas memperlihatkan bentuk persamaan yang rumit/kompleks yang tidak dapat dipecahkan secara analitik (seperti persamaan kuadrat pada paragraf awal). Bila metode analitik tidak dapat menyelesaikan persamaan, maka kita masih bisa mencari solusinya dengan menggunakan metode numerik.

## 3.1 Rumusan Masalah

Persoalan mencari solusi persamaan nirlanjar dapat dirumuskan secara singkat sebagai berikut: tentukan nilai  $x$  yang memenuhi persamaan

$$f(x) = 0 \quad (\text{P.3.1})$$

yaitu nilai  $x = s$  sedemikian sehingga  $f(s)$  sama dengan nol.

## 3.2 Metode Pencarian Akar

Dalam metode numerik, pencarian akar  $f(x) = 0$  dilakukan secara lelaran (iteratif). Sampai saat ini sudah banyak ditemukan metode pencarian akar. Secara umum, semua metode pencarian akar tersebut dapat dikelompokkan menjadi dua golongan besar:

1. **Metode tertutup** atau metode pengurung (*bracketing method*)

Metode yang termasuk ke dalam golongan ini mencari akar di dalam selang  $[a, b]$ . Selang  $[a, b]$  sudah dipastikan berisi minimal satu buah akar, karena itu metode jenis ini selalu berhasil menemukan akar. Dengan kata lain, lelarannya selalu konvergen (menuju) ke akar, karena itu metode tertutup kadang-kadang dinamakan juga **metode konvergen**.

2. **Metode terbuka**

Berbeda dengan metode tertutup, metode terbuka tidak memerlukan selang  $[a, b]$  yang mengandung akar. Yang diperlukan adalah tebakan (*guess*) awal akar, lalu, dengan prosedur lelaran, kita menggunakannya untuk menghitung hampiran akar yang baru. Pada setiap kali lelaran, hampiran akar yang lama dipakai untuk menghitung hampiran akar yang baru. Mungkin saja hampiran akar yang baru mendekati akar sejati (konvergen), atau mungkin juga menjauhinya (divergen). Karena itu, metode terbuka tidak selalu berhasil menemukan akar, kadang-kadang konvergen, kadangkala ia divergen.

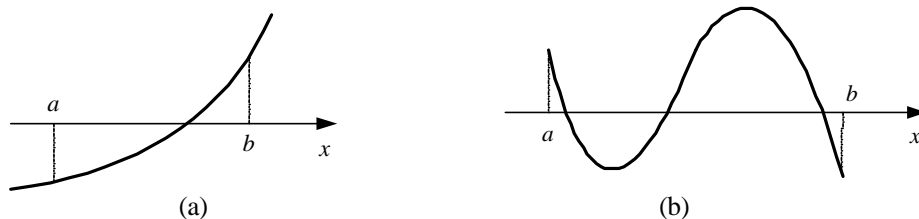
## 3.3 Metode Tertutup

Seperti yang telah dijelaskan, metode tertutup memerlukan selang  $[a, b]$  yang mengandung akar. Sebagaimana namanya, selang tersebut “mengurung” akar sejati. Tata-ancang (*strategy*) yang dipakai adalah mengurangi lebar selang secara sistematis sehingga lebar selang tersebut semakin sempit, dan karenanya menuju akar yang benar.

Dalam sebuah selang mungkin terdapat lebih dari satu buah akar atau tidak ada akar sama sekali. Secara grafik dapat ditunjukkan bahwa jika:

$$(1) f(a)f(b) < 0$$

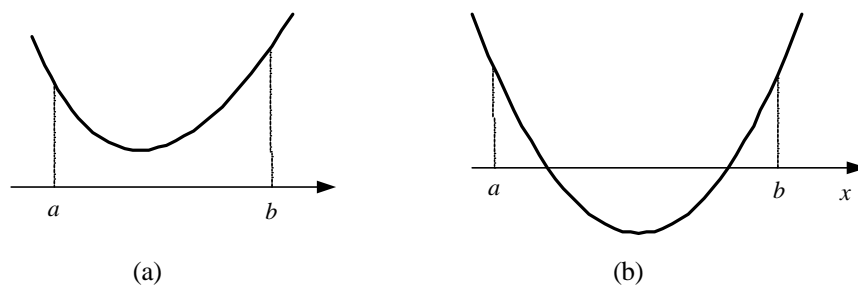
maka terdapat akar sebanyak bilangan ganjil (Gambar 3.1).



**Gambar 3.1** Banyaknya akar ganjil

**(2)  $f(a)f(b) > 0$**

maka terdapat akar sebanyak bilangan genap atau tidak ada akar sama sekali (Gambar 3.2).



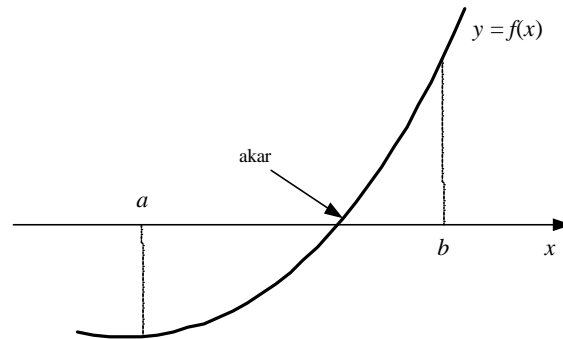
**Gambar 3.2** Banyaknya akar genap

### Syarat Cukup Keberadaan Akar

Gambar 3.1 memperlihatkan bahwa selalu ada akar di dalam selang  $[a, b]$  jika nilai fungsi berbeda tanda (+/-) di  $x = a$  dan  $x = b$ . Tidak demikian halnya jika nilai fungsi di ujung-ujung selang sama tandanya, yang mengisyaratkan mungkin ada akar atau tidak ada sama sekali. Jadi, jika nilai fungsi berbeda tanda di ujung-ujung selang, pastilah terdapat paling sedikit satu buah akar di dalam selang tersebut. Dengan kata lain, syarat cukup keberadaan akar persamaan kita tulis sebagai berikut:

Jika  $f(a)f(b) < 0$  dan  $f(x)$  menerus di dalam selang  $[a, b]$ , maka paling sedikit terdapat satu buah akar persamaan  $f(x) = 0$  di dalam selang  $[a, b]$ .

Syarat ini disebut syarat cukup<sup>1</sup> -bukan syarat perlu- sebab meskipun nilai-nilai di ujung selang tidak berbeda tanda, mungkin saja terdapat akar di dalam selang tersebut (seperti ditunjukkan pada Gambar 3.2). Syarat cukup keberadaan akar ini ditunjukkan pada Gambar 3.3.



**Gambar 3.3** Lokasi akar

Ada dua masalah yang terjadi karena ketidaktepatan mengambil selang  $[a, b]$ . Masalah pertama adalah bila di dalam selang  $[a, b]$  terdapat lebih dari satu buah akar. Sekali suatu metode tertutup digunakan untuk mencari akar di dalam selang  $[a, b]$ , ia hanya menemukan sebuah akar saja. Karena itu, bila kita mengambil selang  $[a, b]$  yang mengandung lebih dari satu akar, hanya satu buah akar saja yang berhasil ditemukan (lihat kembali Gambar 3.1(b)).

Masalah kedua adalah bila mengambil selang  $[a, b]$  yang tidak memenuhi syarat cukup. Adakalanya kita dapat “kehilangan” akar karena selang  $[a, b]$  yang diambil ternyata tidak memenuhi syarat cukup  $f(a)f(b) < 0$ . Sehingga, kita mungkin sampai pada kesimpulan tidak terdapat akar di dalam selang  $[a, b]$  tersebut, padahal seharusnya ada (lihat kembali Gambar 3.2 (b)).

Untuk mengatasi kedua masalah di atas, pengguna metode tertutup disarankan mengambil selang yang berukuran cukup kecil yang memuat hanya satu akar. Ada dua pendekatan yang dapat kita gunakan dalam memilih selang tersebut.

<sup>1</sup> Bentuk implikasi “jika  $p$  maka  $q$ ” bisa dibaca sebagai “ $p$  adalah syarat cukup untuk  $q$ ”. Di dalam kalkulus proposisi, pernyataan “jika  $p$  maka  $q$ ” (dilambangkan dengan  $p \rightarrow q$ ) adalah benar kecuali jika  $p$  benar dan  $q$  salah. Jadi, pernyataan tersebut tetap benar meskipun  $f(a)f(b) > 0$  dan di dalam selang  $[a, b]$  terdapat paling sedikit satu buah akar atau tidak terdapat akar sama sekali. Pernyataan tersebut jelas salah bila  $f(a)f(b) > 0$  dan di dalam selang  $[a, b]$  terdapat paling sedikit satu buah akar (tidak mungkin).



Pendekatan pertama adalah membuat grafik fungsi di bidang  $X$ - $Y$ , lalu melihat di mana perpotongannya dengan sumbu- $X$ . Dari sini kita dapat mengira-ngira selang yang memuat titik potong tersebut. Grafik fungsi dapat dibuat dengan program yang ditulis sendiri, atau lebih praktis menggunakan paket program yang dapat membuat grafik fungsi.

Pendekatan yang kedua adalah dengan mencetak nilai fungsi pada titik-titik absis yang berjarak tetap. Jarak titik ini dapat diatur cukup kecil. Jika tanda fungsi berubah pada sebuah selang, pasti terdapat minimal satu akar di dalamnya. Program 3.1 berisi prosedur untuk menemukan selang yang cukup kecil yang mengandung akar. Program ini mencetak tabel titik-titik sepanjang selang  $[a, b]$ . Dari tabel tersebut kita dapat menentukan upaselang yang nilai fungsi di ujung-ujungnya berbeda tanda. Keberhasilan dari pendekatan ini bergantung pada jarak antara titik-titik absis. Semakin kecil jarak titik absis, semakin besar peluang menemukan selang yang mengandung hanya sebuah akar.

**Program 3.1** Menemukan selang kecil yang mengandung akar

```

procedure Cari_SelangKecilYangMengandungAkar(a, b, h: real);
{ Menentukan dan mencetak nilai-nilai fungsi untuk absis x di dalam
  selang [a, b]. Jarak antara tiap absis adalah h.
  K.Awal: a dan b adalah ujung-ujung selang, nilainya sudah terdefinisi;
          h adalah jarak antara tiap absis x
  K.Akhir: tabel yang berisi x dan f(x) dicetak ke layar
}
var
  x : real;
begin
  x:=a;
  writeln('-----');
  writeln('      x          f(x)  ');
  writeln('-----');
  while x <= b do begin
    writeln(x:5:2, f(x):10:6);
    x:=x+h;
  end;
  { x > b }
  writeln('-----');
end;

```

Bila Program 3.1 digunakan untuk mencari selang kecil yang mengandung akar pada fungsi  $f(x) = e^x - 5x^2$  mulai dari  $a = -0.5$  sampai  $b = 1.4$  dengan kenaikan absis sebesar  $h = 0.1$ , maka hasilnya tampak pada tabel berikut:

$x$	$f(x)$
-0.50	-0.643469
-0.40	-0.129680
-0.30	0.290818
-0.20	0.618731
-0.10	0.854837
0.00	1.000000
0.10	1.055171
0.20	1.021403
0.30	0.899859
0.40	0.691825
0.50	0.398721
0.60	0.022119
0.70	-0.436247
0.80	-0.974459
0.90	-1.590397
1.00	-2.281718
1.10	-3.045834
1.20	-3.879883
1.30	-4.780703
1.40	-5.744800

Berdasarkan tabel di atas, selang yang cukup kecil yang mengandung akar adalah

$[-0.40, -0.30]$  dan  $[0.60, 0.70]$

karena nilai fungsi berubah tanda di ujung-ujung selangnya. Selang  $[0.00, 1.00]$  juga dapat kita ambil tetapi cukup lebar, demikian juga  $[-0.50, 1.40]$ ,  $[-0.30, 0.80]$ , dan seterusnya.

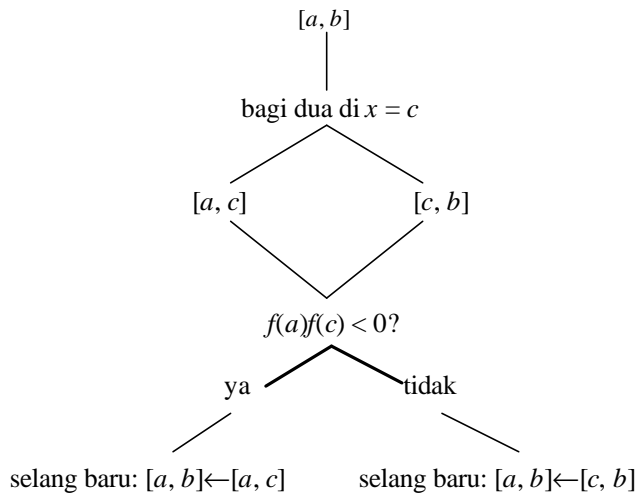
Ada dua metode klasik yang termasuk ke dalam metode tertutup, yaitu **metode bagidua** dan **metode regula-falsi**. Masing-masing metode kita bahas lebih rinci di bawah ini.

### 3.3.1 Metode Bagidua<sup>2</sup>

Misalkan kita telah menentukan selang  $[a, b]$  sehingga  $f(a)f(b) < 0$ . Pada setiap kali lelaran, selang  $[a, b]$  kita bagi dua di  $x = c$ , sehingga terdapat dua buah upaselang yang berukuran sama, yaitu selang  $[a, c]$  dan  $[c, b]$ . Selang yang diambil untuk lelaran berikutnya adalah upaselang yang memuat akar, bergantung pada apakah  $f(a)f(c) < 0$  atau  $f(c)f(b) < 0$ .

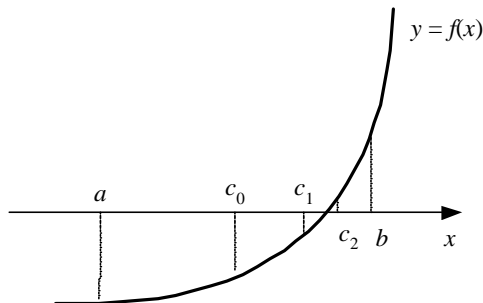
---

<sup>2</sup>Nama lainnya adalah *metode Bolzano*



Selang yang baru dibagi dua lagi dengan cara yang sama. Begitu seterusnya sampai ukuran selang yang baru sudah sangat kecil (lihat Gambar 3.4). Kondisi berhenti lelaran dapat dipilih salah satu dari tiga kriteria berikut:

1. Lebar selang baru:  $|a - b| < \epsilon$ , yang dalam hal ini  $\epsilon$  adalah nilai toleransi lebar selang yang mengurung akar.
2. Nilai fungsi di hampiran akar:  $f(c) = 0$ . Beberapa bahasa pemrograman membolehkan perbandingan dua buah bilangan riil, sehingga perbandingan  $f(c) = 0$  dibenarkan. Namun kalau kita kembali ke konsep awal bahwa dua buah bilangan riil tidak dapat dibandingkan kesamaannya karena representasinya di dalam mesin tidak tepat, maka kita dapat menggunakan bilangan yang sangat kecil (misalnya epsilon mesin) sebagai pengganti nilai 0. Dengan demikian, menguji kesamaan  $f(c) = 0$  dapat kita hampiri dengan  $f(c) < \text{epsilon\_mesin}$ .
3. Galat relatif hampiran akar:  $|(c_{\text{baru}} - c_{\text{lama}})/c_{\text{baru}}| < d$ , yang dalam hal ini  $d$  adalah galat relatif hampiran yang diinginkan.



**Gambar 3.4** Proses pembagian selang  $[a, b]$  dengan metode bagidua

Program 3.2 berisi algoritma metode bagidua. Di dalam algoritma tersebut, format penulisan keluaran tidak dituliskan untuk menghindari kerumitan algoritma dari hal-hal yang tidak esensial.

**Program 3.2** Metode bagidua

```

procedure BagiDua(a,b: real);
{ Mencari akar  $f(x)=0$  di dalam selang  $[a,b]$  dengan metode bagidua
  K.Awal : a dan b adalah ujung-ujung selang sehingga  $f(a)*f(b) < 0$ ,
           nilai a dan b sudah terdefinisi.
  K.Akhir : Hampiran akar tercetak di layar.
}
const
  epsilon1 = 0.000001;      {batas lebar selang akhir lelaran}
  epsilon2 = 0.00000001;    {bilangan yang sangat kecil, mendekati nol}
begin
  repeat
    c:=(a+b)/2;      { titik tengah  $[a,b]$ }
    if f(a)*f(c) < 0 then
      b:=c           {selang baru  $[a,b]=[a,c]$ }
    else
      a:=c;          {selang baru  $[a,b]=[c,b]$ }
  until (ABS(a-b)< epsilon1) or (f(c)) < epsilon2;
  { c adalah akar persamaan }
  writeln('Hampiran kar = ', x:10:6);
end;

```

## Kasus yang Mungkin Terjadi pada Penggunaan Metode Bagidua

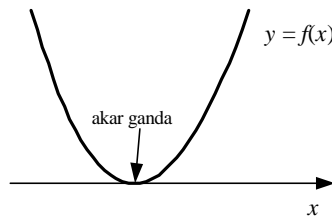
### 1. Jumlah akar lebih dari satu

Bila dalam selang  $[a, b]$  terdapat lebih dari satu akar (banyaknya akar ganjil), hanya satu buah akar yang dapat ditemukan (lihat kembali Gambar 3.1(b)). Cara mengatasinya: gunakan selang  $[a,b]$  yang cukup kecil yang memuat hanya satu buah akar.

### 2. Akar ganda.

Metode bagidua tidak berhasil menemukan akar ganda. Hal ini disebabkan karena tidak terdapat perbedaan tanda di ujung-ujung selang yang baru (Gambar 3.5).

Contoh:  $f(x) = (x - 3)^2 = (x - 3)(x - 3)$ , mempunyai dua akar yang sama, yaitu  $x = 3$ .

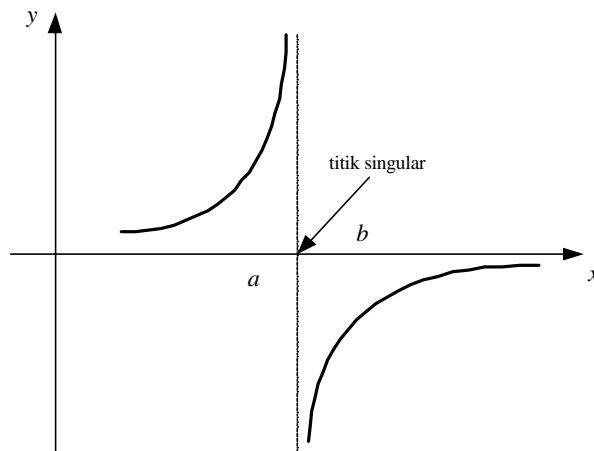


**Gambar 3.4** Akar ganda

Cara mengatasinya: akan dibahas pada upabab 3.5.

### 3. Singularitas.

Pada titik singular, nilai fungsinya tidak terdefinisi. Bila selang  $[a, b]$  mengandung titik singular, lelaran metode bagidua tidak pernah berhenti. Penyebabnya, metode bagidua menganggap titik singular sebagai akar karena lelaran cenderung konvergen. Yang sebenarnya, titik singular bukanlah akar, melainkan *akar semu* (Gambar 3.6)



**Gambar 3.6** Fungsi singular

Cara mengatasinya: periksa nilai  $|f(b) - f(a)|$ . Jika  $|f(b) - f(a)|$  konvergen ke nol, akar yang dicari pasti akar sejati, tetapi jika  $|f(b) - f(a)|$  divergen, akar yang dicari merupakan titik singular (akar semu).

Pada setiap lelaran pada metode bagidua, kita mencatat bahwa selisih antara akar sejati dengan akar hampiran tidak pernah melebihi setengah panjang selang saat itu. Pernyataan ini dinyatakan dengan teorema berikut.

**TEOREMA 3.1.** Jika  $f(x)$  menerus di dalam selang  $[a, b]$  dengan  $f(a)f(b) < 0$  dan  $s \in [a, b]$  sehingga  $f(s) = 0$  dan  $c_r = (a_r + b_r)/2$ , maka selalu berlaku dua ketidaksamaan berikut:

$$(i) \quad |s - c_r| \leq |b_r - a_r| / 2$$

dan

$$(ii) \quad |s - c_r| \leq \frac{|b - a|}{2^{r+1}}, \quad r = 0, 1, 2, \dots$$

**Bukti:**

Misalkan pada lelaran ke- $r$  kita mendapatkan selang  $[a_r, b_r]$  yang panjangnya setengah panjang selang sebelumnya,  $[a_{r-1}, b_{r-1}]$ .

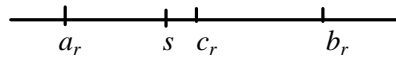
Jadi,

$$|b_r - a_r| = |b_{r-1} - a_{r-1}| / 2$$

Jelaslah bahwa

$$\begin{aligned} |b_1 - a_1| &= |b_0 - a_0| / 2 = |b - a| / 2 \\ |b_2 - a_2| &= |b_1 - a_1| / 2 = |b - a| / 2^2 \\ |b_3 - a_3| &= |b_2 - a_2| / 2 = |b - a| / 2^3 \\ &\dots \\ |b_r - a_r| &= |b_{r-1} - a_{r-1}| / 2 = |b - a| / 2^r \end{aligned}$$

Pada lelaran ke- $r$ , posisi  $c_r$  (akar hampiran) dan  $s$  (akar sejati) adalah seperti diagram berikut:



Berdasarkan diagram di atas jelaslah bahwa

$$|s - c_r| \leq \frac{|b_r - a_r|}{2}$$

Selanjutnya,

$$|s - c_r| \leq \frac{|b_r - a_r|}{2} = \frac{1}{2} \frac{|b - a|}{2^r} = \frac{|b - a|}{2^{r+1}}$$

■

Jadi, selisih antara akar sejati dengan akar hampiran tidak pernah lebih dari setengah epsilon.

Dengan mengingat kriteria berhenti adalah  $|b_r - a_r| < \epsilon$ , maka dari (i) terlihat bahwa

$$|s - c_r| < a / 2$$

sehingga

$$\begin{aligned} \frac{|b-a|}{2^{r+1}} &< \frac{e}{2} \\ \Leftrightarrow 2^r &> |b-a| / e \\ \Leftrightarrow r \ln(2) &> \ln(|b-a|) - \ln(e) && \text{ket: } \ln \text{ adalah logaritma natural} \\ \Leftrightarrow r &> \frac{\ln(|b-a|) - \ln(e)}{\ln(2)} \\ \Leftrightarrow R &> \frac{\ln(|b-a|) - \ln(e)}{\ln(2)} \end{aligned}$$

yang dalam hal ini  $R$  adalah jumlah lelaran (jumlah pembagian selang) yang dibutuhkan untuk menjamin bahwa  $c$  adalah hampiran akar yang memiliki galat kurang dari  $e$ .

### Contoh 3.1

Temukan akar  $f(x) = e^x - 5x^2$  di dalam selang  $[0, 1]$  dan  $e = 0.00001$ .

**Penyelesaian:** Tabel lelaran menggunakan metode bagidua:

$r$	$a$	$c$	$b$	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebar nya
0	0.000000	0.500000	1.000000	1.000000	0.398721	-2.281718	[c, b]	0.500000
1	0.500000	0.750000	1.000000	0.398721	-0.695500	-2.281718	[a, c]	0.250000
2	0.500000	0.625000	0.750000	0.398721	-0.084879	-0.695500	[a, c]	0.125000
3	0.500000	0.562500	0.625000	0.398721	0.173023	-0.084879	[c, b]	0.062500
4	0.562500	0.593750	0.625000	0.173023	0.048071	-0.084879	[c, b]	0.031250
5	0.593750	0.609375	0.625000	0.048071	-0.017408	-0.084879	[a, c]	0.015625
6	0.593750	0.601563	0.609375	0.048071	0.015581	-0.017408	[c, b]	0.007813
7	0.601563	0.605469	0.609375	0.015581	-0.000851	-0.017408	[a, c]	0.003906
8	0.601563	0.603516	0.605469	0.015581	0.007380	-0.000851	[c, b]	0.001953
9	0.603516	0.604492	0.605469	0.007380	0.003268	-0.000851	[c, b]	0.000977

10	0.604492	0.604980	0.605469	0.003268	0.001210	-0.000851	[c, b]	0.000488
11	0.604980	0.605225	0.605469	0.001210	0.000179	-0.000851	[c, b]	0.000244
12	0.605225	0.605347	0.605469	0.000179	-0.000336	-0.000851	[a, c]	0.000122
13	0.605225	0.605286	0.605347	0.000179	-0.000078	-0.000336	[a, c]	0.000061
14	0.605225	0.605255	0.605286	0.000179	0.000051	-0.000078	[c, b]	0.000031
15	0.605255	0.605270	0.605286	0.000051	-0.000014	-0.000078	[a, c]	0.000015
16	0.605255	0.605263	0.605270	0.000051	0.000018	-0.000014	[c, b]	0.000008

Jadi, hampiran akarnya adalah  $x = 0.605263$

Jumlah lelaran yang dibutuhkan

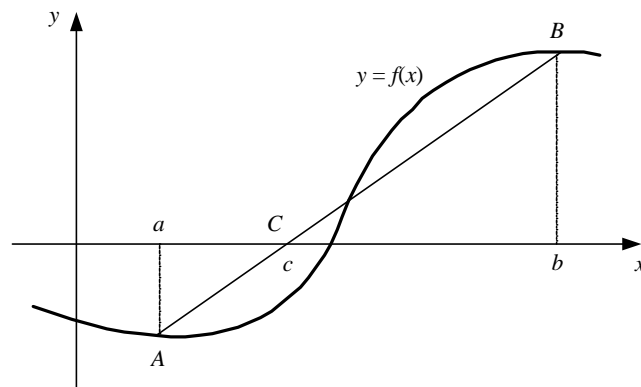
$$R > \frac{\ln(|1 - 0|) - \ln(0.00001)}{\ln(2)}$$

$$> 16.60964$$

Jadi, dibutuhkan minimal 17 kali lelaran ( $r=0$  sampai dengan  $r=16$ ), sesuai dengan jumlah lelaran pada tabel, agar galat akar hampiran kurang dari  $e$ . ■

### 3.3.2 Metode Regula-Falsi

Meskipun metode bagidua selalu berhasil menemukan akar, tetapi kecepatan konvergensinya sangat lambat. Kecepatan konvergensi dapat ditingkatkan bila nilai  $f(a)$  dan  $f(b)$  juga turut diperhitungkan. Logikanya, bila  $f(a)$  lebih dekat ke nol daripada  $f(b)$  tentu akar lebih dekat ke  $x = a$  daripada ke  $x = b$ . Metode yang memanfaatkan nilai  $f(a)$  dan  $f(b)$  ini adalah **metode regula-falsi** (bahasa Latin) atau **metode posisi palsu**. (*false position method*). Dengan metode regula-falsi, dibuat garis lurus yang menghubungkan titik  $(a, f(a))$  dan  $(b, f(b))$ . Perpotongan garis tersebut dengan sumbu- $x$  merupakan taksiran akar yang diperbaiki. Garis lurus tadi seolah-olah berlaku menggantikan kurva  $f(x)$  dan memberikan posisi palsu dari akar.



Gambar 3.7 Metode regula-falsi



Perhatikan Gambar 3.7:

gradien garis AB = gradien garis BC

$$\frac{f(b)-f(a)}{b-a} = \frac{f(b)-0}{b-c}$$

yang dapat disederhanakan menjadi

$$c = b - \frac{f(b)(b-a)}{f(b)-f(a)} \quad (\text{P.3.2})$$

Algoritma regula-falsi (lihat Program 3.3) hampir sama dengan algoritma bagidua kecuali pada perhitungan nilai  $c$ .

**Program 3.3** Metode regula-falsi

```

procedure regula_falsi(a, b: real);
{ Mencari akar  $f(x)=0$  di dalam selang  $[a,b]$  dengan metode regulafalsi
  K.Awal :  $a$  dan  $b$  adalah ujung-ujung selang sehingga  $f(a)*f(b) < 0$ ,
           harga  $a$  dan  $b$  sudah terdefinisi
  K.Akhir : Hampiran akar tercetak di layar
}
const
  epsilon1 = 0.00001;           {batas lebar selang akhir lelaran}
  epsilon2 = 0.000001;         {bilangan yang sangat kecil, bisa diganti }
begin
  repeat
     $c := b - (f(b)*(b-a)/(f(b)-f(a)))$ ;
    if abs(f(c)) < epsilon2 then           {f(c) = 0, c adalah akar}
      begin
        a:=c;
        b:=c;
      end
    else
      if f(a)*f(c) < 0 then
        b:=c; {selang baru  $[a,b]=[a,c]$ }
      else
        a:=c; {selang baru  $[a,b]=[c,b]$ }
      until ABS(a-b) < epsilon1;
      { c adalah hampiran akar }
      writeln('Hampiran akar : ', c:10:6);
    end;

```

Secara umum, metode regula-falsi lebih cepat konvergensinya dibandingkan dengan metode bagidua. Namun, pada beberapa kasus kecepatan konvergensinya justru lebih lambat. Bila kita memakai Program 3.4 untuk menghitung akar  $f(x) = e^x - 5x^2$  di dalam selang  $[0, 1]$  dan  $\epsilon = 0.00001$ , maka tabel lelarannya yang dihasilkan adalah sebagai berikut:

$r$	$a$	$c$	$b$	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebarnya
0	0.000000	0.304718	1.000000	1.000000	0.891976	-2.281718	[c,b]	0.695282
1	0.304718	0.500129	1.000000	0.891976	0.398287	-2.281718	[c,b]	0.499871
2	0.500129	0.574417	1.000000	0.398287	0.126319	-2.281718	[c,b]	0.425583
3	0.574417	0.596742	1.000000	0.126319	0.035686	-2.281718	[c,b]	0.403258
4	0.596742	0.602952	1.000000	0.035686	0.009750	-2.281718	[c,b]	0.397048
5	0.602952	0.604641	1.000000	0.009750	0.002639	-2.281718	[c,b]	0.395359
6	0.604641	0.605098	1.000000	0.002639	0.000713	-2.281718	[c,b]	0.394902
7	0.605098	0.605222	1.000000	0.000713	0.000192	-2.281718	[c,b]	0.394778
8	0.605222	0.605255	1.000000	0.000192	0.000052	-2.281718	[c,b]	0.394745
9	0.605255	0.605264	1.000000	0.000052	0.000014	-2.281718	[c,b]	0.394736
10	0.605264	0.605266	1.000000	0.000014	0.000004	-2.281718	[c,b]	0.394734
11	0.605266	0.605267	1.000000	0.000004	0.000001	-2.281718	[c,b]	0.394733
12	0.605267	0.605267	1.000000	0.000001	0.000000	-2.281718	[c,b]	0.394733
13	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
14	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
15	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
16	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
17	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
18	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
19	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
20	0.605267	0.605267	1.000000	0.000000	0.000000	-2.281718	[c,b]	0.394733
21	0.605267	0.605267	1.000000	0.000000	-0.000000	-2.281718	[a,c]	0.000000

Jumlah lelaran tabel di atas = 22, lebih banyak daripada jumlah lelaran metode bagidua. Bila diperhatikan, dari lelaran 12 sampai lelaran 21, nilai  $a$ ,  $b$ ,  $c$  tidak pernah berubah, padahal  $f(c)$  sudah sangat kecil ( $\approx 0$ ). Kasus seperti ini akan terjadi bila kurva fungsinya cekung (konkaf) di dalam selang  $[a, b]$ . Akibatnya, garis potongnya selalu terletak di atas kurva (bila kurvanya cekung ke atas) atau selalu terletak di bawah kurva (bila kurvanya cekung ke bawah). Perhatikan Gambar 3.8.

**Gambar 3.8** Garis potong selalu terletak di atas kurva  $y = f(x)$

Pada kondisi yang paling ekstrim,  $|b - a_r|$  tidak pernah lebih kecil dari  $\epsilon$ , sebab salah satu titik ujung selang, dalam hal ini  $b$ , selalu tetap untuk setiap lelaran  $r = 0, 1, 2, \dots$ . Titik ujung selang yang tidak pernah berubah itu dinamakan **titik mandek** (*stagnant point*). Pada titik mandek,

$$|b_r - a_r| = |b - a_r| \quad r = 0, 1, 2, \dots$$

yang dapat mengakibatkan program mengalami *looping*. Untuk mengatasi hal ini, kondisi berhenti pada algoritma regula-falsi harus kita tambah dengan memeriksa apakah nilai  $f(c)$  sudah sangat kecil sehingga mendekati nol. Jadi, kondisi pada repeat-until menjadi

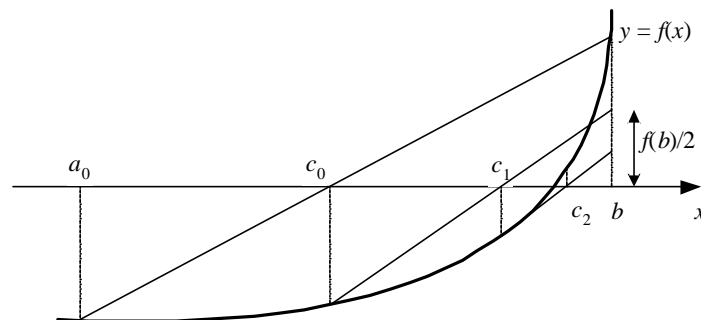
**until** (ABS(a-b) < epsilon1) **or** (ABS(f(c)) < epsilon2)

Bila perubahan ini diterapkan pada soal pencarian akar di atas dengan  $\epsilon_{\text{psilon2}} = 0.000001$ , lelarannya akan berhenti pada  $r = 12$  dengan akar  $x = 0.605267$ .

### Perbaikan Metode Regula-Falsi

Untuk mengatasi kemungkinan kasus titik mandek, metode regula-falsi kemudian diperbaiki (*modified false position method*). Caranya, pada akhir lelaran  $r = 0$ , kita sudah memperoleh selang baru akan dipakai pada lelaran  $r = 1$ . Berdasarkan selang baru tersebut, tentukan titik ujung selang yang tidak berubah (jumlah perulangan  $> 1$ ) - yang kemudian menjadi titik mandek. Nilai  $f$  pada titik mandek itu diganti menjadi setengah kalinya, yang akan dipakai pada lelaran  $r = 1$ .

Misalkan fungsi  $f(x)$  cekung ke atas di dalam selang  $[a, b]$  seperti yang ditunjukkan pada Gambar 3.9.



**Gambar 3.9** Perbaikan metode regula-falsi

Setelah menghitung nilai  $c_0$  pada lelaran  $r = 0$ , ujung selang  $b$  untuk lelaran  $r = 1$  tidak berubah. Titik  $b$  menjadi titik mandek. Karena itu, untuk lelaran  $r = 1$ , nilai  $f(b)$  yang dipakai adalah  $f(b)/2$ . Begitu juga untuk lelaran  $r = 2$ , nilai  $f(b)$  yang dipakai adalah setengah dari nilai  $f(b)$  sebelumnya. Pada akhir lelaran  $r = 2$ ,  $c_2$  sudah terletak di bawah kurva  $y = f(x)$ . Selang yang dipakai selanjutnya adalah  $[c_1, c_2]$ . Dengan cara ini kita dapat menghilangkan titik mandek yang berkepanjangan. Program 3.3 kita modifikasi menjadi Program 3.4.

**Program 3.4** Metode regula-falsi yang diperbaiki

```

procedure perbaikan_regula_falsi(a, b: real);
{ Mencari akar  $f(x)=0$  di dalam selang  $[a,b]$  dengan metode regula-falsi
  yang diperbaiki
  K.Awal : a dan b adalah ujung-ujung selang sehingga  $f(a)*f(b) < 0$ ,
           harga a dan b sudah terdefinisi
  K.Akhir : akar persamaan tercetak di layar
}
const
  epsilon1 = 0.00001; {batas lebar selang akhir lelaran}
  epsilon2 = 0.000001; {batas galat nilai fungsi di hampiran akar}
var
  FA, FB, simpan : real;
  mandek_kiri, mandek_kanan : integer; {jumlah perulangan titik
                                         ujung selang}
begin
  FA:=f(a); FB:=f(b);
  mandek_kiri:=1; mandek_kanan:=1;
  repeat
    c:=b-(FB*(b-a)/(FB-FA));
    if abs(f(c)) < epsilon2 then {f(c) = 0, c adalah akar}
      begin
        a:=c;
        b:=c;
      end
    else
      begin
        if f(a)*f(c) < 0 then
          begin
            b:=c {selang baru [a,b]=[a,c]}
            FB:=f(c);
            mandek_kiri:=mandek_kiri + 1;
            mandek_kanan:=0;
            if mandek_kiri > 1 then
              FA:=FA/2; {a menjadi titik mandek }
            end
          else
            begin
              a:=c; {selang baru [a,b]=[c,b]}
              FA:=f(c);
              mandek_kanan:=mandek_kanan + 1;
              mandek_kiri:=0;
              if mandek_kanan > 1 then
                FB:=FB/2; {b menjadi titik mandek}
              end
            end
          end
        end
      end
  until abs(f(c)) < epsilon1;
end;

```

```

end;
until (ABS(a-b) < epsilon1) OR (ABS(f(c)) < epsilon2);
{ c adalah taksiran akar }
writeln('Hampiran akar : ', c:10:6);
end;

```

Tabel lelaran dari Program 3.4 untuk menghitung akar  $f(x) = e^x - 5x^2$  di dalam selang  $[0, 1]$  dengan  $\epsilon = 0.00001$  dan  $\delta = 0.000001$  adalah sebagai berikut:

$r$	$a$	$c$	$b$	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebar nya
0	0.000000	0.304718	1.000000	1.000000	0.891976	-2.281718	[c,b]	0.695282
						(*2)		
1	0.304718	0.609797	1.000000	0.891976	-0.019205	-1.140859	[a,c]	0.305079
2	0.304718	0.603367	0.609797	0.891976	0.008005	-0.019205	[c,b]	0.006430
3	0.603367	0.605259	0.609797	0.008005	0.000035	-0.019205	[c,b]	0.004538
						(*2)		
4	0.605259	0.605275	0.609797	0.000035	-0.000035	-0.009602	[a,c]	0.000017
5	0.605259	0.605267	0.605275	0.000035	0.000000	-0.000035	[c,b]	0.000008

Hampiran akar  $x = 0.605267$

Terlihat bahwa jumlah lelarannya berkurang menjadi sepertiga semula. Harus dicatat bahwa metode regula-falsi yang diperbaiki tetap berlaku untuk fungsi yang tidak cekung sekalipun. Jadi, jika anda memprogram dengan metode regula-falsi, pakailah Program 3.4 ini untuk semua kemungkinan kasus fungsi.

### 3.4 Metode Terbuka

Tidak seperti pada metode tertutup, metode terbuka tidak memerlukan selang yang mengurung akar. Yang diperlukan hanya sebuah tebakan awal akar atau dua buah tebakan yang tidak perlu mengurung akar. Inilah alasan mengapa metodenya dinamakan metode terbuka. Hampiran akar sekarang didasarkan pada hampiran akar sebelumnya melalui prosedur lelaran. Kadangkala lelaran konvergen ke akar sejati, kadangkala ia divergen. Namun, apabila lelarannya konvergen, konvergensinya itu berlangsung sangat cepat dibandingkan dengan metode tertutup.

Yang termasuk ke dalam metode terbuka:

1. Metode lelaran titik-tetap (*fixed-point iteration*)
2. Metode Newton-Raphson
3. Metode *secant*

### 3.4.1. Metode Lelaran Titik-Tetap

Metode ini kadang-kadang dinamakan juga metode lelaran sederhana, metode langsung, atau metode sulih beruntun. Kesederhanaan metode ini karena pembentukan prosedur lelarannya mudah dibentuk sebagai berikut:

Susunlah persamaan  $f(x) = 0$  menjadi bentuk  $x = g(x)$ . Lalu, bentuklah menjadi prosedur lelaran

$$x_{r+1} = g(x_r) \quad (\text{P.3.3})$$

dan terkalah sebuah nilai awal  $x_0$ , lalu hitung nilai  $x_1, x_2, x_3, \dots$ , yang mudah-mudahan konvergen ke akar sejati  $s$  sedemikian sehingga

$$f(s) = 0 \text{ dan } s = g(s).$$

Kondisi berhenti lelaran dinyatakan bila

$$|x_{r+1} - x_r| < e$$

atau bila menggunakan galat relatif hampiran

$$\left| \frac{x_{r+1} - x_r}{x_{r+1}} \right| < d$$

dengan  $e$  dan  $d$  telah ditetapkan sebelumnya. Program lelaran titik-tetap ditunjukkan oleh Program 3.5.

**Program 3.5** Metode lelaran titik-tetap

```
procedure lelaran_titik_tetap(x:real);
{ mencari akar  $f(x) = 0$  dengan metode lelaran titik-tetap
  K.Awal : x adalah tebakan awal akar, nilainya sudah terdefinisi
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon = 0.000001;
var
  x_sebelumnya: real;

  function g(x:real): real;
  {mengembalikan nilai  $g(x)$ . Definisikan  $g(x)$ , ), lihat Contoh 3.2 }

begin
  repeat
    x_sebelumnya:=x;
```

```

x:=g(x);
until ABS(x-x_sebelumnya) < epsilon;
{ x adalah hampiran akar }

write('Hampiran akar x = ', x:10:6);
end;

```

Program 3.5 hanya menangani lelaran yang konvergen. Program harus dimodifikasi menjadi Program 3.6 untuk menangani lelaran yang divergen. Salah satu cara penanganannya adalah dengan membatasi jumlah maksimum lelaran ( $N_{maks}$ ). Jika jumlah lelaran lebih besar dari  $N_{maks}$ , maka diasumsikan lelarannya divergen.

**Program 3.6** Metode lelaran titik-tetap (dengan penanganan kasus divergen)

```

procedure lelaran_titik_tetap(x:real);
{ mencari akar  $f(x) = 0$  dengan metode lelaran titik-tetap
  K.Awal : x adalah tebakan awal akar, nilainya sudah terdefinisi
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon = 0.000001;
  Nmaks = 30;
var
  x_sebelumnya: real; { hampiran nilai akar pada lelaran sebelumnya }
  i : integer;        { pencacah jumlah lelaran }

  function g(x:real): real;
  {mengembalikan nilai  $g(x)$ . Definisikan  $g(x)$  di sini, lihat Contoh 3.2 }

begin
  i:=0;
  repeat
    x_sebelumnya:=x;
    x:=g(x);
    i:=i+1;
  until (ABS(x-x_sebelumnya) < epsilon) or (i > Nmaks);
  { x adalah hampiran akar }

  if i > Nmaks then
    write('Divergen!')
  else
    write('Hampiran akar x = ', x:10:6);
end;

```

### Contoh 3.2

Carilah akar persamaan  $f(x) = x^2 - 2x - 3 = 0$  dengan metode lelaran titik-tetap. Gunakan  $\epsilon = 0.000001$ .

### Penyelesaian:

Terdapat beberapa kemungkinan prosedur lelaran yang dapat dibentuk.

(a)  $x^2 - 2x - 3 = 0$

$$x^2 = 2x + 3$$

$$x = \sqrt{2x + 3}$$

Dalam hal ini,  $g(x) = \sqrt{2x + 3}$ . Prosedur lelarannya adalah  $x_{r+1} = \sqrt{2x_r + 3}$ .  
Ambil terkaan awal  $x_0=4$

Tabel lelarannya:

$r$	$x_r$	$ x_{r+1} - x_r $
0	4.000000	-
1	3.316625	0.683375
2	3.103748	0.212877
3	3.034385	0.069362
4	3.011440	0.022945
5	3.003811	0.007629
6	3.001270	0.002541
7	3.000423	0.000847
8	3.000141	0.000282
9	3.000047	0.000094
10	3.000016	0.000031
11	3.000005	0.000010
12	3.000002	0.000003
13	3.000001	0.000001
14	3.000000	0.000000

Hampiran akar  $x = 3.000000$  (konvergen monoton)

(b)  $x^2 - 2x - 3 = 0$   
 $x(x-2) = 3$   
 $x = 3/(x - 2)$

Dalam hal ini,  $g(x) = 3/(x - 2)$ . Prosedur lelarannya adalah  $x_{r+1} = 3/(x_r - 2)$ .  
Ambil terkaan awal  $x_0 = 4$

Tabel lelarannya:

$i$	$x_r$	$ x_{r+1} - x_r $
0	4.000000	-
1	1.500000	2.500000
2	-6.000000	7.500000
3	-0.375000	5.625000
4	-1.263158	0.888158
5	-0.919355	0.343803
6	-1.027624	0.108269



7	-0.990876	0.036748
8	-1.003051	0.012175
9	-0.998984	0.004066
10	-1.000339	0.001355
11	-0.999887	0.000452
12	-1.000038	0.000151
13	-0.999987	0.000050
14	-1.000004	0.000017
15	-0.999999	0.000006
16	-1.000000	0.000002
17	-1.000000	0.000001

Hampiran akar  $x = -1.000000$  (konvergen beresilasi)

(c)  $x^2 - 2x - 3 = 0$   
 $x = (x^2 - 3)/2$

Prosedur lelarannya adalah  $x_{r+1} = (x_r^2 - 3)/2$ . Ambil terkaan awal  $x_0=4$

Tabel lelarannya:

$i$	$x_r$	$ x_{r+1} - x_r $
0	4.000000	-
1	6.500000	2.500000
2	19.625000	13.125000
3	191.070313	171.445312
4	18252.432159	18061.361847
...		

Ternyata lelarannya divergen! ■

### Contoh 3.3

Apa yang terjadi dengan pemilihan beragam nilai  $x_0$  pada pencarian akar persamaan

$$x^3 + 6x - 3 = 0$$

dengan prosedur lelaran

$$x_{r+1} = \frac{-x_r^3 + 3}{6} \quad [\text{PUR84}]$$

Cobakan dengan:  $x_0=0.5$ ,  
 $x_0=1.5$ ,  
 $x_0=2.2$ ,  
 $x_0=2.7$

### Penyelesaian:

Tabel lelarannya adalah sebagai berikut:

$r$	$x_r$	$r$	$x_r$	$r$	$x_r$	$r$	$x_r$
0	0.5	0	1.5	0	2.2	0	2.7
1	0.4791667	1	-0.0625	1	-1.2744667	1	-2.7805
2	0.4816638	2	0.5000407	2	0.8451745	2	4.0827578
3	0.4813757	3	0.4791616	3	0.3993792	3	-10.842521
...	...	4	0.4816644	4	0.4893829	4	212.9416
7	0.4814056	...	...	...	...	5	-16909274.5
8	0.4814056	9	0.4814056	9	0.4814054		
		10	0.4814056	10	0.4814056		
				11	0.4814056		

Konvergen

Divergen

Terlihat dengan pengambilan  $x_0$  yang cukup dekat ke akar sejati, proses akan konvergen, tetapi jika kita mengambil  $x_0$  terlalu jauh dari akar sejati, ia akan divergen. ■

Kadang-kadang lelaran konvergen, kadang-kadang ia divergen. Adakah suatu “tanda” bagi kita untuk mengetahui kapan suatu lelaran konvergen dan kapan divergen?

### Kriteria konvergensi

Diberikan prosedur lelaran

$$x_{r+1} = g(x_r) \quad (\text{P.3.4})$$

Misalkan  $x = s$  adalah solusi  $f(x) = 0$  sehingga  $f(s) = 0$  dan  $s = g(s)$ . Selisih antara  $x_{r+1}$  dan  $s$  adalah

$$\begin{aligned} x_{r+1} - s &= g(x_r) - s \\ &= \frac{g(x_r) - s}{(x_r - s)} (x_r - s) \end{aligned} \quad (\text{P.3.5})$$

Terapkan teorema nilai rata-rata pada persamaan (P.3.5) sehingga

$$x_{r+1} - s = g'(t)(x_r - s) \quad (\text{P.3.6})$$

yang dalam hal ini  $x_{r+1} < t < s$ . Misalkan galat pada lelaran ke- $r$  dan lelaran ke- $(r+1)$  adalah

$$\mathbf{e}_r = x_r - s \text{ dan } \mathbf{e}_{r+1} = x_{r+1} - s$$

Persamaan (P.4.6) dapat kita tulis menjadi

$$\mathbf{e}_{r+1} = g'(t) \mathbf{e}_r \quad (\text{P.3.7})$$

atau dalam tanda mutlak

$$|\mathbf{e}_{r+1}| = |g'(t)| |\mathbf{e}_r| \leq K |\mathbf{e}_r|$$

Berapakah batas-batas nilai  $K$  itu?

Misalkan  $x_0$  dan  $x$  berada di dalam selang sejauh  $2h$  dari  $s$ , yaitu  $s - h < x < s + h$ . Jika lelaran konvergen di dalam selang tersebut, yaitu  $x_0, x_1, x_2, x_3, \dots$  menuju  $s$ , maka galat setiap lelaran berkurang. Jadi, haruslah dipenuhi kondisi

$$|\mathbf{e}_{r+1}| \leq K |\mathbf{e}_r| \leq K^2 |\mathbf{e}_{r-1}| \leq K^3 |\mathbf{e}_{r-2}| \leq \dots \leq K^{r+1} |\mathbf{e}_0|$$

Kondisi tersebut hanya berlaku jika

$$g'(x) \leq K < 1$$

Karena  $K < 1$ , maka  $K^{r+1} \rightarrow 0$  untuk  $r \rightarrow \infty$ ; di sini  $|x_{r+1} - s| \rightarrow 0$ .

**TEOREMA 3.2.** Misalkan  $g(x)$  dan  $g'(x)$  menerus di dalam selang  $[a, b] = [s-h, s+h]$  yang mengandung titik tetap  $s$  dan nilai awal  $x_0$  dipilih dalam selang tersebut. Jika  $|g'(x)| < 1$  untuk semua  $x \in [a, b]$  maka lelaran  $x_{r+1} = g(x_r)$  akan konvergen ke  $s$ . Pada kasus ini  $s$  disebut juga *titik atraktif*. Jika  $|g'(x)| > 1$  untuk semua  $x \in [a, b]$  maka lelaran  $x_{r+1} = g(x_r)$  akan divergen dari  $s$ .

Teorema 3.2 dapat kita sarikan sebagai berikut:

Di dalam selang  $I = [s-h, s+h]$ , dengan  $s$  titik tetap,

jika  $0 < g'(x) < 1$  untuk setiap  $x \in I$ , maka lelaran *konvergen monoton*;

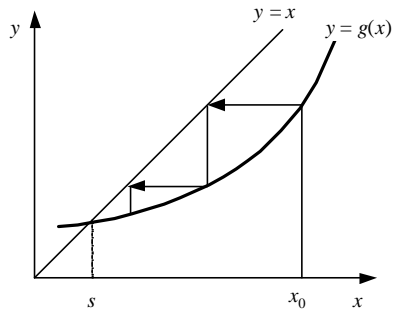
jika  $-1 < g'(x) < 0$  untuk setiap  $x \in I$ , maka lelaran *konvergen bersosilasi*;

jika  $g'(x) > 1$  untuk setiap  $x \in I$ , maka lelaran *divergen monoton*;

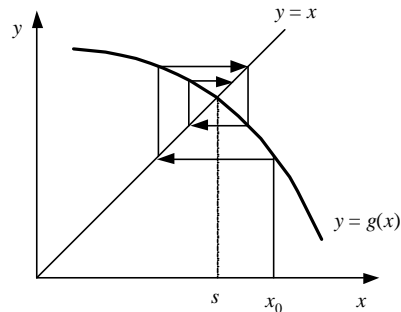
jika  $g'(x) < -1$  untuk setiap  $x \in I$ , maka lelaran *divergn bersosilasi*.

Semuanya dirangkum seperti pada Gambar 3.10..

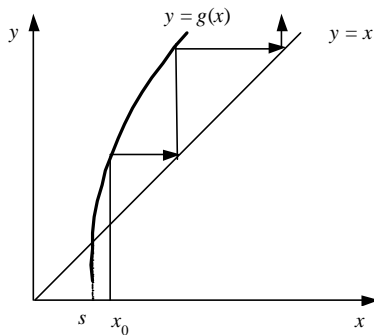
Sebagai catatan, keadaan  $|g'(x)| = 1$  tidak didefinisikan. Catat juga bahwa semakin dekat nilai  $|g'(x)|$  ke nol di dekat akar, semakin cepat kekonvergenan metode lelaran titik-tetap ini [PUR84].



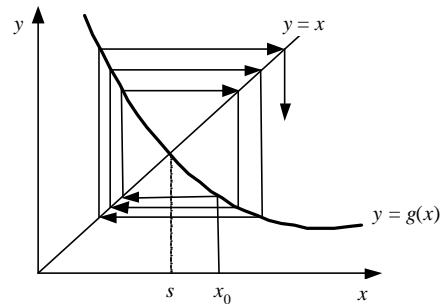
(a) Konvergen monoton:  $0 < g'(x) < 1$



(b) Konvergen berosilasi:  $-1 < g'(x) < 0$



(c) Divergen monoton:  $g'(x) > 1$



(d) Divergen berosilasi:  $g'(x) < -1$

**Gambar 3.10** Jenis-jenis kekonvergenan

Sekarang, mari kita analisis mengapa pencarian akar persamaan  $x^2 - 2x - 3 = 0$  pada Contoh 3.2 dan pencarian akar persamaan  $x^3 + 6x - 3 = 0$  pada Contoh 3.3 dengan bermacam-macam prosedur lelaran dan tebakan awal kadang-kadang konvergen dan kadang-kadang divergen.

(i) Prosedur lelaran pertama:  $x_{r+1} = \sqrt{2x_r + 3}$

$$g(x) = \sqrt{2x + 3}$$

$$g'(x) = \frac{1}{2\sqrt{2x+3}}$$

Terlihat bahwa  $|g'(x)| < 1$  untuk  $x$  di sekitar titik-tetap  $s = 3$ . Karena itu, pengambilan tebakan awal  $x_0 = 4$  akan menghasilkan lelaran yang konvergen sebab  $|g'(4)| = |1/[2\sqrt{8+3}]| = 0.1508 < 1$ .

(ii) Prosedur lelaran kedua:  $x_{r+1} = 3/(x_r - 2)$

$$g(x) = 3/(x-2)$$

$$g'(x) = -3/(x-2)^2$$

Terlihat bahwa  $|g'(x)| < 1$  untuk  $x$  di sekitar titik-tetap  $s = 3$ . Karena itu, pengambilan tebakan awal  $x_0 = 4$  akan menghasilkan lelaran yang konvergen sebab

$$|g'(4)| = |-3/(4-2)^2| = 0.75 < 1.$$

(iii) Prosedur lelaran ketiga  $x_{r+1} = (x_r^2 - 3)/2$

$$g(x) = (x^2 - 3)/2$$

$$g'(x) = x$$

Terlihat bahwa  $|g'(x)| > 1$  untuk  $x$  di sekitar titik-tetap  $s = 3$ . Karena itu, pengambilan tebakan awal  $x_0 = 4$  akan menghasilkan lelaran yang divergen sebab

$$|g'(4)| = |4| = 4 > 1.$$

(iv) Prosedur lelaran pada Contoh 3.3:  $x_{r+1} = (-x_r^3 + 3)/6$

$$g(x) = (-x^3 + 3)/6$$

$$g'(x) = -x^2/2$$

Terlihat bahwa  $|g'(x)| < 1$  untuk  $x$  di sekitar titik-tetap  $s = 0.48$ . Pemilihan  $x_0 = 0.5$  akan menjamin lelaran konvergen sebab  $|g'(x_0)| < 1$ . Untuk  $x_0 = 1.5$  dan  $x_0 = 2.2$  memang nilai  $|g'(x_0)| > 1$  tetapi lelarannya masih tetap konvergen, namun  $x_0 = 2.7$  terlalu jauh dari titik-tetap sehingga lelarannya divergen. Dapatkah kita menentukan batas-batas selang yang menjamin prosedur lelaran akan konvergen di dalamnya? Temukan jawabannya pada Contoh 3.4 di bawah ini.

### Contoh 3.4

Pada Contoh 3.3 di atas, tentukan selang sehingga prosedur lelaran

$$x_{r+1} = (-x_r^3 + 3)/6$$

konvergen?

### Penyelesaian:

$$g(x) = (-x^3 + 3)/6$$

$$g'(x) = -x^2/2$$

Syarat konvergen adalah  $|g'(x)| < 1$ . Jadi,

$$\Leftrightarrow |-x^2/2| < 1$$

$$\Leftrightarrow -1 < -x^2/2 < 1$$

$$\Leftrightarrow 2 > x^2 > -2$$

$$\Leftrightarrow -2 < x^2 < 2$$

Urai satu per satu:

(i)  $x^2 > -2$  { tidak ada  $x$  yang memenuhi)

(ii)  $x^2 < 2$ , dipenuhi oleh

$$\Leftrightarrow x^2 - 2 < 0$$

$$\Leftrightarrow -\sqrt{2} < x < \sqrt{2}$$

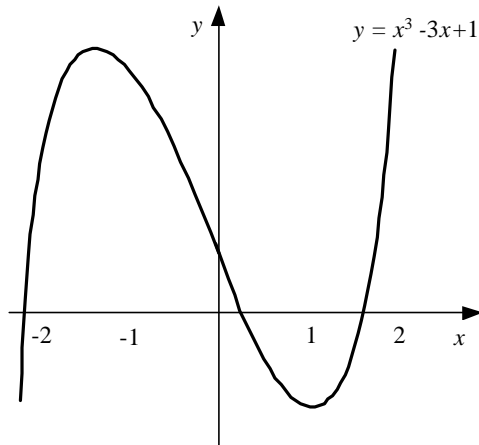
Jadi, prosedur lelaran  $x_{r+1} = (-x_r^3 + 3)/6$  konvergen di dalam selang  $-\sqrt{2} < x < \sqrt{2}$ . Kita dapat memilih  $x_0$  dalam selang tersebut yang menjamin lelaran akan konvergen. ■

### Contoh 3.5

Gunakan metode lelaran titik-tetap untuk mencari akar persamaan

$$x^3 - 3x + 1 \text{ dalam selang } [1, 2] \quad [\text{PUR84}]$$

**Catatan :** selang  $[1, 2]$  ini sebenarnya tidak digunakan dalam proses lelaran sebagaimana halnya pada metode bagidua. Selang ini diberikan untuk memastikan bahwa suatu prosedur lelaran titik-tetap konvergen di dalamnya. Kurva fungsi  $y = x^3 - 3x + 1$  diperlihatkan pada Gambar 3.11.



**Gambar 3.11** Kurva  $y = x^3 - 3x + 1$

**Penyelesaian:**

(i)  $x_{r+1} = (x_r^3 + 1)/3$

Tetapi, karena  $|g'(x)| = |x^2| > 1$  dalam selang  $[1, 2]$ , maka prosedur lelaran ini tidak digunakan.

(ii)  $x_{r+1} = -1/(x_r^2 - 3)$

Tetapi, karena  $|g'(x)| = |2x/(x^2 - 3)^3| > 1$  dalam selang  $[1, 2]$ , maka prosedur lelaran ini tidak digunakan.

(iii)  $x_{r+1} = 3/x_r - 1/x_r^2$

Ternyata  $|g'(x)| = |(-3x + 2)/x^3| \leq 1$  di dalam selang  $[1, 2]$ , yaitu,  $g'(x)$  naik dari  $g'(1) = -1$  ke  $g'(2) = -1/2$ . Jadi,  $|g'(x)|$  lebih kecil dari 1 dalam selang  $[1, 2]$ . Dengan mengambil  $x = 1.5$ , prosedur lelarannya konvergen ke akar  $x = 1.5320889$  seperti pada tabel berikut ini.

$r$	$x$
0	1.5
1	1.5555556
2	1.5153061
...	...
43	1.5320888
44	1.5320889
45	1.5320889

■

Contoh 3.5 menunjukkan bahwa ada dua hal yang mempengaruhi kekonvergenan prosedur lelaran:

1. Bentuk formula  $x_{r+1} = g(x_r)$
2. Pemilihan tebakan awal  $x$

**Catatan:** Meskipun  $|g'(x)| > 1$  menyatakan lelaran divergen, tetapi kita harus hati-hati dengan pernyataan ini. Sebabnya, walaupun  $x_r$  divergen dari suatu akar, runtunan lelarannya mungkin konvergen ke akar yang lain. Kasus seperti ini ditunjukkan pada Contoh 3.6 di bawah ini.

### Contoh 3.6

Tentukan akar persamaan  $f(x) = x^2 - 4x + 3 = 0$  dengan prosedur lelaran

$$x_{r+1} = (x_r^2 + 3)/4$$

#### Penyelesaian:

Jika prosedur lelaran  $x_{r+1} = (x_r^2 + 3)/4$  konvergen ke titik-tetap  $s$ , maka

$$\begin{aligned} \text{limit } x_r &= s \\ r &\rightarrow \infty \end{aligned}$$

sehingga

$$\begin{aligned} s &= (s^2 + 3)/4 \\ s^2 - 4s + 3 &= 0 \\ (s - 3)(s - 1) &= 0 \end{aligned}$$

yang memberikan  $s_1 = 1$  atau  $s_2 = 3$ . Jadi, lelaran konvergen ke akar  $x = 1$  atau akar  $x = 3$ . Dari

$$g(x) = (x^2 + 3)/4$$

diperoleh

$$g'(x) = x/2$$

Gambarkan kurva  $y = x$  dan  $y = (x^2 + 3)/4$  seperti pada Gambar 3.12.

Prosedur lelaran akan konvergen bila

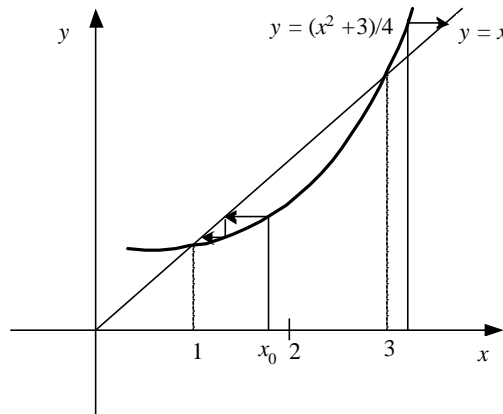
$$\begin{aligned} &\Leftrightarrow |g'(x)| < 1 \\ &\Leftrightarrow -1 < x/2 < 1 \end{aligned}$$

atau

$$-2 < x < 2$$



Sehingga pemilihan  $x_0$  dalam selang  $-2 < x < 2$  menjamin lelaran konvergen ke akar  $x = 1$ . Dari Gambar 3.12 terlihat bahwa lelaran juga konvergen ke akar  $x = 1$  untuk pemilihan  $x_0$  dalam selang  $2 < x < 3$ . Padahal, kalau dihitung, dalam selang  $2 < x < 3$ ,  $|g'(x)| > 1$  yang menyatakan bahwa lelarannya divergen. Lelaran divergen dari akar  $x = 3$  tetapi konvergen ke akar  $x = 1$ . ■



**Gambar 3.12** Kurva  $y=x$  dan  $y=(x^2 + 3)/4$

Sebagai contoh terakhir metode lelaran titik-tetap, mari kita hitung akar fungsi pada Contoh 3.1, yaitu  $f(x) = e^x - 5x^2$ .

### Contoh 3.7

Hitunglah akar  $f(x) = e^x - 5x^2$  dengan metode lelaran titik-tetap. Gunakan  $\varepsilon = 0.00001$ . Tebakan awal akar  $x_0=1$ .

#### Penyelesaian:

Salah satu prosedur lelaran yang dapat dibuat adalah

$$\begin{aligned} e^x - 5x^2 &= 0 \\ e^x &= 5x^2 \\ x &= \sqrt{e^x/5} \\ x_{r+1} &= \text{SQRT}(\text{EXP}(x_r)/5) \end{aligned}$$

Tabel lelarannya:

$i$	$x_r$	$ x_{r+1} - x_r $
0	0.500000	-
1	0.574234	0.074234

2	0.595948	0.021714
3	0.602453	0.006506
4	0.604416	0.001963
5	0.605010	0.000593
6	0.605189	0.000180
7	0.605244	0.000054
8	0.605260	0.000016
9	0.605265	0.000005
10	0.605266	0.000002
11	0.605267	0.000000

Hampiran akar  $x = 0.605267$  ■

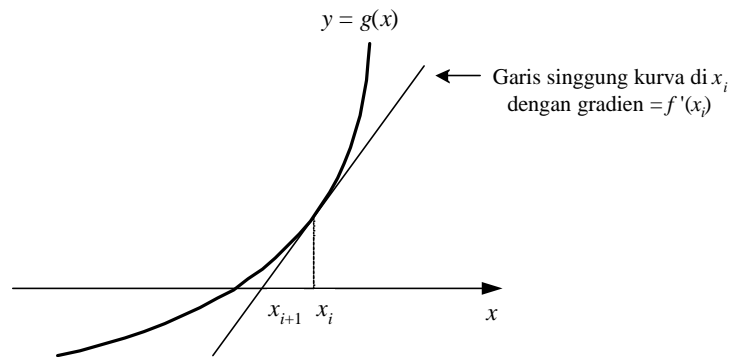
### 3.4.2 Metode Newton-Raphson<sup>3</sup>

Di antara semua metode pencarian akar, metode Newton-Raphsonlah yang paling terkenal dan paling banyak dipakai dalam terapan sains dan rekayasa. Metode ini paling disukai karena konvergensinya paling cepat diantara metode lainnya.

Ada dua pendekatan dalam menurunkan rumus metode Newton-Raphson, yaitu:

- (a) penurunan rumus Newton-Raphson secara geometri,
- (b) penurunan rumus Newton-Raphson dengan bantuan deret Taylor.

#### (a) Penurunan rumus Newton-Raphson secara geometri



**Gambar 3.13** Tafsiran geometri metode Newton-Raphson

<sup>3</sup>Beberapa buku menyebutnya *metode Newton* saja. Joseph Raphson (1648 - 1715) adalah matematikawan Inggris yang mempublikasikan metode Newton.

Dari Gambar 3.13, gradien garis singgung di  $x_r$  adalah

$$m = f'(x_r) = \frac{\Delta y}{\Delta x} = \frac{f(x_r) - 0}{x_r - x_{r+1}} \quad (\text{P.3.8})$$

atau

$$f'(x_r) = \frac{f(x_r)}{x_r - x_{r+1}} \quad (\text{P.3.9})$$

sehingga prosedur lelaran metode Newton-Raphson adalah

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)}, \quad f'(x_r) \neq 0. \quad (\text{P.3.10})$$

**(b) Penurunan rumus Newton-Raphson dengan bantuan deret Taylor**

Uraikan  $f(x_{r+1})$  di sekitar  $x_r$  ke dalam deret Taylor:

$$f(x_{r+1}) \approx f(x_r) + (x_{r+1} - x_r)f'(x_r) + \frac{(x_{r+1} - x_r)^2}{2} f''(t), \quad x_r < t < x_{r+1} \quad (\text{P.3.11})$$

yang bila dipotong sampai suku orde-2 saja menjadi

$$f(x_{r+1}) \approx f(x_r) + (x_{r+1} - x_r)f'(x_r) \quad (\text{P.3.12})$$

dan karena persoalan mencari akar, maka  $f(x_{r+1}) = 0$ , sehingga

$$0 = f(x_r) + (x_{r+1} - x_r)f'(x_r) \quad (\text{P.3.13})$$

atau

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)}, \quad f'(x_r) \neq 0 \quad (\text{P.3.14})$$

yang merupakan rumus metode Newton-Raphson.

Kondisi berhenti lelaran Newton-Raphsin adalah bila

$$|x_{r+1} - x_r| < e$$

atau bila menggunakan galat relatif hampiran

$$\left| \frac{x_{r+1} - x_r}{x_{r+1}} \right| < d$$

dengan **e** dan **d** adalah toleransi galat yang diinginkan.

**Catatan:**

1. Jika terjadi  $f'(x_r) = 0$ , ulang kembali perhitungan lelaran dengan  $x_0$  yang lain.
2. Jika persamaan  $f(x) = 0$  memiliki lebih dari satu akar, pemilihan  $x_0$  yang berbeda-beda dapat menemukan akar yang lain.
3. Dapat pula terjadi lelaran konvergen ke akar yang berbeda dari yang diharapkan (seperti halnya pada metode lelaran titik-tetap).

**Program 3.7** Metode Newton-Raphson

```
procedure Newton_Raphson(x:real);
{ Mencari akar persamaan f(x) = 0 dengan metode Newton-Raphson
  K.Awal : x adalah tebakan awal akar, nilainya sudah terdefinisi
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon = 0.000001;
var
  x_sebelumnya: real;

  function f(x:real):real;
  { mengembalikan nilai f(x). Definisi f(x) bergantung pada persoalan }

  function f_aksen(x:real):real;
  { mengembalikan nilai f'(x). Definisi f'(x) bergantung
    pada persoalan }

begin
  repeat
    x_sebelumnya:=x;
    x:=x - f(x)/f_aksen(x);
  until (ABS(x-x_sebelumnya) < epsilon)

  { x adalah hampiran akar persamaan }
  write('Hampiran akar x = ', x:10:6);

end;
```

**Catatan:** Program 3.7 ini belum menangani kasus pembagian dengan 0 atau  $\approx 0$  dan kasus divergen. Program 3.8 di bawah ini merupakan modifikasi dari Program 3.7 untuk menangani pembagian dengan 0 dan kasus divergen.

**Program 3.8** Metode Newton-Raphson (dengan penanganan kasus divergen dan pembagian dengan 0)

```

procedure Newton_Raphson(x:real);
{ Mencari akar persamaan  $f(x) = 0$  dengan metode Newton-Raphson
  K.Awal : x adalah tebakan awal akar, nilainya sudah terdefinisi
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon1 = 0.000001;      { toleransi galat akar hampiran }
  epsilon2 = 0.000000001;   { toleransi nilai yang hampir 0 }
  Nmaks = 30;               { jumlah maksimum lelaran }
var
  x_sebelumnya: real;
  i : integer;
  berhenti : boolean;      { jika  $f'(x) \ll 0$ , stop ! }

  function f(x:real):real;
  { mengembalikan nilai  $f(x)$ . Definisi  $f(x)$  bergantung pada persoalan }

  function f_aksen(x:real):real;
  { mengembalikan nilai  $f'(x)$ . Definisi  $f'(x)$  bergantung
    pada persoalan }

begin
  i:=0;
  berhenti:=false;
  repeat
    if ABS(f_aksen(x)) < epsilon2 then
      berhenti:=true; { menghindari pembagian bilangan yang » 0 }
    else
      begin
        x_sebelumnya:=x;
        x:=x - f(x)/f_aksen(x);
        i:=i+1;
      end;
    until (ABS(x-x_sebelumnya) < epsilon1) or (berhenti) or (i > Nmaks)

    if berhenti then
      writeln('Pembagian dengan bilangan yang hampir 0')
    else
      if i > Nmaks then
        writeln('Divergen')
      else
        { x adalah hampiran akar persamaan }
        write('Hampiran akar x = ', x:10:6);
      endif
    endif
  end;

```

**Contoh 3.8**

Hitunglah akar  $f(x) = e^x - 5x^2$  dengan metode Newton-Raphson. Gunakan  $\epsilon = 0.00001$ .  
Tebakan awal akar  $x_0 = 1$ .

**Penyelesaian:**

$$f(x) = e^x - 5x^2$$

$$f'(x) = e^x - 10x$$

Prosedur lelaran Newton-Raphson:

$$x_{r+1} = x_r - \frac{e^x - 5x^2}{e^x - 10x}$$

Tebakan awal  $x_0 = 1$

Tabel lelarannya:

$i$	$x_r$	$ x_{r+1} - x_r $
0	0.500000	-
1	0.618976	0.118976
2	0.605444	0.013532
3	0.605267	0.000177
4	0.605267	0.000000

Hampiran akar  $x = 0.605267$  ■

Contoh 3.8 di atas memperlihatkan bahwa metode Newton-Raphson memerlukan sedikit lelaran, dibandingkan dengan metode bagidua, metode regula falsi, dan metode lelaran titik-tetap. Metode Newton-Raphson sangat berguna untuk menghitung fungsi-fungsi dasar, seperti akar bilangan, nilai  $e$ , arcsin ( $x$ ), dan sebagainya. Contoh 3.9 dan Contoh 3.10 memperlihatkan penggunaan metode Newton-Raphson untuk menghitung akar bilangan dan nilai pecahan.

**Contoh 3.9**

Tentukan bagaimana cara menentukan  $\sqrt{c}$  dengan metode Newton-Raphson.

**Penyelesaian:**

Misalkan  $\sqrt{c} = x$ . Kuadratkan kedua ruas sehingga  $c = x^2 \Leftrightarrow x^2 - c = 0$ .

Di sini  $f(x) = x^2 - c$  dan  $f'(x) = 2x$ . Prosedur lelaran Newton-Raphsonnya adalah

$$x_{r+1} = x_r - \frac{x_r^2 - c}{2x_r} = 0.5(x_r + c/x_r)$$

Untuk  $c = 2$ , dengan memilih  $x_0 = 1$  dan  $\varepsilon = 0.000001$ , kita peroleh

$$\begin{aligned}x_1 &= 1.500000 \\x_2 &= 1.416667 \\x_3 &= 1.414216 \\x_4 &= 1.414214\end{aligned}$$

Jadi,  $\sqrt{2} \approx 1.414214$  ■

### Contoh 3.10

Bagaimana menghitung nilai  $1/c$  dengan metode Newton-Raphson?

#### Penyelesaian:

Misalkan  $1/c = x \Leftrightarrow 1/x = c \Leftrightarrow 1/x - c = 0$ . Di sini  $f(x) = 1/x - c$  dan  $f'(x) = -1/x^2$ .

Prosedur Newton-Raphsonnya adalah

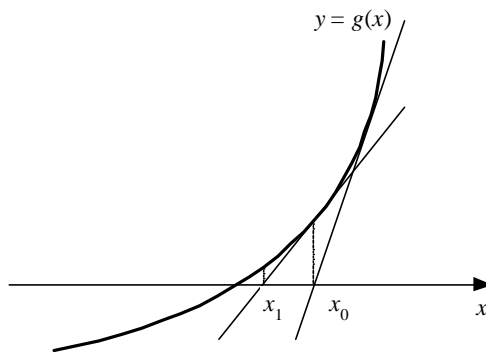
$$x_{r+1} = x_r - \frac{(1/x_r - c)}{-1/x_r^2} = x_r(2 - cx_r)$$

Untuk  $c = 7$ , dengan memilih  $x_0 = 0.2$  dan  $\varepsilon = 0.0000001$ , kita peroleh

$$\begin{aligned}x_1 &= 0.1200000 \\x_2 &= 0.1392000 \\x_3 &= 0.1427635 \\x_4 &= 0.1428570 \\x_5 &= 0.1428751 \\x_6 &= 0.1428571\end{aligned}$$

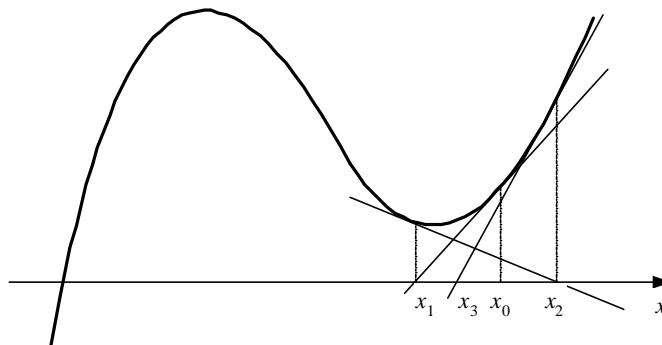
Jadi,  $1/7 \approx 0.1428571$  ■

Secara umum, bila metode Newton-Raphson konvergen, kekonvergenannya itu berlangsung sangat cepat, seperti yang dilukiskan pada Gambar 3.14. Titik potong garis singgung fungsi dengan sumbu- $x$  semakin cepat bergerak mendekati akar sejati.



**Gambar 3.14** Kecepatan konvergensi metode Newton-Raphson sangat cepat

Karena metode Newton-Raphson tergolong metode terbuka, maka dalam beberapa kasus lelarannya mungkin divergen. Bahkan, kalau kurvanya seperti pada Gambar 3.15 serta pemilihan  $x_0$  yang jauh dari akar sejati, lelarannya akan berosilasi di sekitar cekungan lain.



**Gambar 3.15** Lelaran metode Newton-Raphson yang divergen

Membuat grafik fungsi sangat membantu dalam pencarian akar. Grafik fungsi dapat memperlihatkan secara visual lokasi akar sejati. Dengan demikian tebakan awal yang bagus untuk akar dapat diturunkan. Pemilihan tebakan awal sebaiknya cukup dekat dengan akar. Selain itu, kita juga dapat mengetahui apakah fungsi tersebut mempunyai akar tidak. Pada kasus tidak ada akar, lelarannya akan divergen berosilasi.



### Kriteria konvergensi metode Newton-Raphson

Apakah persyaratan agar metode Newton-Raphson konvergen? Tinjau kembali bentuk umum prosedur lelaran metode terbuka,

$$x_{r+1} = g(x_r)$$

Karena metode Newton-Raphson termasuk metode terbuka, maka dalam hal ini,

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

Dengan mengingat syarat perlu agar lelaran konvergen adalah  $|g'(x)| < 1$ , maka

$$\begin{aligned} g'(x) &= 1 - \frac{[f'(x)f'(x) - f(x)f''(x)]}{[f'(x)]^2} \\ &= \frac{f(x)f''(x)}{[f'(x)]^2} \end{aligned} \quad (\text{P.3.18})$$

Karena itu, metode Newton-Raphson akan konvergen bila

$$\left| \frac{f(x)f''(x)}{[f'(x)]^2} \right| < 1$$

dengan syarat  $f'(x) \neq 0$ .

### 3.4.3 Orde Konvergensi Metode Terbuka

Prosedur lelaran pada setiap metode terbuka dapat ditulis dalam bentuk

$$x_{r+1} = g(x_r) \quad (\text{P.3.19})$$

misalnya pada metode Newton-Raphson  $g(x_r) = x_r - f(x_r)/f'(x_r)$ . Misalkan  $x_r$  adalah hampiran terhadap akar sejati  $s$  sehingga  $s = g(s)$ . Maka, berdasarkan konsep galat yang sudah dijelaskan di dalam Bab 2,  $s = x_r + \mathbf{e}_r$  dengan  $\mathbf{e}_r$  adalah galat dari  $x_r$ . Uraikan  $g(s)$  di sekitar  $x_r$ :

$$\begin{aligned} g(s) &= g(x_r) + g'(x_r)(s - x_r) + \frac{1}{2} g''(x_r)(s - x_r)^2 + \dots \\ &= g(x_r) + g'(x_r)\mathbf{e}_r + \frac{1}{2} g''(x_r)\mathbf{e}_r^2 + \dots \end{aligned} \quad (\text{P.3.20})$$

Kurangi persamaan (P.3.20) dengan persamaan (P.3.19):

$$\begin{array}{r}
g(s) = g(x_r) + g'(x_r)e_r + \frac{1}{2} g''(x_r)e_r^2 + \dots \\
- x_{r+1} = g(x_r) \\
\hline
g(s) - x_{r+1} = g'(x_r)e_r + \frac{1}{2} g''(x_r)e_r^2 + \dots
\end{array}$$

Karena  $g(s) = s$ , maka

$$s - x_{r+1} = g'(x_r)e_r + \frac{1}{2} g''(x_r)e_r^2 + \dots$$

Misalkan  $s - x_{r+1} = e_{r+1}$ , sehingga

$$e_{r+1} = g'(x_r)e_r + \frac{1}{2} g''(x_r)e_r^2 + \dots \quad (\text{P.3.21})$$

Bilangan pangkat dari  $e_r$  menunjukkan orde (atau laju) konvergensi prosedur lelaran:

$$(a) \quad e_{r+1} \approx g'(t)e_r, \quad x_r < t < x_{r+1} : \text{prosedur lelaran berorde satu} \quad (\text{P.3.22})$$

$$(b) \quad e_{r+1} \approx \frac{1}{2} g''(t_r)e_r^2, \quad x_r < t < x_{r+1} : \text{prosedur lelaran berorde dua} \quad (\text{P.3.23})$$

Metode Newton-Raphson termasuk ke dalam metode terbuka berorde dua. Pernyataan ini kita buktikan di bawah ini.

### Orde konvergensi metode Newton-Raphson

Pada metode Newton-Raphson,  $g(x_r) = x_r - f(x_r) / f'(x_r)$ . Turunan pertama dari  $g(x_r)$  adalah (dari persamaan P.3.18):

$$g'(x_r) = \frac{f(x_r)f''(x_r)}{[f'(x_r)]^2} \quad (\text{P.3.24})$$

Jika  $x_r$  adalah akar persamaan  $f(x) = 0$ , maka  $f(x_r) = 0$ , sehingga

$$g'(x_r) = 0$$

Ini berarti metode Newton-Raphson paling sedikit berorde dua. Turunan kedua dari  $g(x_r)$  adalah

$$g''(x_r) = f''(x_r) / f'(x_r) \quad (\text{P.3.25})$$

Sulihkan (P.3.25) ke dalam (P.3.23):

$$\mathbf{e}_{r+1} = \frac{f''(x_r) \mathbf{e}_r^2}{2f'(x_r)} \quad (\text{P.3.26})$$

Persamaan (P.3.26) ini mempunyai tiga arti:

1. Galat lelaran sekarang sebanding dengan kuadrat galat lelaran sebelumnya. Jika galat lelaran sekarang misalnya 0.001, maka pada lelaran berikutnya galatnya sebanding dengan 0.000001. Hal inilah yang menyebabkan metode Newton-Raphson sangat cepat menemukan akar (jika lelarannya konvergen).
2. Jumlah angka bena akan berlipat dua pada tiap lelaran. Ini merupakan konsekuensi dari hal nomor 1 di atas.
3. Orde konvergensi metode Newton-Raphson adalah kuadratik. sehingga ia dinamakan juga *metode kuadratik*.

Cara lain untuk menemukan orde konvergensi metode Newton-Raphson adalah dengan meneruskan penurunan rumus Newton-Raphson dari deret Taylornya sebagai berikut. Perhatikan kembali persamaan (P.3.11) di atas. Bila  $x_{r+1} = s$  sehingga  $f(x_{r+1}) = f(s) = 0$ , dalam hal ini  $s$  adalah akar sejati, sulihkan  $s$  ke dalam persamaan (P.3.11) di atas:

$$0 = f(x_r) + (s - x_r)f'(x_r) + \frac{(s - x_r)^2 f''(t)}{2} \quad (\text{P.3.27})$$

Kurangi (P.3.27) dengan (P.3.13):

$$\begin{aligned} 0 &= f(x_r) + (s - x_r)f'(x_r) + \frac{(s - x_r)^2 f''(t)}{2} \\ 0 &= f(x_r) + (x_{r+1} - x_r)f'(x_r) \quad - \\ \hline 0 &= (s - x_{r+1})f'(x_r) + \frac{(s - x_r)^2 f''(t)}{2} \end{aligned} \quad (\text{P.3.28})$$

Misalkan  $s - x_{r+1} = \mathbf{e}_{r+1}$  dan  $s - x_r = \mathbf{e}_r$ , maka persamaan (P.3.28) dapat ditulis menjadi

$$\mathbf{e}_{r+1} f'(x_r) + \frac{\mathbf{e}_r^2 f''(t)}{2} = 0$$

atau

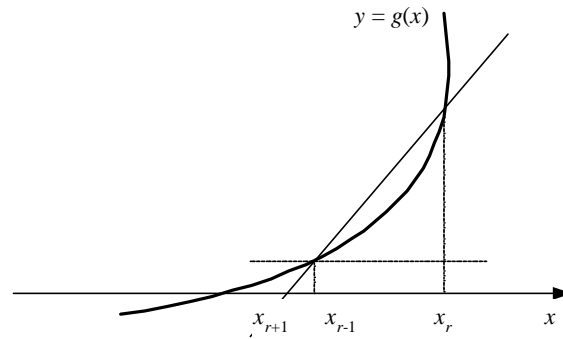
$$e_{r+1} = \frac{f''(t)e_r^2}{2f'(x_r)} \quad (\text{P.3.29})$$

yang sama dengan (P.3.26), kecuali pada  $f''(x_r)$  dan  $f''(t)$ , tetapi perbedaan ini tidak begitu penting, sebab yang dicari adalah pangkat dari  $e_r$ .

Pada proses pencarian akar dengan metode Newton-Raphson, muncul kesulitan jika  $|f'(x)|$  terlalu dekat ke nol, dan kita harus menggunakan bilangan berketelitian ganda untuk memperoleh  $f(x)$  dan  $f'(x)$  cukup teliti [KRE88]. Persamaan nirlanjara  $f(x) = 0$  yang mempunyai kasus seperti ini disebut berkondisi buruk (lihat pembahasan kondisi buruk di dalam Bab 2).

### 3.4.4 Metode Secant

Prosedur lelaran metode Newton-Raphson memerlukan perhitungan turunan fungsi,  $f'(x)$ . Sayangnya, tidak semua fungsi mudah dicari turunannya, terutama fungsi yang bentuknya rumit. Turunan fungsi dapat dihilangkan dengan cara menggantinya dengan bentuk lain yang ekuivalen. Modifikasi metode Newton-Raphson ini dinamakan *metode secant*.



**Gambar 3.16** Metode Secant

Berdasarkan Gambar 3.16, dapat kita hitung gradien

$$f'(x_r) = \frac{\Delta y}{\Delta x} = \frac{AC}{BC} = \frac{f(x_r) - f(x_{r-1})}{x_r - x_{r-1}} \quad (\text{P.3.30})$$

Sulihkan (P.3.30) ke dalam rumus Newton-Raphson:

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)}$$

sehingga diperoleh

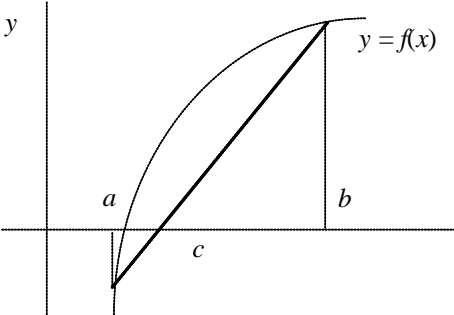
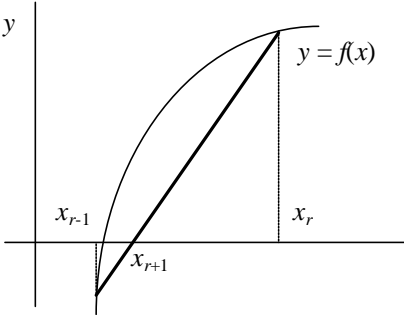
$$x_{r+1} = x_r - \frac{f(x_r)(x_r - x_{r-1})}{f(x_r) - f(x_{r-1})} \quad (\text{P.3.31})$$

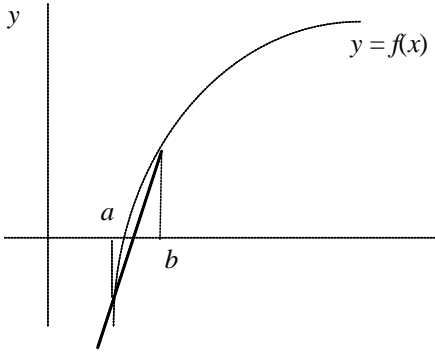
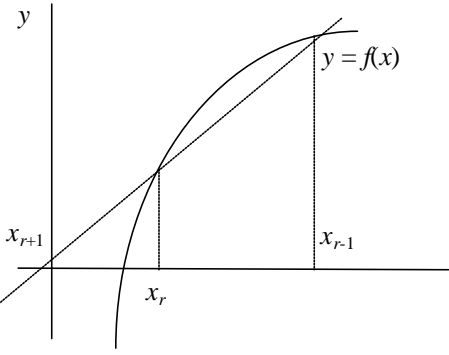
yang merupakan prosedur lelaran metode secant. Dalam hal ini, diperlukan dua buah tebakan awal akar, yaitu  $x_0$  dan  $x_1$ . Kondisi berhenti lelaran adalah bila

$$|x_{r+1} - x_r| < \mathbf{e} \text{ (galat mutlak) atau } \left| \frac{x_{r+1} - x_r}{x_{r+1}} \right| < \mathbf{d} \text{ (galat hampiran)}$$

dengan  $\mathbf{e}$  dan  $\mathbf{d}$  adalah toleransi galat.

Sepintas metode secant mirip dengan metode regula-falsi, namun sesungguhnya prinsip dasar keduanya berbeda, seperti yang dirangkum pada tabel di bawah ini:

Metode Regula Falsi	Metode Secant
1. Diperlukan dua buah nilai awal $a$ dan $b$ (ujung-ujung selang) sedemikian sehingga $f(a)f(b) < 0$ .	1. Diperlukan dua buah nilai awal $x_0$ dan $x_1$ (tebakan awal akar), tetapi <u>tidak harus</u> $f(x_0)f(x_1) < 0$ .
<p>2. <u>Lelaran pertama</u>:</p>  <p>Pada lelaran pertama, tidak ada perbedaan antara regula-falsi dan secant. Perbedaan baru muncul pada lelaran kedua.</p>	<p>2. <u>Lelaran pertama</u>:</p>  <p>Pada lelaran pertama tidak ada perbedaan antara secant dan regula falsi. Perbedaan baru muncul pada lelaran kedua.</p>

<p><u>Lelaran kedua:</u></p>  <p>Perpotongan garis lurus dengan sumbu-<math>x</math> tetap berada di dalam selang yang mengandung akar.</p>	<p><u>Lelaran kedua:</u></p>  <p>Perpotongan garis lurus dengan sumbu-<math>x</math> mungkin menjauhi akar.</p>
<p>3. Berdasarkan nomor 2 di atas, lelarannya <i>selalu</i> konvergen</p>	<p>3. Berdasarkan nomor 2 di atas, lelarannya <i> mungkin </i>divergen.</p>

Program 3.9 berikut berisi algoritma metode *secant*.

**Program 3.9** Metode *Secant*

```

procedure Secant(x0, x1:real);
{ Mencari akar persamaan  $f(x) = 0$  dengan metode secant
  K.Awal :  $x_0$  dan  $x_1$  adalah tebakan awal akar, terdefinisi nilainya
  K.Akhir: akar persamaan tercetak di layar
}
const
  epsilon = 0.000001;      { toleransi galat akar hampiran }
var
  x_sebelumnya: real;

  function f(x:real):real;
  { mengembalikan nilai  $f(x)$ . Definisi  $f(x)$  bergantung pada persoalan }

begin
  repeat
    x_sebelumnya:=x1;
    x:=x-(f(x1)*(x1 - x0)/(f(x1)-f(x0)));
    x0:=x1;
    x1:=x;
  until (ABS(x-x_sebelumnya) < epsilon);
  { x adalah hampiran akar persamaan }
  write('Hampiran akar x = ', x:10:6);
end;

```

**Catatan:** Program 3.9 belum menangani kasus pembagian dengan 0 atau  $\approx 0$  dan kasus divergen. Program harus dimodifikasi untuk menangani pembagian dengan 0 atau  $\approx 0$  dan kasus divergen menjadi Program 3.10 berikut.

**Program 3.10** Perbaikan metode *Secant*

```

procedure Secant(x0, x1:real);
{ Mencari akar persamaan  $f(x) = 0$  dengan metode secant
  K.Awal :  $x_0$  dan  $x_1$  adalah tebakan awal akar, terdefinisi nilainya
  K.Akhir: Hampiran akar tercetak di layar
}
const
  epsilon1 = 0.000001;    { toleransi galat akar hampiran }
  epsilon2 = 0.00000001; { toleransi nilai yang hampir 0 }
  Nmaks = 30;             { jumlah maksimum lelaran }
var
  x_sebelumnya: real;
  berhenti: boolean;
  i : integer;

  function f(x:real):real;
  { mengembalikan nilai  $f(x)$ . Definisi  $f(x)$  bergantung pada persoalan }

begin
  i:=0;
  repeat
    if ABS(f(x1)- f(x0)) < epsilon2 then
      berhenti:=true; { menghindari pembagian bilangan yang » 0 }
    else
      begin
        x_sebelumnya:=x1;
        x:=x-(f(x1)*(x1 - x0)/(f(x1)-f(x0)));
        x0:=x1;
        x1:=x;
        i:=i+1;
      end;
  until (ABS(x-x_sebelumnya) < epsilon1) or (berhenti) or (i > Nmaks);

  if berhenti then
    writeln('Pembagian dengan bilangan yang hampir 0')
  else
    if i > Nmaks then
      writeln('Divergen')
    else
      { x adalah hampiran akar persamaan }
      write('Hampiran akar x = ', x:10:6);
    {endif}
  {endif}
end;

```

**Contoh 3.11**

Hitunglah akar  $f(x) = e^x - 5x^2$  dengan metode secant. Gunakan  $\epsilon = 0.00001$ . Tebakan awal akar  $x_0 = 0.5$  dan  $x_1 = 1$ .

**Penyelesaian:**

Tabel lelarannya:

$i$	$x_r$	$ x_{r+1} - x_r $
0	0.500000	-
1	1.000000	0.500000
3	-0.797042	1.797042
4	10.235035	11.032077
5	-0.795942	11.030977
6	-0.794846	0.001096
7	-0.472759	0.322087
8	-0.400829	0.071930
9	-0.374194	0.026635
10	-0.371501	0.002692
11	-0.371418	0.000083
12	-0.371418	0.000000

Akar  $x = -0.371418$

Ternyata lelarannya mengarah ke akar yang lain, yaitu  $x = -0.371418$  ■

## 3.5 Akar Ganda

Akar ganda (*multiple roots*) terjadi bila kurva fungsi menyinggung sumbu- $x$ , misalnya:

- (i)  $f(x) = x^3 - 5x^2 + 7x - 3 = (x-3)(x-1)(x-1)$  memiliki akar ganda dua di  $x = 1$
- (ii)  $f(x) = x^4 - 6x^3 + 12x^2 - 10x + 3 = (x-3)(x-1)(x-1)(x-1)$  memiliki akar ganda tiga di  $x = 1$ .

Pada pembahasan terdahulu kita telah menyinggung bahwa metode bagidua dan metode tertutup lainnya tidak dapat digunakan untuk mencari akar ganda, sebab fungsi tidak berubah tanda di sekeliling akar. Metode terbuka, seperti metode Newton-Raphson, sebenarnya dapat diterapkan di sini. Tetapi, bila digunakan metode Newton-Raphson untuk mencari akar ganda, kecepatan konvergensi berjalan secara lanjar, tidak lagi kuadratis sebagaimana aslinya. Agar konvergensi metode Newton-Raphson tetap kuadratik untuk akar ganda, maka Ralston dan Rabinowitz mengusulkan alternatif metode Newton-Raphson [CHA91] sebagai berikut:



$$x_{r+1} = x_r - m \frac{f(x_r)}{f'(x_r)} \quad (\text{P.3.32})$$

dengan  $m$  adalah **bilangan multiplisitas** akar, misalnya .

- akar tunggal,  $m = 1$ ,
- akar ganda dua,  $m = 2$ ,
- akar ganda tiga,  $m = 3$ , dan seterusnya.

Namun alternatif ini tidak memuaskan karena kita perlu tahu terlebih dahulu bilangan multiplisitas akar. Disamping itu, untuk  $x$  dekat akar ganda, nilai  $f(x) \approx 0$  dan juga nilai  $f'(x) \approx 0$ , yang dapat mengakibatkan pembagian dengan nol. Pembagian dengan nol ini dapat dihindari dengan melihat fakta bahwa  $f(x)$  lebih dulu nol sebelum  $f'(x)$ . Jadi,

**if**  $f(x) \approx 0$  **then** hentikan lelaran

Ralston dan Rabinowitz mengusulkan alternatif lain [CHA91]. Didefinisikan

$$u(x) = f(x) / f'(x) \quad (\text{P.3.33})$$

(Perhatikan, bentuk  $u(x)$  ini memiliki akar yang sama dengan  $f(x)$ , sebab, jika  $u(x) = 0$  maka  $f(x) = 0$ ).

Selanjutnya,

$$x_{r+1} = x_r - \frac{u(x_r)}{u'(x_r)} \quad (\text{P.3.34})$$

yang dalam hal ini,

$$u'(x) = [f(x)/f'(x)]' = \frac{f'(x)f'(x) - f''(x)f(x)}{[f'(x)]^2} = \frac{[f'(x)]^2 - f''(x)f(x)}{[f'(x)]^2} \quad (\text{P.3.35})$$

sehingga

$$x_{r+1} = x_r - \frac{f(x_r)/f'(x_r)}{\frac{[f'(x_r)]^2 - f''(x_r)f(x_r)}{[f'(x_r)]^2}}$$

atau

$$x_{r+1} = x_r - \frac{f(x_r)f'(x_r)}{[f'(x_r)]^2 - f''(x_r)f(x_r)} \quad (\text{P.3.36})$$

Meskipun rumus (P.3.36) ini lebih disukai untuk akar ganda, namun ia kurang mangkus sebab memerlukan lebih banyak komputasi daripada metode Newton-Raphson yang baku. Rumus (P.3.36) berlaku secara umum, yaitu ia tetap dapat dipakai untuk pencarian akar tidak ganda sekalipun.

Metode secant juga dapat dimodifikasi dengan menyulihkan  $u(x) = f(x)/f'(x)$  ke dalam rumusnya. Rumus yang dihasilkan adalah [CHA91]:

$$x_{r+1} = x_r - \frac{u(x_r)(x_{r-1} - x_r)}{u(x_{r-1}) - u(x_r)} \quad (\text{P.3.37})$$

### Contoh 3.12

Hitung akar  $f(x) = x^3 - 5x^2 + 7x - 3$  dengan metode Newton-Raphson baku dan metode Newton-Raphson yang diperbaiki. Tebakan awal  $x_0 = 0$ . [CHA91]

#### Penyelesaian:

$$\begin{aligned} f(x) &= x^3 - 5x^2 + 7x - 3 \\ f'(x) &= 3x^2 - 10x + 7 \\ f''(x) &= 6x - 10 \end{aligned}$$

Dengan metode Newton-Raphson baku:

$$x_{r+1} = \frac{x_r - \frac{x_r^3 - 5x_r^2 + 7x_r - 3}{3x_r^2 - 10x_r + 7}}{1}$$

Dengan metode Newton-Raphson yang dimodifikasi:

$$x_{r+1} = x_r - \frac{(x_r^3 - 5x_r^2 + 7x_r - 3)(3x_r^2 - 10x_r + 7)}{(3x_r^2 - 10x_r + 7)^2 - (6x_r - 10)(x_r^3 - 5x_r^2 + 7x_r - 3)}$$

Tabel lelarannya adalah:

Metode Newton Raphson baku		Metode Newton Raphson yang dimodifikasi	
$r$	$x_r$	$r$	$x_r$
0	0.000000000	0	0.000000000
1	0.428571429	1	1.105263158
2	0.685714286	2	1.003081664
3	0.832865400	3	1.000002382
4	0.913328983		
5	0.955783293		
6	0.977655101		

Lelaran konvergen ke akar  $x = 1$ . Terlihat dari tabel di atas bahwa metode Newton yang dimodifikasi memiliki jumlah lelaran lebih sedikit. ■

## 3.6 Akar-Akar Polinom

Bentuk baku polinom derajat  $\leq n$  adalah

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \quad (\text{P.3.38})$$

dengan  $a_i$  adalah konstanta riil,  $i = 0, 1, 2, \dots, n$ , dan  $a_n \neq 0$ . Polinom  $p(x)$  memiliki  $n$  buah akar, baik akar nyata maupun akar kompleks. Akar kompleks muncul dalam pasangan konjugasi,  $w = u + vi$  dan  $w = u - vi$ , dengan  $i = \sqrt{-1}$ . Contohnya, polinom  $p(x) = 5 - 4x + x^2$  mempunyai akar  $2 + i$  dan  $2 - i$ .

Semua metode pencarian akar dapat diterapkan pada polinom. Misalnya dengan metode Newton-Raphson,

$$x_{r+1} = x_r - \frac{p(x_r)}{p'(x_r)} \quad (\text{P.3.39})$$

Masalahnya, evaluasi polinom,  $p(x_r)$  dan  $p'(x_r)$  membutuhkan banyak operasi perkalian (termasuk perpangkatan). Semakin tinggi derajat polinomnya tentu semakin banyak operasi perkalian yang diperlukan, yang berarti semakin besar rambatan galat pembulatan (ingat, komputer menggunakan bilangan titik-kambang). Karena itu, harus dicari suatu metode perhitungan polinom dengan sedikit operasi perkalian.

### 3.6.1 Metode Horner untuk Evaluasi Polinom

Menghitung langsung  $p(x)$  untuk  $x = 1$  tidak mangkus sebab melibatkan banyak operasi perkalian. Metode Horner, atau disebut juga metode perkalian bersarang (*nested multiplication*) menyediakan cara perhitungan polinom dengan sedikit operasi perkalian. Dalam hal ini, polinom  $p(x)$  dinyatakan sebagai perkalian bersarang

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \dots + x(a_{n-1} + a_n x)))\dots)) \quad (\text{P.3.40})$$

#### Contoh 3.12

Nyatakan  $p(x) = 8 + 6x + 2x^2 + 5x^3$  dalam bentuk perkalian bersarang.

**Penyelesaian:**

$$\begin{aligned} p(x) &= 8 + 6x + 2x^2 + 5x^3 && (6 \text{ buah perkalian}) \\ &= 8 + x(6 + x(2 + 5x)) && (\text{hanya } 3 \text{ buah perkalian}) \end{aligned}$$

Perhitungan  $p(x)$  untuk  $x = 2$  adalah

$$p(2) = 8 + 2(6 + 2(2 + 5 \cdot 2)) = 68 \quad \blacksquare$$

Metode perkalian bersarang untuk menghitung  $p(t)$  seringkali dinyatakan dalam bentuk tabel Horner berikut:

$t$	$a_n$	$a_{n-1}$	$a_{n-2}$	....	$a_2$	$a_1$	$a_0$
		$tb_n$	$tb_{n-1}$	....	$tb_3$	$tb_2$	$tb_1$

$$b_n = a_n \quad b_{n-1} = a_{n-1} + tb_n \quad b_{n-2} = a_{n-2} + tb_{n-1} \quad b_2 = a_2 + tb_3 \quad b_1 = a_1 + tb_2 \quad b_0 = a_0 + tb_1$$

polinom sisa

Hasil evaluasi:  $p(t) = b_0$

Jadi, untuk Contoh 3.12 di atas,

2	5	2	6	8
		10	24	60
	5	12	30	$68 = p(2)$

dan menghasilkan polinom sisa  $5x^2 + 12x + 30$ .

**Program 3.11** Menghitung  $p(x)$  untuk  $x = t$  dengan metode Horner

```
{ Dalam program utama telah didefinisikan:
  const n=...; {derajat polinom}
  var a, b, c: array[1..n] of real
}

function p(t:real):real;
{ menghitung p(t) dengan metode Horner}
var
  k: integer;
begin
  b[n]:=a[n];
  for k:=n-1 downto 0 do
    b[k]:=a[k] + b[k+1]*t;
  {end for}
  p:=b[0];
end;
```

### 3.6.2 Pencarian Akar-akar Polinom

Proses perhitungan  $p(x)$  untuk  $x = t$  dengan menggunakan metode Horner sering dinamakan *pembagian sintetis*  $p(x):(x - t)$ , menghasilkan  $q(x)$  dan sisa  $b_0$ ,

$$\left[ \frac{p(x)}{(x-t)} = q(x) \right] + \text{sisa } b_0 \quad (\text{P.3.41})$$

atau

$$p(x) = b_0 + (x-t) q(x) \quad (\text{P.3.42})$$

yang dalam hal ini,

$$q(x) = b_n x^{n-1} + b_{n-1} x^{n-2} + \dots + b_3 x^2 + b_2 x + b_1 \quad (\text{P.3.43})$$

Untuk Contoh 3.12 di atas,

$$p(x) = 8 + 6x + 2x^2 + 5x^3 = 68 + (x-2) (5x^2 + 12x + 30)$$

Jika  $t$  adalah hampiran akar polinom  $p(x)$  maka

$$p(t) = b_0 + (t - t) q(t) = b_0 + 0 = b_0$$

(Perhatikan, jika  $t$  akar sejati, maka  $b_0 = 0$ )

Akar-akar lain dari  $p(x)$  dapat dicari dari polinom  $q(x)$  sebab setiap akar  $q(x)$  juga adalah akar  $p(x)$ . Proses reduksi polinom ini disebut *deflasi* (*deflation*). Koefisien-koefisien  $q(x)$ , yaitu  $b_n, b_{n-1}, \dots, b_3, b_2, b_1$  dapat ditemukan langsung dari tabel Horner,

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + tb_n \\ b_{n-2} &= a_{n-2} + tb_{n-1} \\ &\dots \\ b_2 &= a_2 + tb_3 \\ b_1 &= a_1 + tb_2 \end{aligned}$$

Algoritmanya,

```

b[n]:=a[n];
for k:=n-1 downto 1 do
    b[k]:=a[k] + t*b[k+1]
{endfor}

```

Misalkan akar polinom dihitung dengan metode Newton-Raphson,

$$x_{r+1} = x_r - \frac{p(x_r)}{p'(x_r)}$$

maka proses pencarian akar secara deflasi dapat dirumuskan dalam langkah 1 sampai 4 berikut ini.

### **Langkah 1:**

Menghitung  $p(x_r)$  dapat dilakukan secara mangkus dengan metode Horner.

Misalkan  $t = x_r$  adalah hampiran akar polinom  $p(x)$ ,

$$p(x) = b_0 + (x - x_r) q(x)$$

Perhitungan  $p(x_r)$  menghasilkan

$$p(x_r) = b_0 + (x_r - x_r) q(x_r) = b_0$$

Nilai  $p(x_r) = b_0$  ini dapat dihitung dengan function p

**Langkah 2:**

Menghitung  $p'(x_r)$  secara mangkus:

Misalkan  $t = x_r$  adalah hampiran akar polinom  $p(x)$ ,

$$p(x) = b_0 + (x - x_r) q(x)$$

Turunan dari  $p$  adalah

$$p'(x) = 0 + 1 \cdot q(x) + (x - x_r) q'(x) = q(x) + (x - x_r) q'(x)$$

sehingga

$$p'(x_r) = q(x_r) + (x_r - x_r) q'(x_r) = q(x_r)$$

Koefisien polinom  $q(x)$  dapat ditentukan dari langkah 1. Selanjutnya  $q(x_r)$  dapat dihitung dengan function  $q$  berikut:

**Program 3.12** Menghitung  $p'(t) = q(t)$

```
function q(t:real):real;
{ menghitung p'(t)=q(t) dengan metode Horner}
var
  k : integer;
begin
  c[n]:=b[n];
  for k:=n-1 downto 1 do
    c[k]:=b[k] + t*c[k+1]
  {endfor}
  q:=c[1];
end;
```

**Langkah 3:**

$$x_{r+1} = x_r - \frac{p(x_r)}{p'(x_r)}$$

**Langkah 4:**

Ulangi langkah 1, 2 dan 3 di atas sampai  $|x_{r+1} - x_r| < \epsilon$ .

**Program 3.13** Prosedur Newton-Raphson untuk menghitung akar polinom

```
procedure Newton_Raphson_untuk_polinom(n:integer; x:real);
{ procedure Newton-Raphson untuk menghitung akar polinom p(x) yang
  berderajat n dengan tebakan awal akar x
  K.Awal : n adalah derajat polinom; x adalah tebakan awal akar;
           kedua nilai sudah terdefinisi
  K.Akhir: Hampiran akar polinom tercetak di layar.
}
const
  epsilon = 0.0000001;
var
  x_sebelumnya: real;

  function p(t:real):real;
  {menghitung p(t) dengan metode Horner}
  var
    k: integer;
  begin
    b[n]:=a[n];
    for k:=n-1 downto 0 do
      b[k]:=a[k] + b[k+1]*t;
    {end for}
    p:=b[0];
  end {p};

  function q(t:real):real;
  { menghitung p'(t)=q(t) dengan metode Horner}
  var
    k : integer;
  begin
    c[n]:=b[n];
    for k:=n-1 downto 1 do
      c[k]:=b[k] + t*c[k+1]
    {end for}
    q:=c[1];
  end {q} ;

begin
  repeat
    x_sebelumnya:=x;
    x:=x - p(x)/q(x);
  until ABS(x - x_sebelumnya) < epsilon;

  { x adalah akar polinom }
  writeln('Hampiran akar = ', x:10:6);
end;
```



Program 3.13 ini hanya menemukan satu buah akar polinom. Untuk mencari seluruh akar nyata polinom, harus dilakukan proses deflasi. Setelah akar pertama  $x_1$  diperoleh, polinom  $p(x)$  dapat ditulis sebagai

$$p(x) = (x - x_1) q(x) + b_0$$

yang dalam hal ini  $q(x) = b_n x^{n-1} + b_{n-1} x^{n-2} + \dots + b_3 x^2 + b_2 x + b_1$ . Koefisien-koefisien  $q(x)$ , yaitu  $b_n, b_{n-1}, \dots, b_3, b_2, b_1$  diperoleh di akhir Program 3.11, yang telah tersimpan pada elemen larik  $b[n-1], b[n-2], \dots, b[2], b[1]$ . Selanjutnya panggil Program 3.13 untuk mencari akar polinom  $q(x)$  yang berderajat  $n-1$  dengan tebakan awalnya dapat digunakan  $x_1$  (atau boleh bilangan lain). Setelah akar kedua  $x_2$  diperoleh, polinom  $p(x)$  dapat ditulis sebagai

$$q(x) = (x - x_2) r(x) + b_1$$

yang dalam hal ini

$$r(x) = b_{n-1} x^{n-2} + b_{n-2} x^{n-3} + \dots + b_3 x^2 + b_2 x + b_1.$$

$r(x)$  adalah polinom derajat  $n-2$  dengan koefisien  $b_{n-1}, b_{n-2}, \dots, b_3, b_2, b_1$  diperoleh di akhir Program 3.11 pada elemen larik  $b[n-1], b[n-2], \dots, b[2], b[1]$ . Selanjutnya panggil kembali Program 3.13 untuk mencari akar polinom  $r(x)$  yang berderajat  $n-2$  dengan tebakan awalnya dapat digunakan  $x_2$ . Begitu seterusnya sampai polinom sisa yang ditemukan berderajat 0. Atau, dapat juga sampai polinom sisa berderajat dua.

Algoritma selengkapnya adalah:

```
write('Tebakan awal untuk akar pertama: '); readln(x);
repeat
  Newton_Raphson_untuk_polinom(n, x);
  { salin koefisien b[n], b[n-1], ..., b[1] ke dalam
    a[n-1], a[n-2], ..., a[0] untuk pencarian akar selanjutnya}
  for i:=n downto 1 do
    a[i-1]:=b[i];
  {endfor}
  n:=n-1; { derajat polinom sisa berkurang satu }
until n=0;
```

**Contoh 3.14**

[GER85] Temukan seluruh akar nyata polinom

$$p(x) = x^5 - 12x^4 - 293x^3 + 3444x^2 + 20884x - 240240$$

dengan tebakan awal akar  $x_0 = 11$ .

**Penyelesaian:**

Panggil prosedur

```
Newton_Raphson_untuk_polinom(5, 11);
```

untuk mencari akar polinom  $p(x)$  berderajat 5 dengan tebakan awal akar  $x_0 = 11$ .

Diperoleh akar pertama, yaitu  $x_1 = 13.99990$

$$\text{Deflasi} \rightarrow p(x) = (x - x_1) q(x) + b_0$$

yang dalam hal ini  $q(x) = x^4 + 1999895x^3 - 2650015x^2 - 2659927x + 17160.13$

Panggil prosedur

```
Newton_Raphson_untuk_polinom(4, 13.99990);
```

untuk mencari akar polinom  $q(x)$  berderajat 4 dengan tebakan awal akar  $x_0 = 13.99990$

Diperoleh akar kedua  $x_2 = 12.00016$

$$\text{Deflasi} \rightarrow q(x) = (x - x_2) r(x) + b_1$$

yang dalam hal ini  $r(x) = x^3 + 1400005x^2 - 9699867x - 1429992$

Panggil prosedur

```
Newton_Raphson_untuk_polinom(3, 12.00016);
```

untuk mencari akar polinom  $r(x)$  berderajat 3 dengan tebakan awal akar  $x_0 = 12.00016$

Diperoleh akar  $x_3 = 9.999949$

$$\text{Deflasi} \rightarrow r(x) = (x - x_3) s(x) + b_2$$

yang dalam hal ini  $s(x) = x^2 + 2396998x + 1429999$

Demikian seterusnya sampai kita temukan akar keempat dan akar kelima sebagai berikut:

$$x_4 = -12.99991$$

$$x_5 = -11.00006$$

■

### 3.6.3 Lokasi Akar Polinom

Metode Newton-Raphson memerlukan tebakan awal akar. Bagaimanakah menemukan tebakan awal akar yang bagus untuk polinom? Misalkan akar-akar diberi indeks dan diurut menaik sedemikian sehingga

$$|x_1| \leq |x_2| \leq |x_3| \leq \dots \leq |x_n|$$

Tebakan awal untuk akar terkecil  $x_1$  menggunakan hampiran

$$\begin{aligned} a_0 + a_1 x &\approx 0 \\ x &\approx -a_0/a_1 \end{aligned} \quad (\text{P.3.44})$$

yang dapat dijadikan sebagai tebakan awal untuk menemukan  $x_1$

Tebakan awal untuk akar terbesar  $x_n$  menggunakan hampiran

$$\begin{aligned} a_{n-1}x^{n-1} + a_n x^n &\approx 0 \\ x &\approx -a_{n-1}/a_n \end{aligned} \quad (\text{P.3.45})$$

yang dapat dijadikan sebagai tebakan awal untuk menemukan  $x_n$

#### **Contoh 3.15**

[NOB72] Tentukan tebakan awal untuk mencari akar polinom  $x^2 - 200x + 1 = 0$ .

#### **Penyelesaian:**

Tebakan awal untuk akar terkecil adalah

$$x_0 = -1/(-200) = 1/200$$

Tebakan awal untuk akar terbesar adalah

$$x_0 = -(-200)/1 = 200$$

■

## 3.7 Sistem Persamaan Nirlanjar

Di dalam dunia nyata, umumnya model matematika muncul dalam bentuk sistem persamaan. Persamaan yang diselesaikan tidak hanya satu, tetapi dapat lebih dari satu, sehingga membentuk sebuah sistem yang disebut sistem persamaan nirlanjar. Bentuk umum sistem persamaan nirlanjar dapat ditulis sebagai berikut:

$$\begin{aligned}
f_1(x_1, x_2, \dots, x_n) &= 0 \\
f_2(x_1, x_2, \dots, x_n) &= 0 \\
&\dots \\
f_n(x_1, x_2, \dots, x_n) &= 0
\end{aligned} \tag{P.3.46}$$

Penyelesaian sistem ini adalah himpunan nilai  $x$  simultan,  $x_1, x_2, \dots, x_n$ , yang memenuhi seluruh persamaan. Sistem persamaan dapat diselesaikan secara berlelar dengan metode lelaran titik-tetap atau dengan metode Newton-Raphson.

### 3.7.1 Metode Lelaran Titik-Tetap

Prosedur lelarannya titik-tetap untuk sistem dengan dua persamaan nirlanjar:

$$\begin{aligned}
x_{r+1} &= g_1(x_r, y_r) \\
y_{r+1} &= g_2(x_r, y_r) \\
r &= 0, 1, 2, \dots
\end{aligned} \tag{P.3.47}$$

Metode lelaran titik-tetap seperti ini dinamakan metode **lelaran Jacobi**. Kondisi berhenti (konvergen) adalah

$$|x_{r+1} - x_r| < \epsilon \text{ dan } |y_{r+1} - y_r| < \epsilon$$

Kecepatan konvergensi lelaran titik-tetap ini dapat ditingkatkan. Nilai  $x_{r+1}$  yang baru dihitung langsung dipakai untuk menghitung  $y_{r+1}$ . Jadi,

$$\begin{aligned}
x_{r+1} &= g_1(x_r, y_r) \\
y_{r+1} &= g_2(x_{r+1}, y_r) \\
r &= 0, 1, 2, \dots
\end{aligned} \tag{P.3.48}$$

Metode lelaran titik-tetap seperti ini dinamakan metode **lelaran Seidel**. Kondisi berhenti (konvergen) adalah

$$|x_{r+1} - x_r| < \epsilon \text{ dan } |y_{r+1} - y_r| < \epsilon$$

Untuk fungsi dengan tiga persamaan nirlanjar, lelaran Seidel-nya adalah

$$\begin{aligned}
x_{r+1} &= g_1(x_r, y_r, z_r) \\
y_{r+1} &= g_2(x_{r+1}, y_r, z_r) \\
z_{r+1} &= g_3(x_{r+1}, y_{r+1}, z_r) \\
r &= 0, 1, 2, \dots
\end{aligned} \tag{P.3.49}$$

Kondisi berhenti (konvergen) adalah

$$|x_{r+1} - x_r| < \epsilon \text{ dan } |y_{r+1} - y_r| < \epsilon \text{ dan } |z_{r+1} - z_r| < \epsilon$$

**Contoh 3.16**

[CHA91] Selesaikan sistem persamaan nirlanjar berikut ini,

$$f_1(x, y) = x^2 + xy - 10 = 0$$

$$f_2(x, y) = y + 3xy^2 - 57 = 0$$

(Akar sejatinya adalah  $x = 2$  dan  $y = 3$ )

**Penyelesaian:**

Prosedur lelaran titik-tetapnya adalah

$$x_{r+1} = \frac{10 - x_r^2}{y_r}$$

$$y_{r+1} = 57 - 3x_{r+1}y_r^2$$

Berikan tebakan awal  $x_0 = 1.5$  dan  $y_0 = 3.5$  dan  $\epsilon = 0.000001$

Tabel lelarannya:

r	x	y	$ x_{r+1} - x_r $	$ y_{r+1} - y_r $
0	1.500000	3.500000	-	-
1	2.214286	-24.375000	0.714286	27.875000
2	-0.209105	429.713648	2.423391	454.088648
3	0.023170	-12778.041781	0.232275	13207.755429
...				

Ternyata lelarannya divergen!

Sekarang kita ubah persamaan prosedur lelarannya menjadi

$$x_{r+1} = \sqrt{10 - x_r y_r}$$

$$y_{r+1} = \sqrt{\frac{57 - y_r}{3x_{r+1}}}$$

Tebakan awal  $x_0 = 1.5$  dan  $y_0 = 3.5$  dan  $\epsilon = 0.000001$

Hasilnya,

r	x	y	$ x_{r+1} - x_r $	$ y_{r+1} - y_r $
0	1.500000	3.500000	-	-
1	2.179449	2.860506	0.679449	0.639494
2	1.940534	3.049551	0.238916	0.189045
3	2.020456	2.983405	0.079922	0.066146

4	1.993028	3.005704	0.027428	0.022300
5	2.002385	2.998054	0.009357	0.007650
6	1.999185	3.000666	0.003200	0.002611
7	2.000279	2.999773	0.001094	0.000893
8	1.999905	3.000078	0.000374	0.000305
9	2.000033	2.999973	0.000128	0.000104
10	1.999989	3.000009	0.000044	0.000036
11	2.000004	2.999997	0.000015	0.000012
12	1.999999	3.000001	0.000005	0.000004
13	2.000000	3.000000	0.000002	0.000001
14	2.000000	3.000000	0.000001	0.000000

Akar       $x = 2.000000$   
               $y = 3.000000$



Contoh 3.15 ini memperlihatkan bahwa konvergensi metode lelaran titik-tetap sangat bergantung pada bentuk persamaan prosedur lelaran dan tebakan awal. Syarat perlu kekonvergenan untuk sistem dengan dua persamaan nirlanjar adalah

$$\left| \frac{\partial g_1}{\partial x} \right| + \left| \frac{\partial g_1}{\partial y} \right| < 1$$

dan

$$\left| \frac{\partial g_2}{\partial x} \right| + \left| \frac{\partial g_2}{\partial y} \right| < 1$$

di dalam selang yang mengandung titik tetap  $(p, q)$ .

### 3.7.2 Metode Newton-Raphson

Ingatlah kembali bahwa metode Newton-Raphson dapat diturunkan dari deret Taylor,

$$f(x_{r+1}) \approx f(x_r) + (x_{r+1} - x_r) f'(x_r)$$

dan karena persoalan mencari akar, maka  $f(x_{r+1}) = 0$ , sehingga

$$0 = f(x_r) + (x_{r+1} - x_r) f'(x_r)$$

atau

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)}, \quad f'(x_r) \neq 0$$

Untuk fungsi dengan dua peubah, deret Taylor orde pertama dapat dituliskan untuk masing-masing persamaan sebagai

$$u_{r+1} = u_r + (x_{r+1} - x_r) \frac{\partial u_r}{\partial x} + (y_{r+1} - y_r) \frac{\partial u_r}{\partial y} \quad (\text{P.3.50})$$

dan

$$v_{r+1} = v_r + (x_{r+1} - x_r) \frac{\partial v_r}{\partial x} + (y_{r+1} - y_r) \frac{\partial v_r}{\partial y} \quad (\text{P.3.51})$$

Karena persoalan mencari akar, maka  $u_{r+1} = 0$  dan  $v_{r+1} = 0$ , untuk memberikan

$$\frac{\partial u_r}{\partial x} x_{r+1} + \frac{\partial u_r}{\partial y} y_{r+1} = -u_r + x_r \frac{\partial u_r}{\partial x} + y_r \frac{\partial u_r}{\partial y}$$

$$\frac{\partial v_r}{\partial x} x_{r+1} + \frac{\partial v_r}{\partial y} y_{r+1} = -v_r + x_r \frac{\partial v_r}{\partial x} + y_r \frac{\partial v_r}{\partial y}$$

Dengan sedikit manipulasi aljabar, kedua persamaan terakhir ini dapat dipecahkan menjadi

$$x_{r+1} = x_r - \frac{u_r \frac{\partial v_r}{\partial y} + v_r \frac{\partial u_r}{\partial y}}{\frac{\partial u_r}{\partial x} \frac{\partial v_r}{\partial y} - \frac{\partial u_r}{\partial y} \frac{\partial v_r}{\partial x}} \quad (\text{P.3.52})$$

dan

$$y_{r+1} = y_r + \frac{u_r \frac{\partial v_r}{\partial x} - v_r \frac{\partial u_r}{\partial x}}{\frac{\partial u_r}{\partial x} \frac{\partial v_r}{\partial y} - \frac{\partial u_r}{\partial y} \frac{\partial v_r}{\partial x}} \quad (\text{P.4.53})$$

Penyebut dari masing-masing persamaan ini diacu sebagai **determinan Jacobi** dari sistem tersebut [CHA91]. Metode Newton-Raphson dapat dirampatkan (*generalization*) untuk sistem dengan  $n$  persamaan.

**Contoh 3.17**

[CHA91] Gunakan metode Newton-Raphson untuk mencari akar

$$\begin{aligned}f_1(x, y) = u &= x^2 + xy - 10 = 0 \\f_2(x, y) = v &= y + 3xy^2 - 57 = 0\end{aligned}$$

dengan tebakan awal  $x_0=1.5$  dan  $y_0=3.5$

**Penyelesaian:**

$$\frac{\partial u_o}{\partial x} = 2x + y = 2(1.5) + 3.5 = 6.5$$

$$\frac{\partial u_o}{\partial y} = x = 1.5$$

$$\frac{\partial v_o}{\partial x} = 3y^2 = 3(3.5)^2 = 36.75$$

$$\frac{\partial v_o}{\partial y} = 1 + 6xy = 1 + 6(1.5) = 32.5$$

Determinan Jacobi untuk lelaran pertama adalah

$$6.5(32.5) - 1.5(36.75) = 156.125$$

Nilai-nilai fungsi dapat dihitung dari tebakan awal sebagai

$$\begin{aligned}u_0 &= (1.5)^2 + 1.5(3.5) - 10 = -2.5 \\v_0 &= (3.5)^2 + 3(1.5)(3.5)^2 - 57 = 1.625\end{aligned}$$

Nilai  $x$  dan  $y$  pada lelaran pertama adalah

$$x_0 = \frac{1.5 - (-2.5)(32.5) - 1.625(1.5)}{156.125} = 2.03603$$

dan

$$y_0 = \frac{3.5 - (-2.5)(36.75) - 1.625(6.5)}{156.125} = 2.84388$$

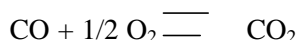
Apabila lelarannya diteruskan, ia konvergen ke akar sejati  $x = 2$  dan  $y = 3$ . ■

Seperti halnya metode lelaran titik-tetap, metode Newton-Raphson mungkin saja divergen jika tebakan awal tidak cukup dekat ke akar. Penggambaran kurva masing-masing persamaan secara grafik dapat membantu pemilihan tebakan awal yang bagus.

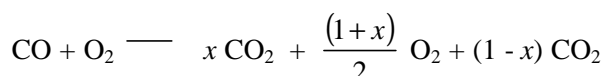


### 3.8 Contoh Soal Terapan

Dalam suatu proses Teknik Kimia, campuran karbon monoksida dan oksigen mencapai kesetimbangan pada suhu 300° K dan tekanan 5 atm. Reaksi teoritisnya adalah



Reaksi kimia yang sebenarnya terjadi dapat ditulis sebagai



Persamaan kesetimbangan kimia untuk menentukan fraksi mol CO yang tersisa, yaitu  $x$ , ditulis sebagai

$$K_p = \frac{(1-x)(3+x)^{1/2}}{x(x+1)^{1/2} p^{1/2}}, \quad 0 < x < 1$$

yang dalam hal ini,  $K_p = 3.06$  adalah tetapan kesetimbangan untuk reaksi  $\text{CO} + 1/2 \text{O}_2$  pada 3000° K dan  $P = 5$  atm. Tentukan nilai  $x$  dengan metode regula falsi yang diperbaiki.

#### Penyelesaian:

Persoalan ini memang lebih tepat diselesaikan dengan metode tertutup karena  $x$  adalah fraksi mol yang nilainya terletak antara 0 dan 1.

Fungsi yang akan dicari akarnya dapat ditulis sebagai

$$f(x) = \frac{(1-x)(3+x)^{1/2}}{x(x+1)^{1/2} p^{1/2}} - K_p, \quad 0 < x < 1$$

dengan  $K_p = 3.06$  dan  $P = 5$  atm.

Selang yang mengandung akar adalah  $[0.1, 0.9]$ . Nilai fungsi di ujung-ujung selang adalah

$$f(0.1) = 3.696815 \text{ dan } f(0.9) = -2.988809$$

yang memenuhi  $f(0.1)f(0.9) < 0$ .

Tabel lelarannya adalah:

$r$	$a$	$c$	$b$	$f(a)$	$f(c)$	$f(b)$	Selang baru	Lebarnya
0	0.100000	0.542360	0.900000	3.696815	-2.488120	-2.988809	[a, c]	0.442360
1	0.100000	0.288552	0.542360	1.848407	-1.298490	-2.488120	[a, c]	0.188552
2	0.100000	0.178401	0.288552	0.924204	0.322490	-1.298490	[c, b]	0.110151
3	0.178401	0.200315	0.288552	0.322490	-0.144794	-1.298490	[a, c]	0.021914
4	0.178401	0.193525	0.200315	0.322490	-0.011477	-0.144794	[a, c]	0.015124
5	0.178401	0.192520	0.193525	0.161242	0.009064	-0.011477	[c, b]	0.001005
6	0.192520	0.192963	0.193525	0.009064	-0.000027	-0.011477	[a, c]	0.000443
7	0.192520	0.192962	0.192963	0.009064	-0.000000	-0.000027	[a, c]	0.000442

Hampiran akar  $x = 0.192962$

Jadi, setelah reaksi berlangsung, fraksi mol CO yang tersisa adalah 0.192962.

Hal-hal kecil membentuk kesempurnaan,  
tetapi kesempurnaan bukanlah hal yang kecil.  
(Michael Angello)

## Soal Latihan

1. Tahun 1225 Leonardo da Pisa mencari akar persamaan

$$f(x) = x^3 + 2x^2 + 10x - 20 = 0$$

dan menemukan  $x = 1.368808107$ . Tidak seorang pun yang mengetahui cara Leonardo menemukan nilai ini. Sekarang, rahasia itu dapat dipecahkan dengan metode lelaran titik-tetap. Bentuklah semua kemungkinan prosedur lelaran titik-tetap dari  $f(x) = 0$ , lalu dengan memberikan sembarang tebakan awal (misalnya  $x_0 = 1$ ), tentukan prosedur lelaran mana yang menghasilkan akar persamaan yang ditemukan Leonardo itu.

2. Apa yang terjadi jika persamaan  $x^2 = 2$  diatur sebagai  $x_{r+1} = 2/x_r$  dan metode lelaran titik-tetap digunakan untuk menemukan akar kuadrat dari 2?
3. Tentukan titik potong kurva  $f(x) = e^{-x}$  dengan kurva  $g(x) = \sin(x)$  dengan metode Newton-Raphson.

4. Tentukan selang sehingga prosedur lelaran  $x_{r+1} = x_r/2 - \cos(2x_r)$  konvergen di dalam selang itu ( $x$  dalam radian)
5. Perlihatkan bahwa semua akar  $x^{20} - 1 = 0$  berkondisi baik.
6. Gunakan metode
  - (i) bagidua
  - (ii) regula-falsi

untuk menemukan akar persamaan Leonardo dalam selang  $[1, 1.5]$ , dan juga dengan metode

- (iii) Newton-Raphson,  $x_0 = 1$
- (iv) secant,  $x_0=1, x_1=1.5$

Untuk semua metode,  $\varepsilon = 10^{-6}$

7. Diketahui lingkaran  $x^2 + y^2 = 2$  dan hiperbola  $x^2 - y^2 = 1$ . Tentukan titik potong kedua kurva dengan metode lelaran titik-tetap (Soal ini adalah mencari solusi sistem persamaan nirlanjar).
8. Diberikan prosedur lelaran  $x_{r+1} = 0.9x_r$  dan nilai awal  $x_0 = 1$ . Menurut orang matematik, untuk  $r = 1, 2, 3, \dots$

$$x_r = (0.9)x_{r-1} = (0.9)(0.9)x_{r-2} = \dots = (0.9)^r x_0 = (0.9)^r$$

dan menyimpulkan bahwa

$$\lim_{r \rightarrow \infty} x_r = 0$$

Bagi orang numerik, harus ditanyakan terlebih dahulu berapa banyak angka bena yang digunakan.

Berapakah

$$\lim_{r \rightarrow \infty} x_r$$

menurut orang numerik bila digunakan:

- (a) satu angka bena
- (b) dua angka bena

(Petunjuk: lakukan sejumlah lelaran tanpa memprogramnya dengan komputer. Setiap perhitungan harus taat asas dengan jumlah angka bena yang digunakan)

9. Misalkan metode bagidua akan digunakan untuk menemukan akar dalam selang  $[-1,5]$ . Berapa kali selang harus dibagi dua untuk menjamin bahwa hampiran  $c_r$  mempunyai ketelitian  $0.5 \times 10^{-9}$ .
10. Dapatkah metode Newton-Raphson digunakan untuk memecahkan:
- (i)  $f(x) = 0$  jika  $f(x) = x^{1/3}$
  - (ii)  $f(x) = 0$  jika  $f(x) = (x-3)^{1/2}$  dan tebakan awal  $x_0 = 4$ ?

Mengapa?

11. Bagaimana bentuk prosedur lelaran Newton-Raphson untuk menghitung  $e$  (bilangan natural?)
12. Nilai arcsin 1 dapat dihitung dari persamaan

$$\sin(x) = 1 \quad \Leftrightarrow \quad \sin(x) - 1 = 0$$

Persamaan ini mempunyai akar ganda dua di  $x = \pi/2$  (periksa!). Tentukan akar ganda itu dengan:

- (a) metode Newton-Raphson baku
  - (b) metode Newton-Raphson dengan faktor multiplisitas akar
  - (c) metode Newton-Raphson yang dimodifikasi untuk menangani kasus akar ganda
13. Gunakan metode Newton-Raphson untuk menghitung  $(47)^{1/4}$  sampai enam angka bena.
14. Perhatikan bahwa bial metode Newton-Raphson diterapkan pada bermacam-macam  $f(x)$  di bawah ini, maka prosedur lelarannya akan mengarah pada pencarian  $\sqrt[a]{a}$ , untuk  $a > 0$ .
- (i)  $f(x) = x^2 - a$
  - (ii)  $f(x) = 1 - a/x^2$
15. Misalkan  $f(x) = \cos(x)$
- (a) Tentukan prosedur lelaran Newton-Raphsonnya

- (b) Jika kita ingin menghitung akar  $x = 3\pi/2$ , dapatkah kita gunakan tebakan awal  $x_0 = 3$ ? Mengapa?
- (c) Seperti (b), bagaimana jika  $x_0 = 5$ ? Mengapa?

# Bab 4

## Solusi Sistem Persamaan Linjar

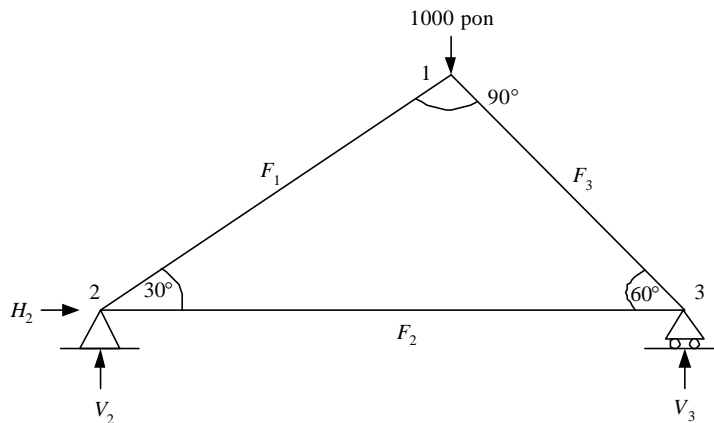
---

Saya tidak dapat memastikan bahwa perubahan akan memperbaiki sesuatu, tetapi saya dapat memastikan bahwa untuk menjadi lebih baik sesuatu harus berubah  
(George C. Lichtenberg)

Dalam praktek rekayasa, perilaku sistem dimodelkan dalam persamaan matematika. Seringkali jumlah persamaan tersebut lebih dari satu dan harus diselesaikan secara serempak atau simultan. Di dalam Bab 3 sudah diberikan contoh penyelesaian sistem dengan dua buah persamaan nirlinjar. Jika sistem persamaan yang dihasilkan berbentuk aljabar linjar (*linier*), maka diperlukan teknik penyelesaian yang lain. Contoh di bawah ini memberi gambaran sistem persamaan linjar dalam bidang rekayasa sipil [CHA91].

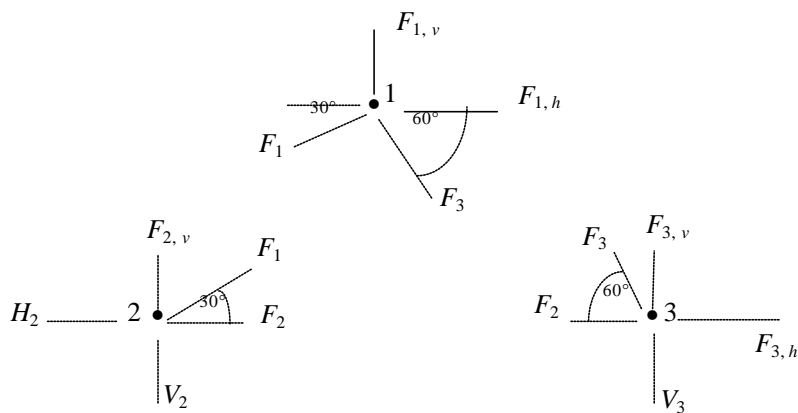
Misalkan seorang insinyur Teknik Sipil merancang sebuah rangka statis yang berbentuk segitiga (Gambar 4.1). Ujung segitiga yang bersudut  $30^\circ$  bertumpu pada sebuah penyangga statis, sedangkan ujung segitiga yang lain bertumpu pada penyangga beroda.

Rangka mendapat gaya eksternal sebesar 1000 pon. Gaya ini disebar ke seluruh bagian rangka. Gaya  $F$  menyatakan tegangan atau kompresi pada anggota rangka. Reaksi eksternal ( $H_2$ ,  $V_2$ , dan  $V_3$ ) adalah gaya yang mencirikan bagaimana rangka berinteraksi dengan permukaan pendukung. Engsel pada simpul 2 dapat menjangkitkan gaya mendatar dan tegak pada permukaan, sedangkan gelinding pada simpul 3 hanya menjangkitkan gaya tegak.



**Gambar 4.1** Gaya-gaya pada rangka statis tertentu

Struktur jenis ini dapat diuraikan sebagai sistem persamaan aljabar linier simultan. Diagram gaya-benda-bebas diperlihatkan untuk tiap simpul dalam Gambar 4.2.



**Gambar 4.2** Diagram gaya-benda-bebas untuk simpul-simpul rangka statis

Menurut hukum Newton, resultan gaya dalam arah mendatar maupun tegak harus nol pada tiap simpul, karena sistem dalam keadaan diam (statis). Oleh karena itu, untuk simpul 1,

$$\begin{aligned}\sum F_H &= 0 = -F_1 \cos 30^\circ + F_3 \cos 60^\circ + F_{1,h} \\ \sum F_V &= 0 = -F_1 \sin 30^\circ - F_3 \sin 60^\circ + F_{1,v}\end{aligned}$$

untuk simpul 2,

$$\begin{aligned}\sum F_H = 0 &= F_2 + F_1 \cos 30^\circ + F_{2,h} + H_2 \\ \sum F_V = 0 &= F_1 \sin 30^\circ - F_{2,v} + V_2\end{aligned}$$

dan untuk simpul 3,

$$\begin{aligned}\sum F_H = 0 &= -F_2 - F_3 \cos 60^\circ + F_{3,h} \\ \sum F_V = 0 &= F_3 \sin 60^\circ + F_{3,v} + V_3\end{aligned}$$

Gaya 1000 pon ke bawah pada simpul 1 berpadanan dengan  $F_{1,v} = -1000$ , sedangkan semua  $F_{i,v}$  dan  $F_{i,h}$  lainnya adalah nol. Persoalan rangka statis ini dapat dituliskan sebagai sistem yang disusun oleh enam persamaan linier dengan 6 peubah yang tidak diketahui:

$$\begin{aligned}\sum F_H = 0 &= -F_1 \cos 30^\circ + F_3 \cos 60^\circ + F_{1,h} = -0.866F_1 + 0.5 F_3 \\ \sum F_V = 0 &= -F_1 \sin 30^\circ - F_3 \sin 60^\circ + F_{1,v} = -0.5F_1 - 0.866 F_3 + 1000 \\ \sum F_H = 0 &= F_2 + F_1 \cos 30^\circ + F_{2,h} + H_2 = F_2 + 0.866F_1 + 0 + H_2 \\ \sum F_V = 0 &= F_1 \sin 30^\circ - F_{2,v} + V_2 = 0.5 F_1 + V_2 \\ \sum F_H = 0 &= -F_2 - F_3 \cos 60^\circ + F_{3,h} = -F_2 - 0.5 F_3 \\ \sum F_V = 0 &= F_3 \sin 60^\circ + F_{3,v} + V_3 = 0.866 F_3 + V_3\end{aligned}$$

Keenam persamaan di atas ditulis ulang kembali dalam susunan yang teratur berdasarkan urutan peubah  $F_1, F_2, F_3, H_2, V_2, V_3$ :

$$\begin{array}{ccccccccc} -0.866F_1 & & & + 0.5 F_3 & & & & & = 0 \\ & -0.5F_1 & & & - 0.866 F_3 & & & & = -1000 \\ -0.866F_1 & - F_2 & & & & & - H_2 & & = 0 \\ & -0.5 F_1 & & & & & & - V_2 & = 0 \\ & & - F_2 & - 0.5 F_3 & & & & & = 0 \\ & & & -0.866 F_3 & & & & - V_3 & = 0 \end{array}$$

atau dalam bentuk matriks:

$$\begin{bmatrix} 0.866 & 0 & -0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 & 0 & 0 \\ -0.866 & -1 & 0 & -1 & 0 & 0 \\ -0.5 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & -0.866 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ H_2 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1000 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Masalah yang ditanyakan adalah nilai  $F_1, F_2, F_3, H_2, V_2$ , dan  $V_3$  yang memenuhi keenam persamaan tersebut secara simultan. Metode penyelesaian sistem persamaan linier seperti di atas merupakan pokok bahasan Bab 4 ini.

## 4.1 Bentuk Umum Sistem Persamaan Linier

Sistem persamaan linier (SPL) dengan dengan  $n$  peubah dinyatakan sebagai

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n \end{aligned} \quad (\text{P.4.1})$$

Dengan menggunakan perkalian matriks, kita dapat menulis (P.4.1) sebagai persamaan matriks

$$Ax = b \quad (\text{P.4.2})$$

yang dalam hal ini,

$A = [a_{ij}]$  adalah matriks berukuran  $n \times n$

$x = [x_j]$  adalah matriks berukuran  $n \times 1$

$b = [b_j]$  adalah matriks berukuran  $n \times 1$  (disebut juga vektor kolom)

yaitu

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Solusi (P.4.1) adalah himpunan nilai  $x_1, x_2, \dots, x_n$  yang memenuhi  $n$  buah persamaan. Metode penyelesaian sistem persamaan linier dengan determinan (aturan Cramer) tidak praktis untuk sistem yang besar. Beberapa metode penyelesaian praktis sistem persamaan linier yang kita bahas di sini adalah:

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Metode dekomposisi  $LU$

5. Metode lelaran Jacobi
6. Metode lelaran Gauss-Seidel.

Walaupun metode penyelesaian SPL beragam, namun sebagian besar metode tersebut, terutama metode 1 sampai 4, tetap didasarkan kepada metode yang paling dasar, yaitu eliminasi Gauss. Metode eliminasi Gauss-Jordan, metode matriks balikan, dan metode dekomposisi *LU* merupakan bentuk variasi lain dari metode eliminasi Gauss. Sedangkan metode lelaran Jacobi dan metode lelaran Gauss-Seidel dikembangkan dari gagasan metode lelaran pada solusi persamaan nirlanjar.

## 4.2 Metode Eliminasi Gauss

Metode ini berangkat dari kenyataan bahwa bila matriks *A* berbentuk *segitiga atas* seperti sistem persamaan berikut ini

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & & & & \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

maka solusinya dapat dihitung dengan **teknik penyulihan mundur** (*backward substitution*):

$$\begin{aligned} a_{nn}x_n &= b_n \quad \rightarrow \quad x_n = b_n/a_{nn} \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &= b_{n-1} \quad \rightarrow \quad x_{n-1} = \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}} \\ a_{n-2,n-2}x_{n-2} + a_{n-2,n-1}x_{n-1} + a_{n-2,n}x_n &= b_{n-2} \quad \rightarrow \quad x_{n-2} = \frac{b_{n-2} - a_{n-2,n-1}x_{n-1} - a_{n-2,n}x_n}{a_{n-2,n-2}} \\ &\vdots \\ \text{dst.} \end{aligned}$$

Sekali  $x_n, x_{n-1}, x_{n-2}, \dots, x_{k+1}$  diketahui, maka nilai  $x_k$  dapat dihitung dengan

$$x_k = \frac{b_k - \sum_{j=k+1}^n a_{kj}x_j}{a_{kk}}, \quad k = n-1, n-2, \dots, 1 \text{ dan } a_{kk} \neq 0. \quad \text{P.4.3)}$$

Kondisi  $a_{kk} \neq 0$  sangat penting, sebab bila  $a_{kk} = 0$ , persamaan (P.4.3) mengerjakan pembagian dengan nol. Apabila kondisi tersebut tidak dipenuhi, maka SPL tidak mempunyai jawaban.

Di dalam Bab 4 ini, kita menggunakan struktur data matriks untuk semua algoritma yang dijelaskan nanti. Pendeklarasiannya adalah sebagai berikut ini:

```
(* KAMUS GLOBAL *)
const
  n = ... ; { ukuran matriks A }
type
  matriks = array[1..n, 1..n] of real;
  vektor = array[1..n] of real;
var
  { larik/matriks yang digunakan untuk sistem  $Ax = b$  }
  A : matriks;
  b : vektor;
  x : vektor;
```

Program 4.1 berikut berisi algoritma penyulihan mundur.

**Program 4.1** Penyulihan Mundur

```
procedure Sulih_Mundur(A : matriks; b : vektor; n: integer;
  var x : vektor);
{ Menghitung solusi sistem persamaan linier yang sudah berbentuk matriks
  segitiga atas
  K.Awal : A adalah matriks yang berukuran  $n \times n$ , elemennya sudah terdefinisi
            harganya; b adalah vektor kolom yang berukuran  $n \times 1$ .
  K.Akhir: x berisi solusi sistem persamaan linier.
}
var
  j, k: integer;
  sigma: real;
begin
  x[n]:=b[n]/a[n,n];
  for k:=n-1 downto 1 do begin
    sigma:=0;
    for j:=k+1 to n do
      sigma:=sigma + a[k, j] * x[j];
    {endfor}
    x[k]:= (b[k] - sigma )/a[k, k];
  end;
end;
```

**Contoh 4.1**

[MAT92] Selesaikan sistem persamaan linier berikut dengan teknik penyulihan mundur

$$\begin{aligned} 4x_1 - x_2 + 2x_3 + 3x_4 &= 20 \\ -2x_2 + 7x_3 - 4x_4 &= -7 \\ 6x_3 + 5x_4 &= 4 \\ 3x_4 &= 6 \end{aligned}$$

**Penyelesaian:**

$$\begin{aligned} x_4 &= 6/3 = 2 \\ x_3 &= \frac{(4 - 5(2))}{6} = -1 \\ x_2 &= \frac{-7 - 7(-1) + 4(2)}{-2} = -4 \\ x_1 &= \frac{20 + 1(-4) - 2(-1) - 3(2)}{4} = 3 \end{aligned}$$

Jadi, solusinya adalah  $x = (3, -4, -1, 2)^T$ . ■

Metode eliminasi Gauss pada prinsipnya bertujuan mentransformasi sistem  $Ax = b$  menjadi sistem

$$Ux = y \quad (\text{P.4.4})$$

dengan  $U$  adalah matriks *segitiga atas*. Selanjutnya solusi  $x$  dapat dihitung dengan teknik penyulihan mundur. Contohnya pada sistem dengan 4 persamaan linier berikut (Elemen matriks  $A$  dan vektor kolom  $b$  disatukan dalam bentuk satu bentuk matriks):

$$\begin{array}{c} \left[ \begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & b_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & b_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & b_4 \end{array} \right] \xrightarrow{\substack{\text{dieliminasi} \\ \text{menjadi } [U, y]}} \left[ \begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} & b_3^{(2)} \\ 0 & 0 & 0 & a_{44}^{(3)} & b_4^{(3)} \end{array} \right] \\ [A, b] \qquad \qquad \qquad [U, y] \end{array}$$

Tanda pangkat (1), (2), (3) menunjukkan bahwa elemen matriks  $A$  telah berubah satu kali, dua kali, dan tiga kali.

Proses eliminasi terdiri atas tiga operasi baris elementer:

1. *Pertukaran* : Urutan dua persamaan dapat ditukar karena pertukaran tersebut tidak mempengaruhi solusi akhir.
2. *Penskalaan* : Persamaan dapat dikali dengan konstanta bukan nol, karena perkalian tersebut tidak mempengaruhi solusi akhir.
3. *Penggantian* : Persamaan dapat diganti dengan penjumlahan persamaan itu dengan  $m$  kali persamaan lain. Misalnya persamaan diganti dengan selisih persamaan itu dengan dua kali persamaan lain; yaitu

$$\text{baris}_r := \text{baris}_r - m_{p,r} \text{baris}_p \quad (\text{P.4.5})$$

Nilai  $a_{r,r}$  pada posisi  $(r, r)$  yang digunakan untuk mengeliminasi  $x_r$  pada baris  $r + 1, r + 2, \dots, N$  dinamakan elemen *pivot* dan persamaan pada baris ke- $r$  disebut **persamaan pivot** [MAT92]. Ada kemungkinan *pivot* bernilai nol sehingga pembagian dengan nol tidak dapat dilakukan. Tata-ancang eliminasi yang tidak mempedulikan nilai *pivot* adalah tataancang yang naif (*naive*) atau sederhana. Metode eliminasi Gauss seperti ini dinamakan **metode eliminasi Gauss naif** (*naive Gaussian elimination*), karena metodenya tidak melakukan pemeriksaan kemungkinan pembagian dengan nol. Pada metode eliminasi Gauss naif tidak ada operasi pertukaran baris dalam rangka menghindari *pivot* yang bernilai nol itu.

PIVOT: Critical, cardinal, or crucial factor  
(Kamus Webster)

#### Contoh 4.2

Selesaikan sistem persamaan linier dengan metode eliminasi Gauss naif:

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &= 5 \\ 4x_1 + 4x_2 - 3x_3 &= 3 \\ -2x_1 + 3x_2 - x_3 &= 1 \end{aligned}$$

**Penyelesaian:**

$$\left[ \begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{array} \right] \xrightarrow{\substack{R_2 - 2R_1 \\ R_3 + R_1}} \left[ \begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{array} \right] \xrightarrow{R_3 - 3R_2} \left[ \begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & -5 & -15 \end{array} \right]$$

- Keterangan: (i) elemen yang dicetak tebal menyatakan *pivot*.  
(ii) simbol “~” menyatakan operasi baris elementer.  
(iii)  $R_i$  menyatakan baris (*row*) ke- $i$

- (iv)  $R_2 - \frac{4}{2}R_1$  artinya elemen-elemen pada baris kedua dikurangi dengan dua kali elemen-elemen pada baris ke satu.

$$\begin{array}{rcll} R_2 & : & 4 & 4 & -3 & 3 \\ 2R_1 & : & 4 & 6 & -2 & 10 & - \\ \hline R_2 - \frac{4}{2}R_1 & : & 0 & -2 & -1 & -7 & \text{(menjadi elemen baris ke-2)} \end{array}$$

Solusi sistem diperoleh dengan teknik penyulihan mundur sebagai berikut:

$$\begin{aligned} -5x_3 &= -15 \rightarrow x_3 = 3 \\ -2x_2 - x_3 &= -7 \rightarrow x_2 = (-7 + 3)/-2 = 2 \\ 2x_1 + 3x_2 - x_3 &= 5 \rightarrow x_1 = (5 + 3 - 6)/2 = 1 \end{aligned}$$

Jadi, solusinya adalah  $x = (1, 2, 3)^T$  ■

#### Program 4.2 Metode Eliminasi Gauss Naif

```

procedure Eliminasi_Gauss_Naif(A : matriks; b : vektor; n:integer;
                                var x : vektor);
{ Menghitung solusi sistem persamaan linier Ax = b
  K.Awal : A adalah matriks yang berukuran n ´ n, elemennya sudah terdefi-
            nisi harganya; b adalah vektor kolom yang berukuran n ´ 1
  K.Akhir: x berisi solusi sistem
}
var
  i; k, j : integer;
  m: real;
begin
  for k:=1 to n-1 do {mulai dari baris pivot 1 sampai baris pivot n-1}
    begin
      for i:=(k+1) to n do {eliminasi mulai dari baris k+1 sampai baris n}
        begin
          m:=a[i,k]/a[k,k]; {hitung faktor pengali}
          for j:=k to n do {eliminasi elemen dari kolom k sampai kolom n}
            a[i,j]:=a[i,j] - m*a[k,j];
          {endfor}
          b[i]:=b[i] - m*b[k]; {eliminasi elemen vektor b pada baris i}
        end;
      end;
    end;
  Sulih_Mundur(A, b, n, x); {dapatkan solusinya dengan teknik penyulihan
                              mundur}
end;

```

#### Kelemahan eliminasi Gauss naif

Jika  $pivot a_{pp} = 0$ , baris ke- $k$  tidak dapat digunakan untuk mengeliminasi elemen pada kolom  $p$ , karena terjadinya pembagian dengan nol. Oleh karena itu, *pivot* yang bernilai nol harus dihindari dengan tata-ancang (*strategy*) *pivoting*.

### 4.2.1 Tata-ancang *Pivoting*

Prinsip tata-ancang *pivoting* adalah sebagai berikut: jika  $a_{p,p}^{(p-1)} = 0$ , cari baris  $k$  dengan  $a_{k,p} \neq 0$  dan  $k > p$ , lalu pertukarkan baris  $p$  dan baris  $k$ . Metode eliminasi Gauss dengan tata-ancang *pivoting* disebut metode eliminasi Gauss yang diperbaiki (*modified Gaussian elimination*).

#### Contoh 4.3

Selesaikan sistem persamaan linier berikut dengan metode eliminasi Gauss yang menerapkan tata-ancang *pivoting*.

$$\begin{aligned} x_1 + 2x_2 + x_3 &= 2 \\ 3x_1 + 6x_2 &= 9 \\ 2x_1 + 8x_2 + 4x_3 &= 6 \end{aligned}$$

$$\left[ \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 3 & 6 & 0 & 9 \\ 2 & 8 & 4 & 6 \end{array} \right] \xrightarrow[\text{operasi baris 1}]{\begin{array}{l} R_2 - 3/1 R_1 \\ R_3 - 2/1 R_1 \end{array}} \left[ \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 0 & -3 & 3 \\ 0 & 4 & 2 & 2 \end{array} \right] \xrightarrow[\text{operasi baris 2}]{\begin{array}{l} R_3 \Leftrightarrow R_2 \\ (*) \end{array}} \left[ \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 4 & 2 & 2 \\ 0 & 0 & -3 & 3 \end{array} \right]$$

Setelah operasi baris 1, elemen  $a_{22}$  yang akan menjadi *pivot* pada operasi baris 2 ternyata sama dengan nol. Karena itu, pada operasi baris 2, elemen baris 2 dipertukarkan dengan elemen baris 3. Tanda (\*) menyatakan pertukaran baris terjadi akibat proses *pivoting*. Sekarang elemen  $a_{22} = 4 \neq 0$  sehingga operasi baris elementer dapat diteruskan. Tetapi, karena matriks  $A$  sudah membentuk matriks  $U$ , proses eliminasi selesai. Solusinya diperoleh dengan teknik penyulutan mundur, yaitu  $x_3 = -1$ ,  $x_2 = 1$ , dan  $x_1 = 1$ . ■

Melakukan pertukaran baris untuk menghindari *pivot* yang bernilai nol adalah cara *pivoting* yang sederhana (*simple pivoting*). Masalah lain dapat juga timbul bila elemen *pivot* sangat dekat ke nol, karena jika elemen *pivot* sangat kecil dibandingkan terhadap elemen lainnya, maka galat pembulatan dapat muncul [CHA91]. Ingatlah kembali bahwa kita bekerja dengan mesin (komputer) yang beroperasi dengan pembulatan bilangan riil. Jadi, disamping menghindari pembagian dengan nol, tata-ancang *pivoting* dapat juga diperluas untuk mengurangi galat pembulatan.

Ada dua macam tatancang *pivoting*:

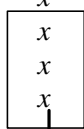
1. *Pivoting* sebagian (*partial pivoting*)

Pada tatancang *pivoting* sebagian, *pivot* dipilih dari semua elemen pada kolom  $p$  yang mempunyai nilai mutlak terbesar,

$$|a_{k,p}| = \max\{|a_{p,p}|, |a_{p+1,p}|, \dots, |a_{n-1,p}|, |a_{n,p}|\}$$

lalu pertukarkan baris ke- $k$  dengan baris ke- $p$ . Misalkan setelah operasi baris pertama diperoleh matriksnya seperti yang digambarkan pada matriks di bawah ini. Untuk operasi baris kedua, carilah elemen  $x$  pada kolom kedua, dimulai dari baris ke-2 sampai baris ke-4, yang nilai mutlaknya terbesar, lalu pertukarkan barisnya dengan baris kedua. Elemen  $x$  yang nilai mutlaknya terbesar itu sekarang menjadi *pivot* untuk operasi baris selanjutnya.

$$\left[ \begin{array}{cccc|c} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{array} \right]$$


 Cari  $|x|$  terbesar, lalu pertukarkan barisnya dengan baris ke-2

Perhatikanlah bahwa teknik *pivoting* sebagian juga sekaligus menghindari pemilihan *pivot* = 0 (sebagaimana pada *simple pivoting*) karena 0 tidak akan pernah menjadi elemen dengan nilai mutlak terbesar, kecuali jika seluruh elemen di kolom yang diacu adalah 0. Apabila setelah melakukan *pivoting* sebagian ternyata elemen *pivot* = 0, itu berarti sistem persamaan linier tidak dapat diselesaikan (*singular system*).

2. *Pivoting* lengkap (*complete pivoting*)

Jika disamping baris, kolom juga diikuti dalam pencarian elemen terbesar dan kemudian dipertukarkan, maka tatancang ini disebut *pivoting lengkap*. *Pivoting* lengkap jarang dipakai dalam program sederhana karena pertukaran kolom mengubah urutan suku  $x$  dan akibatnya menambah kerumitan program secara berarti [CHA91].



**Contoh 4.4**

Dengan menggunakan empat angka bena, selesaikan sistem persamaan berikut dengan metode eliminasi Gauss:

$$0.0003x_1 + 1.566x_2 = 1.569$$

$$0.3454x_1 - 2.436x_2 = 1.018$$

- (a) tanpa tatancang *pivoting* sebagian (Gauss naif)
- (b) dengan tatancang *pivoting* sebagian (Gauss yang dimodifikasi)

(Perhatikan, dengan 4 angka bena, solusi sejatinya adalah  $x_1 = 10.00$  dan  $x_2 = 1.00$ )

**Penyelesaian:**

- (a) tanpa tatancang *pivoting* sebagian:

$$\left[ \begin{array}{cc|c} 0.0003 & 1.566 & 1.569 \\ 0.3454 & -2.436 & 1.018 \end{array} \right]$$

Operasi baris pertama (0.0003 sebagai *pivot*):

$$R_2 \leftarrow \frac{R_2 - 0.3454R_1}{0.0003} = R_2 - 1151 R_1$$

(tanda “ $\leftarrow$ ” berarti “diisi” atau “diganti dengan”)

Jadi,

$$a_{21} \approx 0$$

$$a_{22} \approx -2.436 - (1151)(1.566) \approx -2.436 - 1802 \approx -1804$$

$$b_2 \approx 1.018 - (1151)(1.569) \approx 1.018 - 1806 \approx -1805$$

$$\left[ \begin{array}{cc|c} 0.0003 & 1.566 & 1.569 \\ 0.3454 & -2.436 & 1.018 \end{array} \right] \xrightarrow{R_2 - 1151R_1} \sim \left[ \begin{array}{cc|c} 0.0003 & 1.566 & 1.569 \\ 0 & -1804 & -1805 \end{array} \right]$$

Solusinya diperoleh dengan teknik penyulihan mundur:

$$x_2 = -1805/-1804 = 1.001$$

$$x_1 = \frac{1.569 - (1.566)(1.001)}{0.0003} = \frac{1.569 - 1.568}{0.0003} = \frac{0.001}{0.0003} = 3.333$$

(jauh dari solusi sejati)

Jadi,  $x = (3.333, 1.001)^T$ . Solusi ini sangat jauh berbeda dengan solusi sejatinya. Kegagalan ini terjadi karena  $|a_{11}|$  sangat kecil dibandingkan  $|x_{12}|$ , sehingga galat

pembulatan yang kecil pada  $x_2$  menghasilkan galat besar di  $x_1$ . Perhatikan juga bahwa  $1.569 - 1.568$  adalah pengurangan dua buah bilangan yang hampir sama, yang menimbulkan hilangnya angka bena pada hasil pengurangannya (*loss of significance*).

- (b) dengan tata-ancang *pivoting* sebagian

Baris pertama dipertukarkan dengan baris kedua sehingga 0.3454 menjadi pivot

$$\left[ \begin{array}{cc|c} 0.3454 & -2.436 & 1.018 \\ 0.0003 & 1.566 & 1.569 \end{array} \right] R_2 - \frac{0.0003}{0.3454} R_1 \sim \left[ \begin{array}{cc|c} 0.3454 & -2.436 & 1.018 \\ 0 & 1.568 & 1.568 \end{array} \right]$$

Dengan teknik penyulihan mundur diperoleh

$$x_2 = 1.568/1.568 = 1.000$$

$$x_1 = \frac{1.018 - (-2.436)(1.000)}{0.3454} = 10.02 \text{ (lebih baik daripada solusi (a))}$$

Jadi, solusinya adalah  $x = (10.02, 1.000)^T$ , yang lebih baik daripada solusi (a). Keberhasilan ini karena  $|a_{21}|$  tidak sangat kecil dibandingkan dengan  $|a_{22}|$ , sehingga galat pembulatan yang kecil pada  $x_2$  tidak akan menghasilkan galat yang besar pada  $x_1$ . ■

#### Contoh 4.5

Dengan menggunakan empat angka bena, selesaikan sistem persamaan berikut ini dengan metode eliminasi Gauss:

$$1.133x_1 + 5.281x_2 = 6.414$$

$$24.14x_1 - 1.210x_2 = 22.93$$

- (a) tanpa tatancang *pivoting* sebagian (Gauss naif)  
(b) dengan tatancang *pivoting* sebagian

(Perhatikan, dengan 4 angka bena, solusi sejatinya adalah  $x_1 = x_2 = 1.000$ )

#### Penyelesaian:

- (a) tanpa tatancang *pivoting* sebagian

$$\left[ \begin{array}{cc|c} 1.133 & 5.281 & 6.414 \\ 24.14 & -1.210 & 22.93 \end{array} \right] R_2 - \left( \frac{24.14}{1.133} \right) R_1 \sim \left[ \begin{array}{cc|c} 1.133 & 5.281 & 6.414 \\ 0 & -113.7 & -113.8 \end{array} \right]$$

Solusinya diperoleh dengan teknik penyulihan mundur:

$$x_2 = -113.8/-113.7 = 1.001$$

$$x_1 = \frac{6.414 - (5.281)(1.001)}{1.133} = 0.9956$$

Jadi,  $x = (0.9956, 1.001)^T$ . Solusi ini kurang teliti dibandingkan dengan solusi sejatinya

(b) dengan tatancang *pivoting* sebagian

Baris ke-1 dipertukarkan dengan baris ke-2, sehingga 24.14 menjadi *pivot*.

$$\left[ \begin{array}{cc|c} \mathbf{24.14} & 1.210 & 22.93 \\ 1.133 & 5.281 & 6.414 \end{array} \right] R_2 - (1.133/24.14)R_1 \left[ \begin{array}{cc|c} 24.14 & -1.210 & 22.93 \\ 0 & 5.338 & 5.338 \end{array} \right]$$

Dengan teknik penyulihan mundur, solusinya adalah

$$x_2 = 5.338/5.338 = 1.000$$

$$x_1 = \frac{22.93 + (1.210)(1.000)}{24.14} = 1.000$$

Jadi,  $x = (1.000, 1.000)^T$ . Solusi ini tepat sama dengan solusi sejatinya, jadi lebih baik daripada solusi (a) di atas. ■

Contoh 4.4 dan Contoh 4.5 di atas memperlihatkan bahwa dengan tatancang *pivoting* sebagian galat pembulatan dapat dikurangi. Contoh lainnya untuk sistem dengan tiga persamaan berikut:

$$\left[ \begin{array}{cccc|c} 0 & 2 & 0 & 1 & 0 \\ 2 & 2 & 3 & 2 & -2 \\ 4 & -3 & 0 & 1 & -7 \\ 6 & 1 & -6 & -5 & 6 \end{array} \right] R_1 \Leftrightarrow R_4 \quad (*) \quad \left[ \begin{array}{cccc|c} \mathbf{6} & 1 & -6 & -5 & 6 \\ 2 & 2 & 3 & 2 & -2 \\ 4 & -3 & 0 & 1 & -7 \\ 0 & 2 & 0 & 1 & 0 \end{array} \right] \begin{array}{l} R_2 - \frac{2}{6}R_1 \\ R_3 - \frac{4}{6}R_1 \\ \sim \end{array}$$

$$\left[ \begin{array}{cccc|c} 6 & 1 & -6 & -5 & 6 \\ 0 & 1.6667 & 5 & 3.666 & -4 \\ 0 & -3.6667 & 4 & 4.333 & -4 \\ 0 & 2 & 0 & 1 & 0 \end{array} \right] R_2 \Leftrightarrow R_3 \quad (*) \quad \left[ \begin{array}{cccc|c} 6 & 1 & -6 & -5 & 6 \\ 0 & -3.6667 & 4 & 4.3333 & 11 \\ 0 & 1.6667 & 5 & 3.6667 & -4 \\ 0 & 2 & 0 & 1 & 0 \end{array} \right] \text{dst ...}$$

Metode eliminasi Gauss yang diperbaiki (tidak naif) adalah metode eliminasi Gauss yang melibatkan operasi pertukaran baris dalam rangka memilih elemen *pivot* dengan nilai mutlak terbesar. Program 4.3 berikut berisi algoritma eliminasi Gauss yang diperbaiki.

**Program 4.3** Metode Eliminasi Gauss yang diperbaiki (dengan tatancang *pivoting*)

```

procedure Eliminasi_Gauss(A : matriks; b : vektor; n:integer;
                        var x : vektor);
{ Menghitung solusi sistem persamaan linier  $Ax = b$  dengan metode eliminasi
  Gauss yang diperbaiki.
  K.Awal : A adalah matriks yang berukuran  $n \times n$ , elemennya sudah terdefinisi
            harganya; b adalah vektor kolom yang berukuran  $n \times 1$ 
  K.Akhir: x berisi solusi sistem. Jika tidak ada solusi yang unik, vektor
            x diisi dengan nilai -9999
}

var
  i, k, j, r, s, t : integer;
  m, tampung, pivot : real;
  singular : boolean;      { true jika SPL tidak mempunyai solusi }

begin
  k:=1; singular:=false;
  while (k<=n-1) and (not singular) do
    begin
      {cari elemen pivot dengan nilai mutlak terbesar}
      pivot:=a[k,k]; r:=k; {baris pivot}
      for t:=k+1 to n do {bandingkan dengan elemen pada baris k+1 ..n}
        if ABS(a[t,k]) > ABS(pivot) then
          begin
            pivot:=a[t,k]; r:=t;
          end {if} ;
      {jika pivot=0 maka matriks A singular. Proses dihentikan}
      if pivot = 0 then { atau hampir nol, gunakan suatu epsilon }
        singular:=true
      else
        begin
          if r > k then {jika pivot tetap pada baris k, tidak ada
            pertukaran}
            begin
              {pertukarkan baris k dengan baris r di matriks A}

              for s:=1 to n do
                begin
                  tampung:=a[k,s]; a[k,s]:=a[r,s]; a[r,s]:=tampung;
                end;

              {pertukarkan juga b[k] dengan b[r]}
              tampung:=b[k]; b[k]:=b[r]; b[r]:=tampung;
            end {if} ;
          for i:=(k+1) to n do {eliminasi dari baris k+1 sampai
            baris n}
            begin
              m:=a[i,k]/a[k,k]; {hitung faktor pengali}
              for j:=k to n do {eliminasi dari kolom k sampai kolom n}
                a[i,j]:=a[i,j] - m*a[k,j];
            end
          end
        end
      end
    end
  end

```

```

        {endfor}
        b[i]:=b[i] - m*b[k];      {eliminasi vektor b pada baris i}
    end {for} ;
    end {if} ;
    k:=k+1;
end {while};
{ k = n or singular }

if not singular then
    Sulih_Mundur(A, b, n, x); {dapatkan solusinya dengan teknik penyulihan
                                mundur}
else
    { solusi tidak ada, tetapi vektor x harus tetap diisi }
    for i:=1 to n do
        x[i]:=-9999;
    {endfor}
    {endif}
end;

```

Untuk hasil terbaik, penerapan tatancang *pivoting* dan penggunaan bilangan berketelitian ganda dapat mengurangi galat pembulatan.

Pertukaran elemen baris, sebagai akibat dari pemilihan *pivot*, memakan waktu, khususnya pada SPL yang berukuran besar. Waktu pertukaran ini dapat dikurangi bila elemen-elemen baris tidak benar-benar ditukar secara aktual. Urutan baris dicatat di dalam larik `BAR[1..n]`. Pertukaran yang dikerjakan hanyalah pertukaran elemen larik `BAR`. Pada mulanya larik `BAR` berisi indeks baris matriks:

```
for i:=1 to n do BAR[i]:=i;
```

Elemen matriks diacu sebagai

```
A[BAR[i], k]
```

Maka, pertukaran baris  $k$  dan baris  $r$  dikerjakan sebagai

```
tampung:=BAR[r];
BAR[r]:=BAR[k];
BAR[k]:=tampung;
```

## 4.2.2 Penskalaan

Selain dengan *pivoting* sebagian, penskalaan (*scaling*) juga dapat digunakan untuk mengurangi galat pembulatan pada SPL yang mempunyai perbedaan koefisien yang mencolok. Situasi demikian sering ditemui dalam praktek rekayasa yang menggunakan ukuran satuan yang berbeda-beda dalam menentukan persamaan

simultan. Misalnya pada persoalan rangkaian listrik, tegangan listrik dapat dinyatakan dalam satuan yang berkisar dari mikrovolt sampai kilovolt. Pemakaian satuan yang berbeda-beda dapat menuju ke koefisien yang besarnya sangat berlainan. Ini berdampak pada galat pembulatan, dan karena itu mempengaruhi *pivoting* [CHA91]. Dengan penskalaan berarti kita menormalkan persamaan. Cara menskala adalah membagi tiap baris persamaan dengan nilai mutlak koefisien terbesar di ruas kirinya. Akibat penskalaan, koefisien maksimum dalam tiap baris adalah 1. Cara menskala seperti ini dinamakan dengan **menormalkan SPL**.

#### Contoh 4.6

Selesaikan sistem persamaan linier berikut sampai 3 angka bena dengan menggunakan metode eliminasi Gauss yang menerapkan penskalaan dan tanpa penskalaan:

$$\begin{aligned} 2x_1 + 100000x_2 &= 100000 \\ x_1 + x_2 &= 2 \end{aligned}$$

(Solusi sejatinya dalam 3 angka bena adalah  $x_1 = x_2 = 1.00$ )

#### Penyelesaian:

(i) Tanpa penskalaan :

$$\left[ \begin{array}{cc|c} 2 & 100000 & 100000 \\ 1 & 1 & 2 \end{array} \right] R_2 - 1/2 R_1 \rightarrow \left[ \begin{array}{cc|c} 2 & 100000 & 100000 \\ 0 & -50000 & -50000 \end{array} \right]$$

Solusinya adalah

$$\begin{aligned} x_2 &= 1.00 \\ x_1 &= 0.00 \quad (\text{salah}) \end{aligned}$$

(ii) Dengan penskalaan :

$$\begin{array}{lcl} 2x_1 + 100000x_2 = 100000 & : 100000 & 0.00002 \quad x_1 + x_2 = 1 \\ x_1 + x_2 = 2 & : 1 & x_1 + x_2 = 2 \end{array}$$

$$\left[ \begin{array}{cc|c} 0.00002 & 1 & 1 \\ 1 & 1 & 2 \end{array} \right] R_1 \leftrightarrow R_2 \begin{array}{c} (*) \end{array} \left[ \begin{array}{cc|c} 1 & 1 & 2 \\ 0.00002 & 1 & 1 \end{array} \right] \sim \left[ \begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1 & 1.00 \end{array} \right]$$

Solusinya,

$$\begin{aligned} x_2 &= 1.00 \\ x_1 &= 1.00 \quad (\text{benar}) \end{aligned}$$

yang sesuai dengan solusi sejati. Contoh di atas juga memperlihatkan bahwa penskalaan dapat mengubah pemilihan *pivot*. ■

## 4.2.2 Kemungkinan Solusi SPL

Tidak semua SPL mempunyai solusi. Ada tiga kemungkinan solusi yang dapat terjadi pada SPL:

- (a) mempunyai solusi yang unik,
- (b) mempunyai banyak solusi, atau
- (c) tidak ada solusi sama sekali.

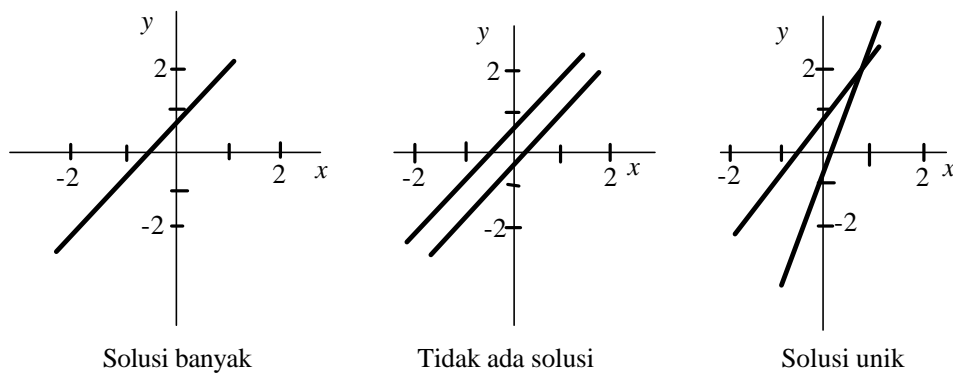
Dengan grafik, ketiga kemungkinan solusi ini diperlihatkan oleh tiga SPL dengan dua persamaan berikut [NAK92]:

(i)  $-x + y = 1$   
 $-2x + 2y = 2$

(ii)  $-x + y = 1$   
 $-x + y = 0$

(iii)  $-x + y = 1$   
 $2x - y = 0$

Grafik ketiga SPL diperlihatkan pada Gambar 4.3. Grafik pertama memperlihatkan bahwa kedua persamaan berimpit pada satu garis lurus. Solusinya terdapat di sepanjang garis tersebut (banyak solusi). Grafik kedua memperlihatkan kedua persamaan menyatakan dua garis yang sejajar. Tidak ada perpotongan kedua garis tersebut (tidak ada solusi). Sedangkan pada grafik ketiga, kedua persamaan berpotongan pada sebuah titik (solusinya tunggal atau unik).



**Gambar 4.3** Kemungkinan solusi sistem persamaan linjar

Untuk SPL dengan tiga buah persamaan atau lebih (dengan tiga peubah atau lebih), tidak terdapat tafsiran geometrinya (tidak mungkin dibuat ilustrasi grafisnya) seperti pada SPL dengan dua buah persamaan. Namun, kita masih dapat memeriksa masing-masing kemungkinan solusi itu berdasarkan pada bentuk matriks akhirnya. Agar lebih jelas, tinjau contoh pada SPL yang disusun oleh tiga persamaan.

### 1. Solusi unik/tunggal

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 2 & 3 & 1 & 1 \\ 3 & 1 & 2 & 1 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[ \begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & -3 & 3 \end{array} \right]$$

Solusi:  $x_1 = 1, x_2 = 0, x_3 = -1$

### 2. Solusi banyak/tidak terhingga

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 2 & -1 & 1 & 2 \\ 1 & 2 & 3 & 6 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[ \begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 0 & -3 & -3 & -6 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

Perhatikan hasil eliminasi Gauss pada baris terakhir. Persamaan yang bersesuaian dengan baris terakhir tersebut adalah

$$0x_1 + 0x_2 + 0x_3 = 0$$

yang dipenuhi oleh banyak nilai  $x$ . Solusinya diberikan dalam bentuk parameter:

Misalkan  $x_3 = k$ ,

maka  $x_2 = -6 + 3k$  dan  $x_1 = 10 - 5k$ , dengan  $k \in R$ .

Terdapat tidak berhingga nilai  $k$ , berarti solusi SPL banyak sekali.

### 3. Tidak ada solusi

$$\left[ \begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 2 & -1 & 1 & 2 \\ 1 & 2 & 3 & 7 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[ \begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 0 & -3 & -3 & -6 \\ 0 & 0 & 0 & 1 \end{array} \right]$$

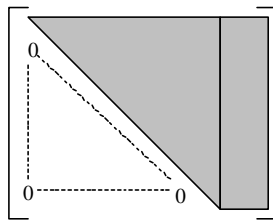


Perhatikan hasil eliminasi Gauss pada baris terakhir. Persamaan yang bersesuaian dengan baris terakhir tersebut adalah

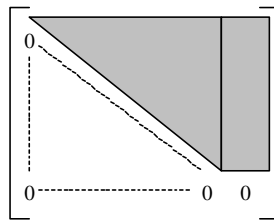
$$0x_1 + 0x_2 + 0x_3 = 1$$

yang dalam hal ini, tidak nilai  $x_i$  yang memenuhi,  $i = 1, 2, 3$

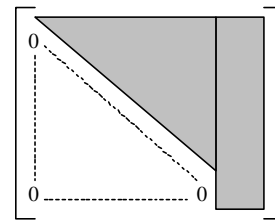
Bentuk akhir matriks setelah eliminasi Gauss untuk ketiga kemungkinan solusi di atas dapat digambarkan sebagai berikut:



Solusi unik



Solusi banyak



Tidak ada solusi

Kita rangkum “pertanda” kemungkinan solusi SPL di bawah ini:

1. Jika pada hasil eliminasi Gauss tidak terdapat baris yang semuanya bernilai 0 (termasuk elemen pada baris yang bersesuaian pada vektor kolom  $b$ ), maka solusi SPL dipastikan unik.
2. Jika pada hasil eliminasi Gauss terdapat paling sedikit satu baris yang semuanya bernilai 0 (termasuk elemen pada baris yang bersesuaian pada vektor kolom  $b$ ), maka SPL mempunyai banyak solusi.
3. Jika pada hasil eliminasi Gauss terdapat baris yang semuanya bernilai 0 tetapi elemen pada baris yang bersesuaian pada vektor kolom  $b$  tidak 0, maka SPL tidak mempunyai solusi.

Program eliminasi Gauss harus dapat menangani ketiga kemungkinan solusi tersebut.

## 4.3 Metoda Eliminasi Gauss-Jordan

Metode eliminasi Gauss-Jordan merupakan variasi dari metode eliminasi Gauss. Dalam hal ini, matriks  $A$  dieliminasi menjadi matriks identitas  $I$ . Di sini tidak diperlukan lagi teknik penyulihan mundur untuk memperoleh solusi SPL. Solusinya langsung diperoleh dari vektor kolom  $b$  hasil proses eliminasi.

$$Ax = b \rightarrow Ix = b'$$

Dalam bentuk matriks, eliminasi Gauss-Jordan ditulis sebagai

$$\left[ \begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & b_3 \\ \vdots & & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & b_n \end{array} \right] \longrightarrow \left[ \begin{array}{cccc|c} 1 & 0 & 0 & \dots & 0 & b_1' \\ 0 & 1 & 0 & \dots & 0 & b_2' \\ 0 & 0 & 1 & \dots & 0 & b_3' \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & b_n' \end{array} \right]$$

$$\begin{array}{rcl} \text{Solusinya:} & x_1 & = b_1' \\ & x_2 & = b_2' \\ & \dots & \dots \\ & x_n & = b_n' \end{array}$$

Seperti pada metode eliminasi Gauss naif, metode eliminasi Gauss-Jordan naif tidak menerapkan tata-ancang pivoting dalam proses eliminasinya.

**Program 4.4** Metode Eliminasi Gauss-Jordan Naif

```

procedure Eliminasi_Gauss_Jordan_Naif(A : matriks; b: vektor; n:integer;
var x : vektor);
{ Menghitung solusi sistem persamaan linjar Ax = b dengan metode eliminasi
  Gauss-Jordan.
  K.Awal : A adalah matriks yang berukuran n ´ n, elemennya sudah terdefinisi
            harganya; b adalah vektor kolom yang berukuran n ´ 1
  K.Akhir: x berisi solusi sistem
}
var
  i; k, j : integer;
  m, tampung: real;
begin
  for k:=1 to n do
    begin
      tampung:=a[k,k];
      for j:=1 to n do {bagi elemen baris k dengan a[k,k]}
        a[k,j]:=a[k,j]/tampung;
      {endfor}
      b[k]:=b[k]/tampung; {jangan lupa b[k] juga dibagi dengan a[k,k]}
      for i:=1 to n do {eliminasi elemen baris i s/d baris n, i¹k}

```

```

begin
  if i<>k then
    begin
      m:=a[i,k];
      for j:=1 to n do {eliminasi elemen dari kolom 1 s/d kolom n}
        a[i,j]:=a[i,j] - m*a[k,j];
      {endfor}
      b[i]:=b[i] - m*b[k]; {eliminasi elemen vektor b pada baris i}
    end;
  end;
end;
{Solusi langsung didapat dari vektor kolom b}
for i:=1 to n do x[i]:=b[i];
end;

```

Seperti halnya metode eliminasi Gauss, tatancang *pivoting* dan penskalaan juga dapat diterapkan pada metoda ini untuk memperkecil galat pembulatan.

#### Contoh 4.7

[CHA91] Selesaikan sistem persamaan linjar di bawah ini dengan metode eliminasi Gauss- Jordan.

$$\begin{aligned}
 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\
 0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\
 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4
 \end{aligned}$$

**Penyelesaian:**

$$\left[ \begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{array} \right] \xrightarrow{R_1/3} \left[ \begin{array}{ccc|c} 1 & -0.0333333 & -0.0666667 & 2.61667 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{array} \right]$$

$$\begin{array}{l} R_2 - 0.1 R_1 \\ R_3 - 0.3 R_1 \\ \sim \end{array} \left[ \begin{array}{ccc|c} 1 & -0.0333333 & -0.0666667 & 2.61667 \\ 0 & 7.00333 & -0.293333 & -19.5617 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{array} \right]$$

$$R_2 / 7.00333 \left[ \begin{array}{ccc|c} 1 & -0.0333333 & -0.0666667 & 2.61667 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{array} \right]$$

$$\begin{array}{l} R_1 - (-0.003333)R_2 \\ R_3 - (-0.190000)R_2 \end{array} \left[ \begin{array}{ccc|c} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 10.01200 & 70.0843 \end{array} \right]$$

$$R_3 / 10.0200 \quad \left[ \begin{array}{ccc|c} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 1 & 7.00003 \end{array} \right]$$

$$\begin{array}{l} R_1 - (-0.0680629) R_3 \\ R_2 - (-0.0418848) R_3 \end{array} \quad \left[ \begin{array}{ccc|c} 1 & 0 & 0 & 3.00000 \\ 0 & 1 & 0 & -2.50001 \\ 0 & 0 & 1 & 7.00003 \end{array} \right]$$

$$\begin{array}{l} \text{Solusi: } x_1 = 3.00000 \\ x_2 = -2.50001 \\ x_3 = 7.00003 \end{array} \quad \blacksquare$$

Penyelesaian SPL dengan metode eliminasi Gauss-Jordan membutuhkan jumlah komputasi yang lebih banyak daripada metode eliminasi Gauss. Karena alasan itu, metode eliminasi Gauss sudah cukup memuaskan untuk digunakan dalam penyelesaian SPL. Namun metode eliminasi Gauss-Jordan merupakan dasar pembentukan matriks balikan yang akan dibahas di bawah ini.

### Matriks Balikan (*inverse matrices*)

Matriks balikan,  $A^{-1}$ , banyak dipakai dalam pengolahan matriks. Misalnya dalam pengukuran statistik, pencocokan fungsi pada data hasil pengamatan menggunakan metode kuadrat terkecil (*least square*). Di sini, nilai  $A^{-1}$  memberikan informasi tentang galat mutlak yang dikandung data. Selain itu, matriks balikan juga dapat dipakai untuk menghitung solusi sistem persamaan linier (akan dibahas pada metode matriks balikan). Akan ditunjukkan juga bahwa matriks balikan dapat diperoleh dengan metode eliminasi Gauss-Jordan. Tetapi sebelum membahasnya, ingatlah kembali cara menghitung matriks balikan untuk matriks  $2 \times 2$ .

Untuk matriks  $2 \times 2$ ,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

matriks balikannya adalah

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}, \quad a_{11}a_{22} - a_{12}a_{21} \neq 0.$$

Nilai  $a_{11}a_{22} - a_{21}a_{12}$  ini disebut **determinan**. Determinan dilambangkan dengan dua buah garis tegak ( $|$ ). Lebih jauh tentang determinan ini akan dijelaskan pada bagian lain bab ini. Bila determinan  $A = 0$ , matriks  $A$  tidak mempunyai balikan, sehingga dinamakan *matriks singular*. Sistem persamaan linier yang mempunyai matriks  $A$  singular (sistem singular) tidak mempunyai solusi yang unik, yaitu solusinya banyak atau solusinya tidak ada.

Untuk matriks  $n \times n$ , matriks balikkannya dapat diperoleh dengan metode eliminasi Gauss-Jordan, yaitu:

$$[A \mid I] \xrightarrow{\text{eliminasi G-J}} [I \mid A^{-1}]$$

$$\left[ \begin{array}{cccc|cccc} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & \dots & 1 \end{array} \right] \longrightarrow \left[ \begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \dots & p_{1n} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & & \vdots & \vdots & & & \vdots \\ 0 & 0 & \dots & 1 & p_{n1} & p_{n2} & \dots & p_{nn} \end{array} \right]$$

$A$                        $I$                                        $I$                        $A^{-1}$

#### Contoh 4.8

Tentukan matriks balikan dari matriks  $A$  berikut

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 0 & 1 \\ 1 & 0 & 2 \end{bmatrix}$$

**Penyelesaian:**

$$\left[ \begin{array}{ccc|ccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 \end{array} \right] \xrightarrow{\substack{R_2 - 3R_1 \\ R_3 - R_1}} \left[ \begin{array}{ccc|ccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 0 & 3 & -5 & 3 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{array} \right]$$

$$\sim \dots \sim \left[ \begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0.4 & -0.2 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -0.5 & 0.6 \end{array} \right]$$

Jadi, matriks balikan dari  $A$  adalah

$$A^{-1} = \begin{bmatrix} 0 & 0.4 & -0.2 \\ -1 & 0 & 1 \\ 0 & -0.5 & 0.6 \end{bmatrix}$$

■

Penerapan tata-ancang *pivoting* dan penggunaan bilangan berketelitian ganda dapat memperbaiki hasil matriks balikan.

## 4.4 Metode Matriks Balikan

Misalkan  $A^{-1}$  adalah matriks balikan dari  $A$ . Hasil kali  $A$  dengan  $A^{-1}$  menghasilkan matriks identitas  $I$ ,

$$AA^{-1} = A^{-1}A = I \quad (\text{P.4.6})$$

Bila matriks  $A$  dikalikan dengan  $I$  akan menghasilkan matriks  $A$  sendiri,

$$AI = IA = A \quad (\text{P.4.7})$$

Berdasarkan dua kesamaan di atas, sistem persamaan linier  $Ax = b$  dapat diselesaikan sebagai berikut:

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b && \{\text{kalikan kedua ruas dengan } A^{-1}\} \\ Ix &= A^{-1}b \\ x &= A^{-1}b \end{aligned} \quad (\text{P.4.8})$$

Jadi, penyelesaian sistem persamaan linier  $Ax = b$  adalah  $x = A^{-1}b$  dengan syarat  $A^{-1}$  ada. Cara penyelesaian dengan mengalikan matriks  $A^{-1}$  dengan  $b$  itu dinamakan **metode matriks balikan**. Tetapi, penyelesaian dengan SPL metode matriks balikan tidak lebih mangkus daripada metode eliminasi Gauss, sebab lebih banyak proses komputasi yang dibutuhkan. Metode matriks balikan baru mangkus bila digunakan untuk penyelesaian sejumlah SPL dengan matriks  $A$  yang sama tetapi dengan vektor kolom  $b$  yang berbeda-beda:

$$\begin{aligned} Ax &= b_I \\ Ax &= b_{II} \\ Ax &= b_{III} \\ &\dots \text{ dst} \end{aligned}$$

Sekali  $A^{-1}$  telah diperoleh, maka ia dapat dipakai untuk menyelesaikan sejumlah SPL tersebut.

#### Contoh 4.9

Selesaikan sistem persamaan linjar

$$\begin{aligned}x_1 - x_2 + 2x_3 &= 5 \\3x_1 + x_3 &= 10 \\x_1 + 2x_3 &= 5\end{aligned}$$

dengan metode matriks balikan.

**Penyelesaian:**

$$\begin{aligned}&\left[ \begin{array}{ccc|ccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 \end{array} \right] \xrightarrow[R_3 - R_1]{R_2 - 3R_1} \left[ \begin{array}{ccc|ccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 0 & 3 & -5 & -3 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \end{array} \right] \\&\sim \dots \sim \left[ \begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0.4 & -0.2 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -0.2 & 0.6 \end{array} \right] \\&\quad \quad \quad \underbrace{\hspace{1.5cm}}_{A^{-1}}\end{aligned}$$

Solusinya adalah  $x = A^{-1} b$ .

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 0.4 & -0.2 \\ -1 & 0 & 1 \\ 0 & -0.2 & 0.6 \end{bmatrix} \begin{bmatrix} 5 \\ 10 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 & + & 4 & - & 1 \\ -5 & + & 0 & + & 5 \\ 0 & - & 2 & + & 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} \quad \blacksquare$$

Perlu diperhatikan, apabila selama pembentukan matriks balikan terdapat proses *pivoting* (pertukaran baris), baris-baris pada  $b$  juga harus dipertukarkan.

## 4.5 Metode Dekomposisi LU

Jika matriks  $A$  *non-singular* maka ia dapat difaktorkan (diuraikan atau di-dekomposisi) menjadi matriks segitiga bawah  $L$  (*lower*) dan matriks segitiga atas  $U$  (*upper*):

$$A = LU \quad (\text{P.4.9})$$

Dalam bentuk matriks, pemfaktoran ini ditulis sebagai

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Pada matriks segitiga bawah  $L$ , semua elemen diagonal adalah 1, sedangkan pada matriks  $U$  tidak ada aturan khusus pada elemen diagonalnya<sup>1</sup>.

Sebagai contoh, matriks  $3 \times 3$  di bawah ini difaktorkan menjadi :

$$\begin{bmatrix} 2 & -1 & -1 \\ 0 & -4 & 2 \\ 6 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & -1 \\ 0 & -4 & 2 \\ 0 & 0 & 4 \end{bmatrix}$$

Metode pemfaktoran  $A$  menjadi  $L$  dan  $U$  akan dijelaskan kemudian. Sekali  $A$  difaktorkan menjadi  $L$  dan  $U$ , kedua matriks tersebut dapat digunakan untuk menyelesaikan  $Ax = b$ . Metode penyelesaian SPL dengan cara ini dikenal dengan nama **metode dekomposisi LU**. Metode ini dinamakan juga **metode pemfaktoran segitiga** (*triangular factorization*). Nanti akan ditunjukkan bahwa metode eliminasi Gauss merupakan suatu dekomposisi  $LU$  dari matriks  $A$ .

Penyelesaian  $Ax = b$  dengan metode dekomposisi  $LU$  adalah sebagai berikut. Tinjau sistem persamaan linier

$$Ax = b$$

Faktorkan  $A$  menjadi  $L$  dan  $U$  sedemikian sehingga

$$A = LU$$

Jadi,

$$\begin{aligned} Ax &= b \\ LUx &= b \end{aligned} \tag{P.4.10}$$

<sup>1</sup> Pada beberapa buku, yang tertera adalah kebalikannya: semua elemen diagonal dari matriks  $U$  adalah 1, sedangkan elemen diagonal matriks  $L$  bebas. Hal ini tidak masalah sebab jika  $L$  dan  $U$  dikalikan, hasilnya tetap sama dengan matriks  $A$ .



Misalkan

$$Ux = y \quad (\text{P.4.11})$$

maka

$$Ly = b \quad (\text{P.4.12})$$

Untuk memperoleh  $y_1, y_2, \dots, y_n$ , kita menggunakan teknik penyulihan maju (*forward substitution*):

$$Ly = b \rightarrow \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \rightarrow \begin{array}{l} \text{diperoleh } y_1, y_2, \dots, \\ y_n \text{ dengan teknik} \\ \text{penyulihan maju} \end{array}$$

Dan untuk memperoleh solusi SPL,  $x_1, x_2, \dots, x_n$ , kita menggunakan teknik penyulihan mundur (*backward substitution*):

$$Ux = y \rightarrow \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \begin{array}{l} \text{diperoleh} \\ x_1, x_2, \dots, x_n \\ \text{dengan teknik} \\ \text{penyulihan} \\ \text{mundur} \end{array}$$

Jadi, langkah-langkah menghitung solusi SPL dengan metode dekomposisi  $LU$  dapat diringkas sebagai berikut:

1. Bentuklah matriks  $L$  dan  $U$  dari  $A$
2. Pecahkan  $Ly = b$ , lalu hitung  $y$  dengan teknik penyulihan maju
3. Pecahkan  $Ux = y$ , lalu hitung  $x$  dengan teknik penyulihan mundur

Sama halnya dengan metode matriks balikan, metode dekomposisi  $LU$  akan mangkus bila digunakan untuk menyelesaikan sejumlah SPL dengan matriks  $A$  yang sama tetapi dengan  $b$  berbeda-beda. Sekali  $A$  difaktorkan menjadi  $L$  dan  $U$ , keduanya dapat digunakan untuk menghitung solusi sejumlah SPL tersebut. Metode dekomposisi  $LU$  merupakan metode yang paling populer untuk memecahkan sistem persamaan linier.

Terdapat dua metode untuk memfaktorkan  $A$  atas  $L$  dan  $U$ :

1. Metode  $LU$  Gauss.
2. Metode reduksi Crout.

Masing-masing metode pemfaktoran kita bahas di bawah ini.

### 4.5.1 Pemfaktoran dengan Metode LU Gauss

Walaupun tidak ada hubungannya dengan dekomposisi  $LU$ , metode eliminasi Gauss dapat digunakan untuk memfaktorkan  $A$  menjadi  $L$  dan  $U$  (karena itulah metode pemfaktoran ini kita namakan metode LU Gauss). Di dalam upabab ini juga akan ditunjukkan bahwa sebenarnya metode eliminasi Gauss dapat dinyatakan sebagai dekomposisi  $LU$ .

Misalkan matriks  $A$  berukuran  $4 \times 4$  difaktorkan atas  $L$  dan  $U$ ,

$$A = LU$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

Di sini kita menggunakan simbol  $m_{ij}$  ketimbang  $l_{ij}$ , karena nilai  $l_{ij}$  berasal dari faktor pengali ( $m_{ij}$ ) pada proses eliminasi Gauss. Langkah-langkah pembentukan  $L$  dan  $U$  dari matriks  $A$  adalah sebagai berikut:

1. Nyatakan  $A$  sebagai  $A = IA$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

2. Eliminasi matriks  $A$  di ruas kanan menjadi matriks segitiga atas  $U$ . Tempatkan faktor pengali  $m_{ij}$  pada posisi  $l_{ij}$  di matriks  $I$ .

3. Setelah seluruh proses eliminasi Gauss selesai, matriks  $I$  menjadi matriks  $L$ , dan matriks  $A$  di ruas kanan menjadi matriks  $U$ .

Di bawah ini diberikan dua contoh pemfaktoran  $A$  dengan metode ini, masing-masing untuk kasus tanpa *pivoting* dan dengan *pivoting*.

**Contoh 4.10** (*LU* Gauss naif)

Faktorkan matriks  $A$  berikut dengan metode *LU* Gauss:

$$A = \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix}$$

**Penyelesaian:**

$$A = \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix}$$

Eliminasikan matriks  $A$  di ruas kanan menjadi matriks segitiga atas  $U$ , dan tempatkan faktor pengali  $m_{ij}$  pada posisi  $l_{ij}$  di matriks  $L$ .

$$\begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} \begin{matrix} R_2 - (-2/4)R_1 \\ \sim \\ R_3 - (1/4)R_1 \end{matrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 1.25 & 6.25 \end{bmatrix}$$

Tempatkan  $m_{21} = -2/4 = 0.5$  dan  $m_{31} = 1/4 = 0.25$  ke dalam matriks  $L$ :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & m_{32} & 1 \end{bmatrix}$$

Teruskan proses eliminasi Gauss pada matriks  $A$ ,

$$\begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 1.25 & 6.25 \end{bmatrix} \begin{matrix} R_3 - (1.25/-2.5)R_2 \\ \sim \end{matrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 0 & 8.5 \end{bmatrix} = U$$

Tempatkan  $m_{32} = 1.25/-2.5 = -0.5$  ke dalam matriks  $L$ :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{bmatrix}$$

Jadi,

$$A = \begin{bmatrix} 4 & 3 & -1 \\ -2 & -4 & 5 \\ 1 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & -1 \\ 0 & -2.5 & 4.5 \\ 0 & 0 & 8.5 \end{bmatrix} \quad \blacksquare$$

**Contoh 4.11** (*LU Gauss dengan tata-ancang pivoting*)

Faktorkan matriks  $A$  berikut

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 5 \\ 1 \end{bmatrix}$$

lalu pecahkan sistem  $Ax = b$ .

**Penyelesaian:**

Eliminasikan matriks  $A$  di ruas kanan menjadi matriks segitiga atas  $U$ , dan tempatkan faktor pengali  $m_{ij}$  pada posisi  $l_{ij}$  di matriks  $L$ .

$$\begin{bmatrix} 1 & 1 & -1 \\ 2 & 2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \quad \begin{matrix} R_2 - (2)R_1 \\ \sim \\ R_3 - (-1/1)R_1 \end{matrix} \quad \begin{bmatrix} 1 & 1 & -1 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \end{bmatrix}$$

Tempatkan  $m_{21} = 2$  dan  $m_{31} = 1/1 = 1$  ke dalam matriks  $L$ :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & m_{32} & 1 \end{bmatrix}$$

Teruskan proses eliminasi Gauss pada matriks  $A$ . Dalam hal ini ada *pivoting* karena calon *pivot* bernilai 0, sehingga baris kedua dipertukarkan dengan baris ketiga:

$$\begin{bmatrix} 1 & 1 & -1 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \end{bmatrix} \quad R_2 \Leftrightarrow R_3 \quad \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Jangan lupa mempertukarkan juga  $R_2 \Leftrightarrow R_3$  pada matriks  $L$ , kecuali elemen diagonalnya

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & m_{32} & 1 \end{bmatrix} \quad R_2 \Leftrightarrow R_3 \quad \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & m_{32} & 1 \end{bmatrix}$$

Jangan lupa mempertukarkan juga  $R_2 \Leftrightarrow R_3$  pada vektor  $b$ ,

$$b = \begin{bmatrix} 1 \\ 5 \\ 1 \end{bmatrix} \quad R_2 \Leftrightarrow R_3 \quad \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}$$

Teruskan proses eliminasi Gauss pada matriks  $A$ :

$$R_3 - (0/2)R_2 \quad \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} = U$$

Tempatkan  $m_{32} = 0/2 = 0$  ke dalam matriks  $L$ :

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

Jadi,

$$A = \begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Berturut-turut dihitung  $y$  dan  $x$  sebagai berikut:

$$Ly = b \quad \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}$$

$y_1, y_2$ , dan  $y_3$  dihitung dengan teknik penyulihan maju:

$$\begin{aligned} y_1 &= 1 \\ -y_1 + y_2 &= 1 \rightarrow y_2 = 1 + y_1 = 1 + 1 = 2 \\ 2y_1 + 0y_2 + y_3 &= 5 \rightarrow y_3 = 5 - 2y_1 = 3 \end{aligned}$$

$$Ux = y \longrightarrow \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$x_1, x_2$ , dan  $x_3$  dihitung dengan teknik penyulihan mundur:

$$\begin{aligned} 3x_3 &= 3 \rightarrow x_3 = 1 \\ 2x_2 + 0x_3 &= 2 \rightarrow x_2 = 1 \\ x_1 + x_2 - x_3 &= 1 \rightarrow x_1 = 1 \end{aligned}$$

Jadi, solusi sistem persamaan linier di atas adalah  $x = (1, 1, 1)^T$ . ■

Pertukaran baris untuk matriks yang berukuran besar diperlihatkan oleh matriks di bawah ini:

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ 0 & b_2 & b_3 & b_4 & b_5 & b_6 \\ 0 & 0 & c_3 & c_4 & c_5 & c_6 \\ 0 & 0 & 0 & 0 & d_5 & d_6 \\ 0 & 0 & 0 & e_4 & e_5 & e_6 \\ 0 & 0 & 0 & f_4 & f_5 & f_6 \end{bmatrix} \xrightarrow[\quad (*) \quad]{R_5 \Leftrightarrow R_4} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ 0 & b_2 & b_3 & b_4 & b_5 & b_6 \\ 0 & 0 & c_3 & c_4 & c_5 & c_6 \\ 0 & 0 & 0 & e_4 & e_5 & e_6 \\ 0 & 0 & 0 & 0 & d_5 & d_6 \\ 0 & 0 & 0 & f_4 & f_5 & f_6 \end{bmatrix}$$

Maka, baris ke-5 dan baris ke-4 pada matriks  $L$  juga harus dipertukarkan:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 & 0 & 0 \\ \boxed{m_{41} \quad m_{42} \quad m_{43}} & 1 & 0 & 0 & & \\ \boxed{m_{51} \quad m_{52} \quad m_{53}} & x & 1 & 0 & & \\ m_{61} & m_{62} & m_{63} & x & x & 1 \end{bmatrix} \xrightarrow[\quad (*) \quad]{R_5 \Leftrightarrow R_4} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ m_{21} & 0 & 0 & 0 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 & 0 & 0 \\ \boxed{m_{51} \quad m_{52} \quad m_{53}} & 1 & 0 & 0 & & \\ \boxed{m_{41} \quad m_{42} \quad m_{43}} & x & 1 & 0 & & \\ m_{61} & m_{62} & m_{63} & x & x & 1 \end{bmatrix}$$

## 4.5.2 Metode Reduksi Crout

Meskipun metode  $LU$  Gauss dikenal paling baik untuk melakukan dekomposisi  $LU$ , terdapat metode lain yang digunakan secara luas, yaitu metode reduksi (dekomposisi) Crout (atau **metode reduksi Cholesky** atau **metode Dolittle**).

Dalam membahas metode reduksi Crout, tinjau matriks  $3 \times 3$  berikut:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{3,2} & 1 \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{2,2} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Karena  $LU = A$ , maka hasil perkalian  $L$  dan  $U$  itu dapat ditulis sebagai

$$LU = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} \end{bmatrix} = A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Dari kesamaan dua buah matriks  $LU = A$ , diperoleh

$$u_{11} = a_{11}, \quad u_{12} = a_{12}, \quad u_{13} = a_{13} \quad \} \quad \text{Baris pertama } U$$

$$\left. \begin{aligned} l_{21}u_{11} &= a_{21} \rightarrow l_{21} = \frac{a_{21}}{u_{11}} \\ l_{31}u_{11} &= a_{31} \rightarrow l_{31} = \frac{a_{31}}{u_{11}} \end{aligned} \right\} \quad \text{Kolom pertama } L$$

$$\left. \begin{aligned} l_{21}u_{12} + u_{22} &= a_{22} \rightarrow u_{22} = a_{22} - l_{21}u_{12} \\ l_{21}u_{13} + u_{23} &= a_{23} \rightarrow u_{23} = a_{23} - l_{21}u_{13} \end{aligned} \right\} \quad \text{Baris kedua } U$$

$$l_{31}u_{12} + l_{32}u_{22} = a_{32} \rightarrow l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} \quad \} \quad \text{Kolom kedua } L$$

$$l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33} \rightarrow u_{33} = a_{33} - (l_{31}u_{13} + l_{32}u_{23}) \quad \} \quad \text{Baris ketiga } U$$

Kita perhatikan ada urutan pola teratur dalam menemukan elemen-elemen  $L$  dan  $U$ , yaitu:

- elemen-elemen baris pertama dari  $U$
- elemen-elemen baris pertama dari  $L$
- elemen-elemen baris kedua dari  $U$
- elemen-elemen baris kedua  $L$
- ...
- elemen-elemen baris ke- $k$  dari  $U$
- elemen-elemen baris ke- $k$  dari  $L$

Rumus umum menghitung  $u$  dan  $l$  untuk sistem dengan matriks  $A$  yang berukuran  $3 \times 3$  dapat ditulis sebagai berikut:

$$u_{pj} = a_{pj} - \sum_{k=1}^{p-1} l_{pk} u_{kj}, \quad \begin{matrix} p = 1, 2, 3, \dots, n \\ j = p, p+1, \dots, n \end{matrix} \quad (\text{P.4.13})$$

dan

$$l_{iq} = \frac{a_{iq} - \sum_{k=1}^{q-1} l_{ik} u_{kq}}{u_{qq}}, \quad \begin{matrix} q = 1, 2, 3, \dots, n-1 \\ i = q+1, q+2, \dots, n \end{matrix} \quad (\text{P.4.14})$$

dengan syarat  $u_{qq} \neq 0$

#### Contoh 4.12

Selesaikan

$$\begin{aligned} x_1 + x_2 - x_3 &= 1 \\ 2x_1 + 2x_2 + x_3 &= 5 \\ -x_1 + x_2 + 2x_3 &= 5 \end{aligned}$$

dengan metode dekomposisi  $LU$ , yang dalam hal ini  $L$  dan  $U$  dihitung dengan metode reduksi Crout.

**Penyelesaian:**

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 5 \\ 1 \end{bmatrix}$$



Diperoleh:

$$\begin{aligned}
 u_{11} &= a_{11} = 1 \\
 u_{12} &= a_{12} = 1 \\
 u_{13} &= a_{13} = -1 \\
 l_{21} &= a_{21}/u_{11} = 2/1 = 2 \\
 l_{31} &= a_{31}/u_{11} = -1/1 = -1 \\
 u_{22} &= a_{22} - l_{21}u_{12} = 2 - 2 \cdot 1 = 0
 \end{aligned}$$

Karena  $u_{qq}$  tidak boleh nol, lakukan pertukaran baris, baik untuk matriks  $A$  maupun untuk vektor  $b$ :

<u>Matriks <math>A</math></u>	<u>Vektor <math>b</math></u>
$R_2 \Leftrightarrow R_3 \quad \begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}$	$R_2 \Leftrightarrow R_3 \quad \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}$

Hitung kembali nilai  $l_{21}$ ,  $l_{31}$ , dan  $u_{22}$  (Perhatikan bahwa nilai  $u_{11}$ ,  $u_{12}$ ,  $u_{13}$  tidak berubah)

$$\begin{aligned}
 l_{21} &= a_{21}/u_{11} = -1/1 = -1 \\
 l_{31} &= a_{31}/u_{11} = 2/1 = 2 \\
 u_{22} &= a_{22} - l_{21}u_{12} = 1 - (-1)(1) = 1 + 1 = 2 \\
 u_{23} &= a_{23} - l_{21}u_{13} = 1 - (-1)(-1) = 1 - 1 = 0 \\
 l_{32} &= \frac{a_{32} - l_{31}u_{12}}{u_{22}} = \frac{2 - 2(1)}{2} = 0
 \end{aligned}$$

Diperoleh  $L$  dan  $U$  sebagai berikut,

$$U = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \text{dan } b = \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}$$

Berturut-turut dihitung  $y$  dan  $x$  sebagai berikut:

$$Ly = b \quad \text{---} \quad \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 5 \end{bmatrix}$$

$y_1$ ,  $y_2$ , dan  $y_3$  dihitung dengan teknik penyulihan maju:

$$\begin{aligned}
 y_1 &= 1 \\
 -y_1 + y_2 &= 1 \quad \rightarrow \quad y_2 = 1 + y_1 = 1 + 1 = 2 \\
 2y_1 + 0y_2 + y_3 &= 5 \quad \rightarrow \quad y_3 = 5 - 2y_1 = 3
 \end{aligned}$$

$$Ux = y \longrightarrow \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$x_1, x_2$ , dan  $x_3$  dihitung dengan teknik penyulihan mundur:

$$3x_3 = 3 \rightarrow x_3 = 1$$

$$2x_2 + 0x_3 = 2 \rightarrow x_2 = 1$$

$$x_1 + x_2 - x_3 = 1 \rightarrow x_1 = 1$$

Jadi, solusi sistem persamaan linier di atas adalah  $x = (1, 1, 1)^T$ . ■

Jika diamati elemen segitiga bawah pada matriks  $U$  semuanya bernilai nol, sehingga ruang yang tidak terpakai itu dapat dipakai untuk menyimpan elemen matriks  $L$ . Elemen diagonal matriks  $L$  seluruhnya 1, jadi tidak perlu disimpan (*default*). Dengan demikian, penyimpanan elemen  $L$  dan  $U$  pada satu matriks dapat menghemat penggunaan memori. Selain itu, matriks  $A$  hanya dipakai sekali untuk memperoleh  $L$  dan  $U$ , sesudah itu tidak dipakai lagi. Dengan demikian, setelah  $L$  dan  $U$  diperoleh, elemennya dapat dipindahkan ke dalam  $A$ . Karena alasan ini, maka metode dekomposisi  $LU$  dinamakan juga metode kompaksi memori.

## 4.6 Determinan

Pada pembahasan matriks balikan kita telah menyinggung sedikit mengenai determinan. Menghitung determinan matriks  $2 \times 2$  sangat mudah dan selalu diajarkan di sekolah menengah. Misalkan  $A$  adalah matriks

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

maka determinan matriks  $A$  adalah

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Begitupun menghitung determinan untuk matriks  $3 \times 3$ ,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

maka determinannya dihitung dengan aturan Cramer:

$$\begin{aligned} \det(A) &= \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \\ &= a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ &= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}) \end{aligned}$$

Menghitung determinan untuk matriks  $n \times n$  dengan aturan Cramer menjadi tidak praktis lagi. Metode eliminasi Gauss dapat diterapkan untuk menghitung determinan matriks  $n \times n$ . Determinannya dapat dihitung setelah ia ditransformasi menjadi matriks segitiga atas  $U$ . Pertama-tama kita lihat dulu dua hukum penting determinan [NAK92]:

Hukum 1:  $\det(BC) = \det(B) \times \det(C)$

yaitu, determinan dari perkalian dua buah matriks sama dengan perkalian determinan masing-masing matriks.

Hukum 2:  $\det(M) =$  hasil kali semua elemen diagonal  $M$  jika  $M$  adalah matriks segitiga atas atau matriks segitiga bawah.

Jadi, jika semua elemen diagonal matriks adalah satu, maka determinannya sama dengan satu. Dalam menghitung determinan, pertimbangkan dua kasus berikut berikut: (i) bila eliminasi Gauss-nya tanpa *pivoting* dan (ii) bila eliminasi Gauss-nya dengan *pivoting*.

**Kasus 1: Bila eliminasi Gauss tidak menerapkan tatancang *pivoting*.**

Jika *pivoting* tidak diterapkan, determinan matriks  $A$  adalah

$$\det(A) = \det(LU) = \det(L) \times \det(U) = \det(U) = u_{11} u_{22} u_{33} \dots u_{nn} \quad (\text{P.4.15})$$

yang dalam hal ini  $\det(L) = 1$  sebab semua elemen diagonal  $L$  adalah satu.

**Kasus 2: Bila eliminasi Gauss menerapkan tatancang *pivoting*.**

Tatancang *pivoting* mengakibatkan pertukaran baris. Dekomposisi  $LU$  dengan *pivoting* setara dengan mengerjakan dua proses terpisah berikut:

- (1) Transformasikan matriks  $A$  menjadi matriks  $A'$  dengan cara permutasi baris-baris matriks (sama dengan mengalikan  $A$  dengan matriks permutasi  $P$ ),

$$A' = PA \quad \text{atau setara dengan} \quad A = P^{-1} A' \quad (\text{P.4.16})$$

- (2) Dekomposisi  $A'$  menjadi  $LU$  tanpa *pivoting*

$$A' = LU$$

Dari (1) dan (2),  $L$  dan  $U$  dihubungkan dengan  $A$  oleh

$$A = P^{-1} A' = P^{-1} LU \quad (\text{P.4.17})$$

Determinan  $A$  dapat ditulis sebagai

$$\begin{aligned} \det(A) &= \det(P^{-1}) \times \det(L) \times \det(U) \\ &= \det(P^{-1}) \times 1 \times \det(U) \\ &= \det(P^{-1}) \times \det(U) \\ &= \alpha \det(U) \end{aligned}$$

yang dalam hal ini  $\alpha = \det(P^{-1}) = -1$  atau  $1$  bergantung pada apakah *pivoting* sejumlah bilangan ganjil atau genap. Jika *pivoting* dilakukan sejumlah  $p$  kali, maka  $\alpha$  dapat ditulis sebagai:

$$\alpha = (-1)^p$$

$\alpha$  bernilai  $1$  untuk  $p$  genap dan  $-1$  untuk  $p$  ganjil. Karena itu,

$$\det(A) = (-1)^p \det(U) = (-1)^p u_{11} u_{22} u_{33} \dots u_{nn} \quad (\text{P.4.18})$$

**Contoh 4.13**

Hitung determinan matriks  $A$  berikut:

$$A = \begin{bmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ -2 & 3 & -1 \end{bmatrix}$$

**Penyelesaian:**

$$\begin{bmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ -2 & 3 & -1 \end{bmatrix} \begin{array}{l} R_2 - \frac{4}{2}R_1 \\ R_3 - \frac{-2}{2}R_1 \end{array} \begin{bmatrix} 2 & 3 & -1 \\ 0 & -2 & -1 \\ 1 & 0 & -2 \end{bmatrix} \begin{array}{l} R_3 - \frac{6}{-2}R_2 \end{array} \begin{bmatrix} 2 & 3 & -1 \\ 0 & -2 & -1 \\ 0 & 0 & -5 \end{bmatrix}$$

Tidak ada proses *pivoting* selama eliminasi Gauss, maka

$$\det(A) = (2)(-2)(-5) = 20$$

■

**Contoh 4.14**

Hitung determinan matriks berikut

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 6 & 0 \\ 2 & 8 & 4 \end{bmatrix}$$

**Penyelesaian:**

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 6 & 0 \\ 2 & 8 & 4 \end{bmatrix} \begin{array}{l} R_2 - \frac{3}{1}R_1 \\ \sim \\ R_3 - \frac{2}{1}R_1 \end{array} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & -3 \\ 0 & 4 & 2 \end{bmatrix} \begin{array}{l} R_3 \Leftrightarrow R_2 \\ (*) \end{array} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 4 & 2 \\ 0 & 0 & -3 \end{bmatrix}$$

*Pivoting* diterapkan satu kali ( $p = 1$ ), sehingga determinan matriks  $A$  adalah

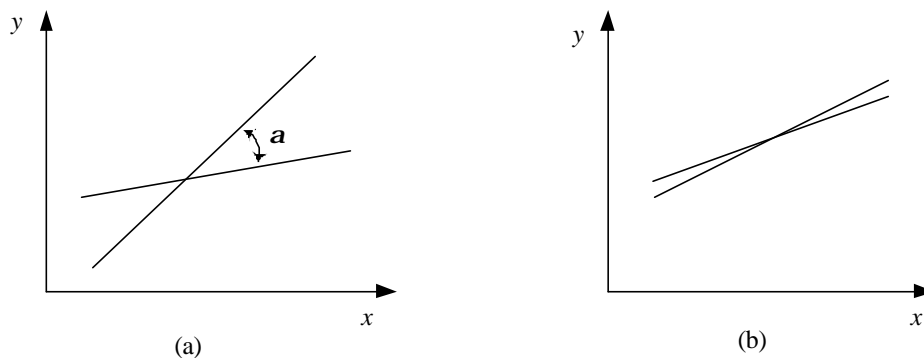
$$\det(A) = (-1)^1 (1)(4)(-3) = 12$$

■

## 4.7 Kondisi Buruk

Matriks  $A$  dikatakan berkondisi buruk (*ill condition*) jika terdapat sebuah vektor kolom  $b$  sehingga untuk perubahan kecil  $A$  atau  $b$  akan menghasilkan perubahan besar pada solusi  $x = A^{-1}b$ . Sistem  $Ax = b$  dikatakan berkondisi buruk bila  $A$  berkondisi buruk. Apabila sistem  $Ax = b$  berkondisi buruk, hasil perhitungannya mempunyai galat yang besar.

Sebagai contoh, dua persamaan linier dengan dua peubah yang tidak diketahui merepresentasikan dua buah garis lurus. Sistem berkondisi buruk jika dan hanya jika sudut  $\alpha$  antara kedua garis kecil, yaitu jika dan hanya jika kedua garis hampir sejajar. Perubahan kecil pada koefisien dapat menyebabkan pergeseran yang besar pada titik potong kedua garis (Gambar 4.3) [KRE88]. Untuk sistem persamaan yang lebih besar situasi tersebut pada prinsipnya sama, namun sayangnya tidak ada tafsiran geometrinya.



**Gambar 4.3** (a) sistem berkondisi baik dan (b) sistem berkondisi buruk

Sebagai contoh, tinjau sistem persamaan linier berikut

$$\begin{aligned} \text{(i)} \quad & x_1 + 2x_2 = 10 \\ & 1.1x_1 + 2x_2 = 10.4 \end{aligned}$$

yang mempunyai solusi sejati  $x_1 = 4$  dan  $x_2 = 3$ . Jika sekarang  $a_{21} = 1.1$  diubah menjadi 1.05,

$$\begin{aligned} \text{(ii)} \quad & x_1 + 2x_2 = 10 \\ & 1.05x_1 + 2x_2 = 10.4 \end{aligned}$$

ternyata solusinya jauh berbeda, yaitu  $x_1 = 8$  dan  $x_2 = 1$ . Penambahan sebesar  $\mathbf{e}$  pada koefisien 1.1 dapat dinyatakan sebagai berikut:

$$\begin{aligned}x_1 + 2x_2 &= 10 \\(1.1 + \mathbf{e})x_1 + 2x_2 &= 10.4\end{aligned}$$

yang mempunyai solusi

$$\begin{aligned}x_1 &= \frac{0.4}{0.1 + \mathbf{e}} \\x_2 &= \frac{0.6 + 10\mathbf{e}}{0.2 + 2\mathbf{e}}\end{aligned}$$

Solusi ini memperlihatkan bahwa sistem berkondisi buruk sebab perubahan kecil  $\mathbf{e}$  menghasilkan perubahan besar pada solusi SPL. Pada contoh di atas,  $\mathbf{e} = -0.05$ , sehingga  $x_1 = 0.4/(0.1 - 0.05) = 8$  dan  $x_2 = (0.6 - 10 \times 0.05)/(0.2 - 2 \times 0.05) = 1$ .

Misalkan  $\hat{x}$  adalah solusi hampiran dari sistem

$$A \hat{x} = b \quad (\text{P.4.19})$$

Terhadap solusi hampiran ini terdapat sisa (residu) sebesar

$$r = b - A \hat{x}$$

Di sini

$$A \hat{x} = b - r \quad (\text{P.4.20})$$

Kurangi (P.4.19) dengan (P.4.20):

$$A(\hat{x} - x) = r \quad (\text{P.4.21})$$

Orang mungkin berpikir bahwa sisa  $r$  yang kecil menandakan bahwa  $\hat{x}$  lebih dekat ke  $x$ . Tetapi, kesimpulan ini ternyata salah. Penyulihan kembali solusi hampiran ke SPL yang asli tidak dapat memberi petunjuk bahwa sistem berkondisi buruk. Bila  $x_1 = 8$  dan  $x_2 = 1$  disulih kembali ke dalam SPL (i):

$$\begin{aligned}8 + 2(1) &= 10 \\1.1(8) + 2(1) &= 10.8\end{aligned}$$

Residunya adalah

$$r = b - A\hat{x} = \begin{bmatrix} 10 \\ 10.4 \end{bmatrix} - \begin{bmatrix} 10 \\ 10.8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.4 \end{bmatrix}$$

ternyata sisa  $r$  cukup kecil meskipun  $x_1 = 8$  dan  $x_2 = 1$  bukan jawaban yang benar untuk masalah semula.

Contoh lainnya, tinjau sistem persamaan linier  $Ax = b$  dengan

$$A = \begin{bmatrix} 3.02 & -1.05 & 2.53 \\ 4.33 & 0.56 & -1.78 \\ -0.83 & -0.54 & 1.47 \end{bmatrix} \quad \text{dan} \quad b = \begin{bmatrix} -1.61 \\ 7.23 \\ -3.38 \end{bmatrix}$$

Solusi sejatinya adalah  $x = (1, 2, -1)^T$ . Solusi hampirannya, bila dihitung dengan metode eliminasi Gauss, adalah  $\hat{x} = (0.880, -2.35, -2.66)^T$ . Bila 3.02 pada matriks  $A$  diubah menjadi 3.00 diperoleh solusi  $\hat{x} = (0.07968, 4.75637, 0.05856)^T$ . Kita menyimpulkan bahwa SPL tersebut berkondisi buruk. Jika kita mengalikan  $A$  dengan  $x$  dengan  $x$  adalah solusi sejati  $x = (1, 2, -1)^T$ , kita peroleh

$$Ax = (-1.61, 7.23, -3.38)^T = b$$

tetapi bila kita hitung dengan solusi hampiran  $\hat{x} = (0.880, -2.35, -2.66)^T$  kita peroleh

$$A\hat{x} = (-1.6047, 7.2292, -2.66),$$

yang sangat dekat ke  $b$ . Penyulihan kembali solusi ke dalam sistem persamaan ternyata tidak dapat dijadikan petunjuk bahwa sistem berkondisi buruk.

Beberapa ukuran untuk kondisi buruk telah dikemukakan para ahli numerik, antara lain  $|\det(A)|$  sangat kecil dibandingkan dengan nilai maksimum  $|a_{ij}|$  dan  $|b_i|$ . Misalnya, SPL dengan dua persamaan dapat ditulis sebagai:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 = b_1 &\rightarrow x_2 = \frac{a_{11}}{a_{12}}x_1 + \frac{b_1}{a_{12}} \\ a_{21}x_1 + a_{22}x_2 = b_2 &\rightarrow x_2 = \frac{a_{21}}{a_{22}}x_1 + \frac{b_2}{a_{22}} \end{aligned}$$



Bila gradien kedua garis tersebut *hampir sama*, maka:

$$\frac{-a_{11}}{a_{12}} \approx \frac{-a_{21}}{a_{22}}$$

dan apabila kedua ruas dikali silang:

$$a_{11}a_{22} \approx a_{12}a_{21}$$

atau dapat ditulis sebagai

$$a_{11}a_{22} - a_{12}a_{21} \approx 0$$

yang dalam hal ini  $a_{11}a_{22} - a_{12}a_{21}$  adalah determinan matriks  $A$  pada SPL di atas. Sistem persamaan linier berkondisi buruk bila determinan matriks  $A$  hampir nol. Jika  $\det(A) = 0$ , maka gradien kedua garis tersebut sama, yang berarti SPL tidak mempunyai jawaban yang unik. Determinan matriks  $A$  pada contoh di atas adalah  $(1)(2) - 2(1.1) = 2 - 2.2 = -0.2$ , yang relatif lebih dekat ke nol.

#### Contoh 4.15

Tentukan solusi  $Ax = b$  berikut

$$A = \begin{bmatrix} 3.02 & -1.05 & 2.53 \\ 4.33 & 0.56 & -1.78 \\ -0.83 & -0.54 & 1.47 \end{bmatrix} \quad b = \begin{bmatrix} -1.61 \\ 7.23 \\ -3.38 \end{bmatrix}$$

#### Penyelesaian:

Matriks akhir hasil eliminasi Gauss-nya sampai 3 angka bena adalah

$$\left[ \begin{array}{ccc|c} 4.33 & 0.56 & -1.78 & 7.23 \\ 0 & -1.44 & 3.77 & -6.65 \\ 0 & 0 & \mathbf{-0.00362} & 0.00962 \end{array} \right]$$

└────────── sangat kecil, rawan pembulatan

Solusi hampirannya adalah  $x = (0.880, -2.53, -2.66)^T$  bandingkan dengan solusi sejatinya,  $x = (1, 2, -1)^T$

Perbedaan kedua solusi ini tidak mengherankan apabila kita hitung determinan matriks  $A$ ,

$$\det(A) = (4.33)(-1.44)(-0.00362) = -0.0226$$

yang nilainya sangat kecil (mendekati nol), yang berarti sistem berkondisi buruk. Bila koefisien  $a_{11}$  diubah dari 3.02 menjadi 3.10 memberikan solusi

$$x = (0.05856, 4.75637, 1.07968)^T$$

Solusi yang lebih baik dapat kita peroleh bila menggunakan bilangan berketelitian yang lebih tinggi, misalnya sampai empat angka bena sebagai berikut:  $x_1 = 0.9998$ ,  $x_2 = 1.9995$ ,  $x_3 = -1.002$ . ■

Sukar dirinci harus *seberapa dekat* determinan ke nol untuk menunjukkan adanya kondisi buruk. Ini diperumit dengan kenyataan bahwa determinan dapat diubah dengan mengalikan satu atau beberapa persamaan dengan suatu faktor skala tanpa mengubah penyelesaiannya. Akibatnya, determinan merupakan nilai yang nisbi yang dipengaruhi oleh besarnya koefisien [CHA91]. Ini diperlihatkan oleh contoh berikut.

#### **Contoh 4.16**

Tentukan determinan matriks  $A$  pada SPL berikut

$$\begin{aligned}x_1 + 2x_2 &= 10 \\ 1.1x_1 + 2x_2 &= 10.4\end{aligned}$$

bila

- (i) SPL seperti apa adanya,
- (ii) kedua persamaan dikali 10.

#### **Penyelesaian:**

- (i) SPL apa adanya  
Determinannya,

$$\det(A) = (1)(2) - (1.1)(2) = -0.2$$

yang dekat ke nol, karena itu SPL berkondisi buruk.

- (ii) Kedua persamaan dikali 10,

$$\begin{aligned}10x_1 + 20x_2 &= 100 \\ 11x_1 + 20x_2 &= 104\end{aligned}$$

Determinannya,

$$\det(A) = (10)(20) - (11)(20) = -20.$$

yang ternyata menjadi lebih besar. Meskipun determinannya besar, namun SPL tetap berkondisi buruk sebab perkalian dengan skala tidak mempengaruhi penyelesaiannya secara grafis. ■

Contoh 4.16 (ii) di atas memperlihatkan bahwa ukuran determinan sukar dikaitkan dengan kondisi buruk. Kesukaran ini dapat diatasi bila SPL dinormalkan sedemikian sehingga koefisien terbesar pada tiap baris persamaan sama dengan 1.

#### **Contoh 4.17**

Normalkan SPL pada Contoh 4.16, lalu hitung determinan matriks  $A$ .

##### **Penyelesaian:**

SPL dinormalkan dengan cara membagi tiap baris dengan koefisien terbesar pada baris itu sehingga koefisien maksimumnya = 1

$$\begin{aligned} 0.5x_1 + x_2 &= 5 \\ 0.55x_1 + x_2 &= 5.2 \end{aligned}$$

Determinannya,

$$\det(A) = (0.5)(1) - (1)(0.55) = -0.55$$

yang dekat ke nol, karena itu berkondisi buruk. ■

Pada sistem yang berkondisi baik, penormalan tetap menghasilkan determinan yang jauh dari nol. Hal ini ditunjukkan pada Contoh 4.17 berikut.

#### **Contoh 4.18**

Hitung determinan matriks  $A$  pada SPL

$$\begin{aligned} 3x_1 + 2x_2 &= 18 \\ -x_1 + 2x_2 &= 2 \end{aligned}$$

bila (i) SPL apa adanya dan bila (ii) SPL dinormalkan[CHA91]

##### **Penyelesaian:**

(i) SPL apa adanya

Determinannya,

$$\det(A) = (3)(2) - 2(-1) = 8$$

yang nilainya jauh dari nol, karena itu berkondisi baik.

(ii) SPL dinormalkan

Penormalan menghasilkan

$$\begin{aligned}x_1 + 0.667x_2 &= 6 \\ -0.5x_1 + x_2 &= 1\end{aligned}$$

Determinannya,

$$\det(A) = (1)(1) - (0.667)(-0.5) = 1.333$$

yang nilainya jauh dari nol, karena itu berkondisi baik. ■

Selain dengan menghitung determinan, ada beberapa ukuran lain yang dapat digunakan untuk memeriksa apakah sistem persamaan linier berkondisi buruk [NAK92]:

1. Mencoba mengubah koefisien dengan perubahan yang cukup kecil, lalu membandingkan solusinya dengan solusi sistem persamaan yang belum diubah. Jika perubahan kecil koefisien menghasilkan solusi yang sangat berbeda dengan solusi sebelum perubahan, maka sistem berkondisi buruk.
2. Membandingkan solusi berketelitian tunggal dengan solusi berketelitian ganda. Jika kedua solusinya berbeda berarti sistem berkondisi buruk.
3. Skalakan  $A$  sehingga elemen terbesar dalam masing-masing baris adalah 1 dan kemudian hitung  $A^{-1}$ . Jika elemen  $A^{-1}$  beberapa orde lebih besar daripada elemen matriks yang diskala semula, maka sistem berkondisi buruk.
4. Menghitung  $\det(A) \times \det(A^{-1})$  apakah berbeda jauh dari 1. Jika ya, berarti sistem berkondisi buruk.
5. Menghitung  $(A^{-1})^{-1}$  apakah berbeda “jauh” dari  $A$ . Jika ya, berarti sistem berkondisi buruk.
6. Menghitung  $AA^{-1}$  apakah berbeda “jauh” dari matriks  $I$ . Jika ya, berarti sistem berkondisi buruk.
7. Menghitung  $(A^{-1})(A^{-1})^{-1}$  apakah berbeda “jauh” dari matriks  $I$ . Jika ya, berarti sistem berkondisi buruk.

Walaupun terdapat beragam cara untuk memeriksa kondisi sistem, akan lebih disukai mendapatkan bilangan tunggal yang dapat berlaku sebagai petunjuk adanya kondisi buruk. Bilangan tersebut dinamakan **bilangan kondisi matriks**.

## 4.8 Bilangan Kondisi Matriks

Bilangan kondisi matriks dinyatakan sebagai :

$$\text{Cond}(A) = \|A\| \|A^{-1}\| \quad (\text{P.4.22})$$

yang dalam hal ini  $\|A\|$  adalah norma (*norm*) tak-hingga ( $\infty$ ) matriks  $A$ , yang didefinisikan sebagai:

$$\|A\| = \|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

Sebagai tambahan, perlu kita ketahui sifat-sifat norma matriks berikut :

- (a)  $|A| \geq 0$  dan  $|A| = 0$  jika dan hanya jika  $A = 0$
- (b)  $|kA| = k |A|$
- (c)  $|A + B| \leq |A| + |B|$
- (d)  $|AB| \leq |A| |B|$

### Contoh 4.19

Hitung bilangan kondisi matriks  $A$  berikut

$$A = \begin{bmatrix} 3.02 & -1.05 & 2.53 \\ 4.33 & 0.56 & -1.78 \\ -0.83 & -0.54 & 1.47 \end{bmatrix}$$

### Penyelesaian:

Tentukan terlebih dahulu matriks balikkannya,

$$A^{-1} = \begin{bmatrix} 5.661 & -7.273 & -18.55 \\ 200.5 & -268.3 & -669.9 \\ 76.85 & -102.6 & -255.9 \end{bmatrix}$$

maka dapat dihitung

$$\begin{aligned}\|A\| &= |4.33| + |0.56| + |1.78| = 6.07 \\ \|A^{-1}\| &= |200.5| + |-268.3| + |-669.9| = 1138.7\end{aligned}$$

sehingga bilangan kondisi matriks  $A$  adalah

$$\text{cond}(A) = (66.7)(1138.7) = 7595$$

■

Bagaimana kita menggunakan bilangan kondisi ini untuk menentukan apakah sistem berkondisi buruk atau berkondisi baik? Ralston dan Rabinowitz (1978) dan Gerald dan Wheatley (1984), memperkenalkan penggunaan bilangan kondisi matriks untuk menjelaskan kasus sistem berkondisi buruk sebagai berikut. Seperti diketahui bahwa kondisi buruk disebabkan oleh kesalahan dalam pengukuran data model atau karena kesalahan pembulatan. Misalkan bahwa kesalahan dalam pengukuran parameter SPL menyebabkan kesalahan pada koefisien  $a_{i,j}$ , sehingga SPL dipecahkan sebagai  $(A + E) \hat{x} = b$ , yang dalam hal ini  $\hat{x}$  menyatakan solusi SPL yang mengandung galat. Misalkan  $\hat{A} = A + E$  menyatakan koefisien matriks yang mengandung kesalahan. Kita ingin menghitung berapa besar selisih  $x - \hat{x}$ .

Dengan menggunakan  $Ax = b$  dan  $\hat{A} \hat{x} = b$ , dapat kita tulis :

$$\begin{aligned}x &= A^{-1} b = A^{-1} (Ax) = A^{-1} (A + \hat{A} - A) \hat{x} \\ &= [I + A^{-1}(\hat{A} - A)] \hat{x} \\ &= \hat{x} + A^{-1}(\hat{A} - A) \hat{x}\end{aligned}$$

Karena  $\hat{A} - A = E$ , maka

$$x - \hat{x} = A^{-1} E \hat{x} \quad (\text{P.4.23})$$

Dengan menggunakan norma, kita peroleh :

$$\|x - \hat{x}\| \leq \|A^{-1}\| \|E\| \|\hat{x}\| = \|A^{-1}\| \|A\| \frac{\|E\|}{\|A\|} \|\hat{x}\|$$

sehingga

$$\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \leq (\text{bilangan kondisi}) \times \frac{\|E\|}{\|A\|} \quad (\text{P.4.24})$$

Persamaan (P.4.24) ini menyatakan bahwa norma galat relatif solusi SPL dapat sebesar norma galat nisbi koefisien matriks  $A$  dikali dengan bilangan kondisi. *Jadi, jika bilangan kondisi matriks  $A$  besar, maka galat relatif solusi SPL juga akan besar. Sebaliknya, jika bilangan kondisinya kecil, galat relatif solusi SPL juga kecil.* Misalnya jika koefisien  $A$  diketahui teliti sampai  $t$  angka bena (yakni, galat pembulatan berorde  $10^{-t}$ ) dan bilangan kondisi  $A = 10^c$ , penyelesaian  $x$  akan teliti sampai  $t - c$  angka bena (galat pembulatan  $\cong 10^{c-t}$ ). Misalnya, jika koefisien  $A$  diketahui hanya 4 angka bena dan bilangan kondisi 1000, vektor  $x$  hanya mempunyai ketelitian satu angka signifikan

**TEOREMA 4.1.** Sistem persamaan linier  $Ax = b$  yang bilangan kondisinya kecil menyatakan sistem berkondisi baik. Bilangan kondisi besar menandakan bahwa sistem berkondisi buruk. [KRE88].

Sistem pada Contoh 4.19 adalah contoh sistem yang berkondisi buruk, karena bilangan kondisinya besar.

Dalam praktek,  $A^{-1}$  tidak diketahui, sehingga untuk menghitung bilangan kondisi matriks  $A$  kita harus menaksir  $|A^{-1}|$ . Metode untuk penaksiran ini tidak dijelaskan di sini.

Di dalam banyak literatur disebutkan bahwa matriks Hilbert adalah contoh matriks yang berkondisi buruk. Bentuk umum matriks Hilbert orde  $n$  adalah

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1} \end{bmatrix}$$

Contohnya, untuk  $n = 3$  matriks Hilbertnya adalah

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

Norma matriks  $H$  adalah

$$\|H\|_{\infty} = |1| + |\frac{1}{2}| + |\frac{1}{3}| = \frac{11}{6}$$

Matriks balikkannya adalah,

$$H^{-1} = \begin{bmatrix} 9 & -18 & 10 \\ -36 & 96 & -60 \\ 30 & -90 & 60 \end{bmatrix}$$

Elemen matriks  $H^{-1}$  jauh lebih besar daripada matriks  $H$ , hal ini menandakan bahwa matriks  $H$  berkondisi buruk. Dapat dihitung norma matriks  $H^{-1}$

$$\|H^{-1}\|_{\infty} = |36| + |96| + |60| = 192$$

Sehingga bilangan kondisi matriks  $H$  adalah

$$\text{cond}(H) = \|H\|_{\infty} \|H^{-1}\|_{\infty} = \frac{11}{6} \times 192 = 352$$

yang nilainya sangat besar, sehingga benarlah bahwa matriks  $H$  berkondisi buruk.

Sekarang kita buktikan mengapa penyulihan kembali solusi ke dalam SPL tidak dapat dijadikan petunjuk bahwa sistem berkondisi buruk. Tinjau kembali persamaan residu

$$r = b - A \hat{x} . \quad (\text{P.4.25})$$

Pada sistem yang berkondisi buruk nilai  $r$  sangat kecil, sehingga kita dapat terkecoh dengan menganggap sistem berkondisi baik. Contoh-contoh sebelum ini memperlihatkan bahwa  $r$  bukanlah ukuran yang bagus untuk galat ( $e = x - \hat{x}$ ) pada sistem yang berkondisi buruk. Bila  $x$  adalah solusi eksak maka  $r = 0$ , atau

$$0 = b - Ax \quad (\text{P.4.26})$$

Kurangi (P.4.25) dengan (P.4.26):



$$\begin{aligned}
(b - A \hat{x}) - (b - Ax) &= r \Leftrightarrow \\
-A \hat{x} + Ax &= r \Leftrightarrow \\
A(x - \hat{x}) &= r \Leftrightarrow \\
A e &= r
\end{aligned} \tag{P.4.27}$$

atau

$$e = A^{-1} r \tag{P.4.28}$$

Pada sistem yang berkondisi buruk, elemen matriks  $A^{-1}$  relatif besar dibandingkan elemen-elemen  $A$ . Dari (P.4.28) terlihat bahwa bila elemen  $A^{-1}$  relatif sangat besar dibandingkan nilai  $r$  yang kecil, maka  $e$  akan besar. Jadi, residu  $r$  yang kecil tidak menjamin solusi yang diperoleh adalah benar. Karena itu digunakan hubungan antara nilai mutlak galat solusi dengan nilai mutlak residu. Dari persamaan (P.4.25) kita peroleh :

$$r = b - A \hat{x} = Ax - A \hat{x} = A(x - \hat{x}) = Ae \tag{P.4.29}$$

Disini,

$$e = A^{-1} r \tag{P.4.30}$$

Dari sifat-sifat norma matriks di atas, maka norma untuk persamaan (P.5.27) , dengan menerapkan sifat (d), dapat kita tulis :

$$|e| \leq |A^{-1}| |r| \tag{P.4.31}$$

Dari  $r = Ae$ , kita juga punya  $|r| \leq |A| |e|$ , yang bila digabung dengan persamaan (P.5.32) memberikan

$$\frac{\|r\|}{\|A\|} \leq |e| \leq |A^{-1}| |r| \tag{P.4.32}$$

Dengan menggunakan cara yang sama untuk  $Ax = b$  dan  $x = A^{-1} b$ , kita peroleh

$$\frac{\|b\|}{\|A\|} \leq |x| \leq |A^{-1}| |b| \tag{P.4.33}$$

Dari persamaan (P.4.32) dan (P.4.33) kita dapatkan hubungan yang penting :

$$\frac{1}{\|A\| \|A^{-1}\| \|b\|} \leq \frac{\|e\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|r\|}{\|b\|} \quad (\text{P.4.34})$$

atau

$$\frac{1}{\text{bil. kondisi}} \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} \leq (\text{bil. kondisi}) \frac{\|r\|}{\|b\|} \quad (\text{P.4.35})$$

Persamaan (P.4.35) memperlihatkan bahwa galat relatif dalam menghitung solusi  $x$  dapat sebesar residu relatif dikali dengan bilangan kondisi. Tentu saja juga akan sekecil residu relatif dibagi dengan bilangan kondisi. Karena itu, jika bilangan kondisi besar, residu  $r$  hanya memberikan sedikit informasi tentang ketelitian  $x$ . Sebaliknya, jika bilangan kondisi dekat ke 1, residu nisbi memberikan ukuran galat  $x$  yang bagus.

Rice pada tahun 1983 menyarankan sebuah cara lain untuk menilai kondisi SPL: jalankan pemecahan SPL yang sama pada dua kompuler yang berbeda (atau pada dua mesin yang berbeda). Karena kode yang dihasilkan kemungkinan besar menerapkan perhitungannya secara berbeda. Kondisi buruk akan jelas terlihat dari eksperimen seperti itu [CHA91].

## 4.9 Metode Lelaran Untuk Menyelesaikan SPL

Metode eliminasi Gauss melibatkan banyak galat pembulatan. Galat pembulatan yang terjadi pada eliminasi Gauss (maupun eliminasi Gauss-Jordan) dapat menyebabkan solusi yang diperoleh “jauh” dari solusi sebenarnya. Gagasan metoda lelaran pada pencarian akar persamaan nirlanjar dapat juga diterapkan untuk menyelesaikan SPL. Dengan metode lelaran, galat pembulatan dapat diperkecil, karena kita dapat meneruskan lelaran sampai solusinya seteliti mungkin, sesuai dengan batas galat yang kita perbolehkan. Dengan kata lain, besar galat dapat dikendalikan sampai batas yang bisa diterima.

Jika metode eliminasi Gauss dan variasi-variasinya serta metode dekomposisi  $LU$  dinamakan **metode langsung** (*direct*) -karena solusi SPL diperoleh tanpa lelaran- maka metode lelaran dinamakan **metode tidak langsung** (*indirect*) atau **metode iteratif**.

Tinjau kembali sistem persamaan linier

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n \end{aligned}$$

Dengan syarat  $a_{kk} \neq 0$ ,  $k = 1, 2, \dots, n$ , maka persamaan larannya dapat ditulis sebagai

$$\begin{aligned} x_1^{(k+1)} &= \frac{b_1 - a_{12}x_2^{(k)} - \dots - a_{1n}x_n^{(k)}}{a_{11}} \\ x_2^{(k+1)} &= \frac{b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \dots - a_{2n}x_n^{(k)}}{a_{22}} \\ \vdots & \\ x_n^{(k+1)} &= \frac{b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \dots - a_{nn-1}x_{n-1}^{(k)}}{a_{nn}} \end{aligned} \quad (\text{P.4.36})$$

dengan  $k = 0, 1, 2, \dots$

Lelaran dimulai dengan memberikan tebakan awal untuk  $x$ ,

$$x_0 = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_n^{(0)} \end{bmatrix}$$

Sebagai kondisi berhenti lelarnya, dapat digunakan pendekatan galat relatif

$$\left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| < \epsilon \quad \text{untuk semua } i = 1, 2, 3, \dots, n$$

Syarat cukup agar lelarnya konvergen adalah sistem **dominan secara diagonal**:

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|, \quad i = 1, 2, 3, \dots, n$$

Syarat cukup ini berarti bahwa agar lelarannya konvergen, cukup dipenuhi syarat itu. Jika syarat tersebut dipenuhi, kekonvergenan dijamin. Meskipun sistem tidak dominan secara diagonal, lelarannya masih mungkin konvergen (lihat kembali makna syarat cukup pada upabab 3.3). Kekonvergenan juga ditentukan oleh pemilihan tebakan awal. Tebakan awal yang terlalu jauh dari solusi sejatinya dapat menyebabkan lelaran divergen.

Sebagai contoh, SPL berikut

$$\begin{aligned} 3x_1 + x_2 - x_3 &= 1 \\ 2x_1 + 4x_2 + x_3 &= 5 \\ -x_1 + 5x_2 + 8x_3 &= 5 \end{aligned}$$

dominan secara diagonal, karena

$$\begin{aligned} |3| &> |1| + |-1| \\ |4| &> |2| + |1| \\ |8| &> |-1| + |5| \end{aligned}$$

karena itu lelarannya pasti konvergen.

Ada dua metode lelaran yang akan kita bahas di sini:

1. Metode lelaran Jacobi
2. Metode lelaran Gauss-Seidel

### 4.9.1 Metode Lelaran Jacobi

Persamaan lelarannya adalah seperti yang ditulis di atas. Misalkan diberikan tebakan awal  $x^{(0)}$ :

$$x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^T$$

Prosedur lelaran untuk lelaran pertama, kedua, dan seterusnya adalah sebagai berikut:

Lelaran pertama:

$$\begin{aligned}
 x_1^{(1)} &= \frac{b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - \dots - a_{1n}x_n^{(0)}}{a_{11}} \\
 x_2^{(1)} &= \frac{b_2 - a_{21}x_1^{(0)} - a_{23}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}}{a_{22}} \\
 &\vdots \\
 x_n^{(1)} &= \frac{b_n - a_{n1}x_1^{(0)} - a_{n2}x_2^{(0)} - \dots - a_{nn-1}x_{n-1}^{(0)}}{a_{nn}}
 \end{aligned}$$

Lelaran kedua:

$$\begin{aligned}
 x_1^{(2)} &= \frac{b_1 - a_{12}x_2^{(1)} - a_{13}x_3^{(1)} - \dots - a_{1n}x_n^{(1)}}{a_{11}} \\
 x_2^{(2)} &= \frac{b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(1)} - \dots - a_{2n}x_n^{(1)}}{a_{22}} \\
 &\vdots \\
 x_n^{(2)} &= \frac{b_n - a_{n1}x_1^{(1)} - a_{n2}x_2^{(1)} - \dots - a_{nn-1}x_{n-1}^{(1)}}{a_{nn}}
 \end{aligned}$$

Rumus umum :

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)}}{a_{ii}}, k = 0, 1, 2, \dots \quad (\text{P.4.37})$$

## 4.9.2 Metode Lelaran Gauss-Seidel

Kecepatan konvergen pada lelaran Jacobi dapat dipercepat bila setiap harga  $x_i$  yang baru dihasilkan segera dipakai pada persamaan berikutnya untuk menentukan harga  $x_{i+1}$  yang lainnya.

Lelaran pertama:

$$x_1^{(1)} = \frac{b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - a_{14}x_4^{(0)}}{a_{11}}$$

$$x_2^{(1)} = \frac{b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)} - a_{24}x_4^{(0)}}{a_{22}}$$

$$x_3^{(1)} = \frac{b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} - a_{34}x_4^{(0)}}{a_{33}}$$

$$x_4^{(1)} = \frac{b_4 - a_{41}x_1^{(1)} - a_{42}x_2^{(1)} - a_{43}x_3^{(1)}}{a_{44}}$$

Lelaran kedua:

$$x_1^{(2)} = \frac{b_1 - a_{12}x_2^{(1)} - a_{13}x_3^{(1)} - a_{14}x_4^{(1)}}{a_{11}}$$

$$x_2^{(2)} = \frac{b_2 - a_{21}x_1^{(2)} - a_{23}x_3^{(1)} - a_{24}x_4^{(1)}}{a_{22}}$$

$$x_3^{(2)} = \frac{b_3 - a_{31}x_1^{(2)} - a_{32}x_2^{(2)} - a_{34}x_4^{(1)}}{a_{33}}$$

$$x_4^{(2)} = \frac{b_4 - a_{41}x_1^{(2)} - a_{42}x_2^{(2)} - a_{43}x_3^{(2)}}{a_{44}}$$

Rumus umum:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^n a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)}}{a_{ii}}, k = 0, 1, 2, \dots$$

**Program 4.5** Metode Lelaran Gauss-Seidel (tanpa penanganan kasus divergen)

```

procedure Gauss_Seidel(A : matriks; b: vektor; n:integer;
    var x : vektor);
{Menghitung solusi SPL Ax = b dengan metode Gauss-Seidel. Diandaikan
lelaran selalu konvergen
K.Awal : A dan b sudah terdefinisi harganya; x sudah berisi vektor
tebakan awal
K.Akhir: x berisi solusi SPL.
}
const
    epsilon = 0.000001;
var
    i, j : integer;
    konvergen : boolean;
    sigma1, sigma2 : real;
    xlama : vektor;
begin

    repeat
        for i:=1 to n do
            begin
                xlama[i]:=x[i]; {simpan nilai x[i] sebelumnya}
                sigma1:=0;
                for j:=1 to i-1 do
                    sigma1:=sigma1 + a[i,j]*x[j];
                {endfor}
                sigma2:=0;
                for j:=i+1 to n do
                    sigma2:=sigma2 + a[i,j]*x[j];
                {endfor}
                x[i]:= (b[i] - sigma1 - sigma2)/a[i,i]; {a[i,i] <> 0}
            end;
            {periksa kekonvergenan}
            konvergen:=true; i:=1;
            while (konvergen) and (i<=n) do
                begin
                    {bila salah satu dari x[i], i=1, 2, ..., n tidak memenuhi
                    ABS(xlama[i] - x[i]) < epsilon
                    berarti lelaran belum konvergen}
                    if ABS(xlama[i] - x[i]) > epsilon then
                        konvergen:=false; {belum konvergen}
                    end if
                    i:=i+1;
                end;
            { konvergen or i > n }
        until konvergen;
    end;

```

**Contoh 4.20**

[MAT92] Tentukan solusi SPL

$$\begin{aligned}4x - y + z &= 7 \\4x - 8y + z &= -21 \\-2x + y + 5z &= 15\end{aligned}$$

dengan nilai awal  $P_0 = (x_0, y_0, z_0) = (1, 2, 2)$ .  
(Solusi sejatinya adalah  $(2, 4, 3)$ )

**Penyelesaian:**

- (a) Metode lelaran Jacobi  
Persamaan lelarannya:

$$\begin{aligned}x_{r+1} &= \frac{7 + y_r - z_r}{4} \\y_{r+1} &= \frac{21 + 4x_r - z_r}{8} \\z_{r+1} &= \frac{15 + 2x_r - y_r}{5}\end{aligned}$$

Lelarannya:

$$\begin{aligned}x_1 &= \frac{7 + 2 - 2}{4} = 1.75 \\y_1 &= \frac{21 + 4(1) - 2}{8} = 3.375 \\z_1 &= \frac{15 + 2(1) - 2}{5} = 3.000 \\x_2 &= \frac{7 + 3.375 - 3.00}{4} = 1.84375 \\y_2 &= \frac{21 + 4(3.375) - 3.00}{8} = 3.875 \\z_2 &= \frac{15 + 2(1.75) - 3.375}{5} = 3.025 \\&\dots \\x_{19} &= 2.00000000 \\y_{19} &= 4.00000000 \\z_{19} &= 3.00000000\end{aligned}$$



- (b) Metode lelaran Gauss-Seidel  
Persamaan lelarannya,

$$x_{r+1} = \frac{7 + y_r - z_r}{4}$$

$$y_{r+1} = \frac{21 + 4x_r - z_r}{8}$$

$$z_{r+1} = \frac{15 + 2x_r - y_r}{5}$$

Lelarannya,

$$x_1 = \frac{7 + 2 - 2}{4} = 1.75$$

$$y_1 = \frac{21 + 4(1.75) + 2}{8} = 3.75$$

$$z_1 = \frac{15 + 2(1.75) - 3.75}{5} = 3.000$$

$$x_2 = \frac{7 + 3.75 - 2.95}{4} = 1.95$$

$$y_2 = \frac{7 + 3.75 - 2.95}{8} = 3.96875$$

$$z_2 = \frac{15 + 2(1.95) - 3.968375}{5} = 2.98625$$

...

$$x_{10} = 2.00000000$$

$$y_{10} = 4.00000000$$

$$z_{10} = 3.00000000$$

Jadi, solusi SPL adalah  $x = 2.00000000$ ,  $y = 4.00000000$ ,  $z = 3.00000000$  ■

## 4.10 Contoh Soal Terapan

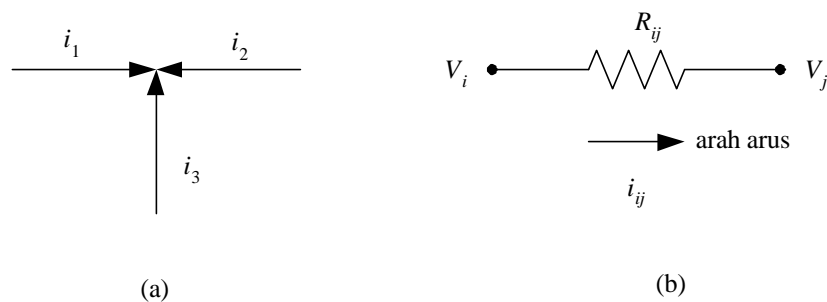
Dalam sebuah rangkaian listrik berlaku hukum-hukum arus Kirchoff menyatakan bahwa jumlah aljabar dari semua arus yang memasuki suatu simpul (Gambar 4.4a) haruslah nol:

$$\sum i = 0 \quad (\text{P.4.38})$$

Dalam hal ini, semua arus  $i$  yang memasuki simpul dianggap bertanda positif. Sedangkan hukum Ohm (Gambar 4.4b) menyatakan bahwa arus  $i$  yang melalui suatu tahanan adalah :

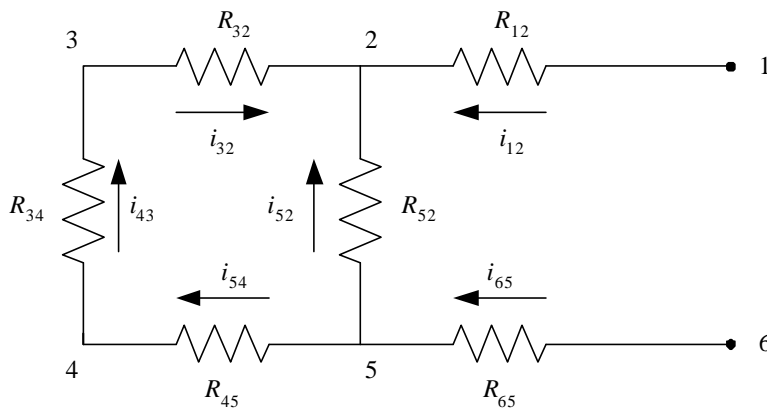
$$i_{ij} = \frac{V_i - V_j}{R_{ij}} \quad (\text{P.4.39})$$

yang dalam hal ini  $V$  adalah tegangan dan  $R$  adalah tahanan.



**Gambar 4.4** (a) Hukum Kirchoff, (b) hukum Ohm

Diberikan sebuah rangkaian listrik dengan 6 buah tahanan seperti pada Gambar 4.5 [CHA91]. Anda diminta menghitung arus pada masing-masing rangkaian.



**Gambar 4.5** Rangkaian listrik dengan 6 buah tahanan

Arah arus dimisalkan seperti diatas. Dengan hukum Kirchoff diperoleh persamaan-persamaan berikut :

$$\begin{array}{rclcl} i_{12} & + i_{52} & + i_{32} & = & 0 \\ i_{65} & - i_{52} & - i_{54} & = & 0 \\ i_{43} & - i_{32} & & = & 0 \\ i_{54} & - i_{43} & & = & 0 \end{array}$$

Dari hukum Ohm didapat :

$$\begin{array}{rclcl} i_{32} R_{32} & - V_3 & + V_2 & = & 0 \\ i_{43} R_{43} & - V_4 & + V_3 & = & 0 \\ i_{65} R_{65} & & + V_5 & = & 0 \\ i_{12} R_{12} & & + V_2 & = & 0 \\ i_{54} R_{54} & - V_5 & + V_4 & = & 0 \\ i_{52} R_{52} & - V_5 & + V_2 & = & 0 \end{array}$$

Dengan menyusun kesepuluh persamaan diatas didapatkan SPL sbb :

$$\begin{array}{cccccccccc} i_{12} & i_{52} & i_{32} & i_{65} & i_{54} & i_{43} & V_2 & V_3 & V_4 & V_5 \\ \left[ \begin{array}{cccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{32} & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_{43} & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & R_{65} & 0 & 0 & 0 & 0 & 0 & 1 \\ R_{12} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{54} & 0 & 0 & 0 & 1 & -1 \\ 0 & R_{52} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{array} \right] \left[ \begin{array}{c} i_{12} \\ i_{52} \\ i_{32} \\ i_{65} \\ i_{54} \\ i_{43} \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \right] & = & \left[ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_6 \\ V_1 \\ 0 \\ 0 \end{array} \right] \end{array}$$

Tentukan

$$i_{12}, \quad i_{52}, \quad i_{32}, \quad i_{65}, \quad i_{54}, \quad i_{43}, \quad V_2, \quad V_3, \quad V_4, \quad V_5$$

bila diketahui

$$\begin{array}{lll} R_{12} = 5 \text{ ohm} , & R_{52} = 10 \text{ ohm} , & R_{32} = 10 \text{ ohm} \\ R_{65} = 20 \text{ ohm} , & R_{54} = 15 \text{ ohm} , & R_{14} = 5 \text{ ohm.} \\ V_1 = 200 \text{ volt} , & V_6 = 0 \text{ volt.} \end{array}$$

### Penyelesaian:

Persoalan ini diselesaikan dengan metode eliminasi Gauss. Matriks awal sebelum proses eliminasi Gauss adalah:

$$\left[ \begin{array}{cccccccccccc|c} 1.000 & 1.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & -1.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 10.000 & 0.000 & 0.000 & 0.000 & -1.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 5.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 20.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 0.000 \\ 5.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 200.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 15.000 & 0.000 & 0.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 \\ 0.000 & 10.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & -1.000 & 0.000 & 0.000 \end{array} \right]$$

Matriks akhir setelah eliminasi adalah:

$$\left[ \begin{array}{cccccccccccc|c} 1.000 & 1.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & -1.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.100 & 0.000 & 0.000 & -0.100 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & -1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.200 & -0.200 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.100 & -0.200 & 0.200 & 0.150 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.600 & 0.600 & 0.350 & 40.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.100 & 0.025 & 20.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.200 & -26.667 & 0.000 \end{array} \right]$$

Dengan teknik penyulihan mundur diperoleh solusinya sebagai berikut:

$$\begin{array}{ll} i_{12} & = 4.444 \text{ ampere,} & i_{52} & = -4.444 \text{ ampere} \\ i_{32} & = 0.000 \text{ ampere,} & i_{65} & = -6.667 \text{ ampere} \\ i_{54} & = -2.222 \text{ ampere,} & i_{43} & = -2.222 \text{ ampere} \\ V_2 & = 177.778 \text{ volt,} & V_3 & = 177.778 \text{ volt} \\ V_4 & = 166.667 \text{ volt,} & V_5 & = 133.333 \text{ volt} \end{array}$$

*Simplex veri sigillum*  
Kesederhanaan adalah tanda kebenaran  
(Peribahasa Latin)

Kebenaran yang paling agung adalah yang paling sederhana.  
Begitu pula orang yang paling agung  
(Campbell)

## Soal Latihan

1. Pecahkan SPL berikut ini:

$$2.51x_1 + 1.48x_2 + 4.53x_3 = 0.05$$

$$1.48x_1 + 0.93x_2 - 1.30x_3 = 1.03$$

$$2.68x_1 + 3.04x_2 - 1.48x_3 = -0.53$$

dengan metode:

- (a) eliminasi Gauss naif ( manual, 3 angka bena)
- (b) eliminasi Gauss yang diperbaiki dengan tataancang *pivoting* (manual, 3 angka bena)
- (c) eliminasi Gauss yang diperbaiki dengan tataancang *pivoting* (komputer, jumlah angka bena sesuai dengan komputer yang digunakan).

Sulihkan jawaban masing-masing (a), (b), dan (c) ke dalam SPL, lalu bandingkan hasilnya dengan ruas kanan (vektor  $b$ )

2. Diberikan sistem persamaan linier  $Ax = b$  dengan  $A$  dan  $b$  sebagai berikut :

$$A = \begin{bmatrix} 1 & 2 & 3 & -1 \\ 2 & 5 & 4 & 8 \\ 4 & 2 & 2 & 1 \\ 6 & 4 & -1 & -2 \end{bmatrix} \quad b = \begin{bmatrix} 10 \\ 8 \\ -2 \\ 4 \end{bmatrix}$$

- (a) Tentukan solusinya dengan metode eliminasi Gauss
- (b) Tentukan determinan matriks  $A$
- (c) Tentukan solusinya dengan metode eliminasi Gauss-Jordan
- (d) Tentukan solusinya dengan metode matriks balikan
- (e) Tentukan solusinya dengan metode dekomposisi LU
- (f) Tentukan solusinya dengan metode lelaran Gauss-Seidell
- (g) Tentukan solusinya dengan metode lelaran Jacobi

Terapkan strategi *pivoting* untuk (a), (b), (c), (d), dan (e).

3. *Pivoting* lengkap jarang diterapkan orang karena kerumitannya. Dari praktek ditemukan bahwa *pivoting* lengkap memberikan hasil yang lebih teliti daripada *pivoting* sebagian meskipun ketelitian ini dibayar dengan waktu komputasi tambahan. Tunjukkan kebenaran pernyataan ini dengan memecahkan SPL berikut :

$$0.002110x + 0.08204y = 0.04313$$

$$0.3370x + 12.84y = 6.757$$

- (a) tanpa *pivoting* (eliminasi Gauss naif)
- (b) dengan *pivoting* sebagian
- (c) dengan *pivoting* lengkap

Semua perhitungan menggunakan empat angka bena (manual).

4. Pecahkan sistem persamaan linier  $Ax = b$  dengan

$$A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

dan  $b$  adalah  $(1,0,0)^T$ ,  $(0,1,0)^T$ , dan  $(0,0,1)^T$ . Metode yang digunakan:

- (a) metode eliminasi Gauss yang diperbaiki (sekali jalan).
- (b) metode eliminasi Gauss-Jordan dengan tataancang *pivoting* (sekali jalan)
- (c) metode matriks balikan
- (d) metode dekomposisi LU

Gunakan komputer dan ketelitian hasil semaksimal mungkin (bilangan berketelitian ganda). Hitung juga determinan matriks  $A$ .

5. Sekumpulan sistem persamaan linier  $Ax = b$  mempunyai matriks  $A$  yang sama tetapi vektor  $b$  berbeda-beda. Matriks  $A$  nya adalah matriks  $A$  yang didefinisikan pada soal nomor 2, sedangkan vektor  $b$  adalah sbb:

$$b_1 = \begin{bmatrix} 1 \\ 4 \\ -2 \\ 0 \end{bmatrix} \quad b_2 = \begin{bmatrix} -2 \\ 5 \\ 1 \\ 3 \end{bmatrix} \quad b_3 = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 10 \end{bmatrix}$$

- (a) selesaikan dengan metode dekomposisi  $LU$
- (b) dengan metode eliminasi Gauss-Jordan, yang dalam hal ini matriks  $A$  digabung (*augmented*) dengan semua vektor  $b$ .

6. Diberikan SPL  $Ax = b$ :

$$A = \begin{bmatrix} 2 & 100 & 3000 \\ 50 & 3 & 4 \\ -1 & 2 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 2000 \\ 10 \\ 1 \end{bmatrix}$$

Tentukan solusinya sampai 4 angka bena dengan :

- (a) metode eliminasi Gauss tanpa penskalaan
- (b) metode eliminasi Gauss dengan penskalaan

Dengan penskalaan, bagilah setiap baris  $i$  dengan  $\max |a_{ij}|, j = 1, 2, 3, \dots, n$ . Periksa solusi anda dengan penyulihan kembali kedalam SPL semula.

7. Pada persoalan  $m$  persamaan dengan  $n$  variabel ( $m < n$ ), tentukan solusi umum dari  $Ax = b$ , yang dalam hal ini:

$$A = \begin{bmatrix} 2 & 3 & 1 & 4 & 1 \\ 2 & 3 & 1 & 1 & -1 \\ 4 & 6 & -1 & 1 & 2 \end{bmatrix} \quad \text{dan} \quad b = \begin{bmatrix} 6 \\ 1 \\ 5 \end{bmatrix}$$

8. Pecahkan sistem persamaan linier berikut dengan metode eliminasi Gauss :

$$\begin{array}{ll} \text{(i)} \quad 6.122x + 1500.5y = 1506.622 & \text{(ii)} \quad 1.001x + 1.5y = 0 \\ \quad \quad 2000x + \quad \quad 3y = 2003 & \quad \quad 2x + \quad 3y = 1 \end{array}$$

- (a) Tanpa *pivoting* (naif);
- (b) dengan *pivoting*.
- (c) Cek jawaban anda dengan menyulihkan solusi kedalam SPL semula.  
Lihat
- (d) bedanya dengan nilai ruas kanan.

Untuk sistem (i) gunakan enam angka bena, dan untuk (ii) gunakan empat angka bena. Ingatlah bahwa setiap komputasi harus dibulatkan ke jumlah angka bena yang diminta (tidak hanya pada hasil akhir saja).

9. Matriks Hilbert adalah contoh klasik matriks yang berkondisi buruk. Misalkan  $A$  adalah matriks Hilbert dan diberikan SPL  $Ax = b$ :

$$\begin{array}{l} x_1 + 1/2 x_2 + 1/3 x_3 + 1/4 x_4 = 1 \\ 1/2 x_1 + 1/3 x_2 + 1/4 x_3 + 1/5 x_4 = 0 \\ 1/3 x_1 + 1/4 x_2 + 1/5 x_3 + 1/6 x_4 = 0 \\ 1/4 x_1 + 1/5 x_2 + 1/6 x_3 + 1/7 x_4 = 0 \end{array}$$

Pecahkan  $Ax = b$  dengan metode eliminasi Gauss naif dengan ketentuan:

- (a) semua bilangan dalam bentuk pecahan, sehingga tidak ada galat akibat pembulatan. Solusinya eksak, misalkan dilambangkan dengan  $x$ . Hitung  $Ax$ , dan bandingkan hasilnya dengan  $b$ .

- (b) semua bilangan dalam tiga angka bena (manual, tanpa komputer). Solusinya hampiran, misalkan dilambangkan dengan  $\hat{x}$ . Hitung  $A\hat{x}$ , dan bandingkan hasilnya dengan  $b$ . Hitung  $e = x - \hat{x}$
- (c) semua bilangan berketelitian tinggi (pakai komputer). Solusinya hampiran, misalkan dilambangkan dengan  $\hat{x}$ . Hitung  $A\hat{x}$ , dan bandingkan hasilnya dengan  $b$ . Hitung  $e = x - \hat{x}$ .
10. (a) Dari soal nomor 4 di atas, tentukan determinan matriks  $A$  untuk masing-masing ketentuan (a), (b), (c). Apa kesimpulan anda?
- (b) Normalkan matriks  $A$ , lalu hitung bilangan kondisi matriks  $A$  (gunakan komputer). Apa kesimpulan anda?

11. Pecahkan sistem persamaan linier

$$\begin{aligned} 8x_1 + x_2 + 3x_3 + 2x_4 &= 0 \\ 2x_1 + 9x_2 - x_3 - 2x_4 &= 1 \\ x_1 + 3x_2 + 2x_3 - x_4 &= 2 \\ x_1 + 6x_3 + 4x_4 &= 3 \end{aligned}$$

dengan metode:

- (a) dekomposisi  $LU$ , yang dalam hal ini  $L$  dan  $U$  dihitung dengan (i) metode  $LU$  Gauss (tidak naif) dan (ii) metode reduksi Crout
- (b) lelaran Jacobi ( $\epsilon = 10^{-10}$ ). Tebakan awal sembarang.
- (c) lelaran Gauss-Seidell ( $\epsilon = 10^{-10}$ ). Tebakan awal sembarang

Gunakan komputer (ketelitian hasil semaksimal mungkin). Untuk (b) dan (c), apakah matriks  $A$  dominan secara diagonal?

12. Dapatkah sistem persamaan linier berikut :

$$\begin{array}{lll} \text{(a)} \quad 5x + 3y = 6 & \text{(b)} \quad 5x + 3y = 6 & \text{(c)} \quad 2x + y - 5z = 9 \\ \quad 4x - 2y = 8 & \quad -6x - 8y = -4 & \quad x - 5y - z = 14 \\ & & \quad 7x - y - 3z = 26 \end{array}$$

diselesaikan dengan metode iterasi Jacobi dan iterasi Gauss-Seidell ?  
Mengapa ?

13. Matriks Hilbert adalah contoh klasik matriks berkondisi buruk. Diberikan matriks Hilbert berukuran  $4 \times 4$  :



$$H = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

Periksa kondisinya dengan :

- (a) Hitung  $HH^{-1}$  apakah berbeda dari matriks identitas
- (b) Hitung  $(H^{-1})^{-1}$  apakah berbeda dari matriks  $H$
- (c) Hitung  $H^{-1} (H^{-1})^{-1}$  apakah berbeda dari matriks identitas  $I$  dan apakah berbeda dari jawaban (a)
- (d) Hitung bilangan kondisinya apakah sangat besar dibandingkan dengan (Normalkan terlebih dahulu matriks  $H$ )

14. Seperti nomor 13, tetapi matriksnya adalah matriks  $A$  pada soal nomor 1.