Kuliah 1 — Angka di Dalam Mesin: Galat, Floating Point, dan Meja Kerja Anda

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 1

Dosen: Irfan Subakti — yifana@gmail.com

Komputer tidak menghitung dengan bilangan real matematika; ia menghitung dengan himpunan terbatas bilangan biner yang hanya menghampirinya. Hari ini kita menyiapkan meja kerja, belajar mengukur galat, membuka 64 bit sebuah double, dan menjumlahkan struk laundry setahun tanpa kehilangan satu rupiah pun. Buku: Bab 1, Subbab 1.1–1.18.

Cakupan Pertemuan Ini

1. **Mengapa metode numerik** — struk yang jumlahnya tidak cocok, dan tiga pertanyaan yang harus dijawab setiap metode.
2. **Meja kerja** — JDK 25, Python 3.14, VS Code, NumPy, dan SciPy di Windows, macOS, dan Linux.
3. **Mengukur galat** — galat sejati, galat relatif, galat hampiran, dan aturan berhenti untuk iterasi.
4. **Cara mesin menyimpan bilangan** — double IEEE 754, epsilon mesin, dan lima percobaan.
5. **Galat pembulatan bertemu galat pemotongan** — deret Taylor, dilema ukuran langkah, dan buku kas laundry yang benar.

Setiap program mata kuliah ini ada dalam Java dan Python, dan kedua versi mencetak keluaran yang persis sama. Setiap keluaran yang ditampilkan hari ini adalah keluaran yang sungguh dicetak program. Buku: Bab 1.

Titik Berangkat: Struk yang Jumlahnya Tidak Cocok

Bu Wati mengelola laundry kecil dekat kampus. Tarifnya Rp7.500 per kilogram; satu kantong 4,1 kg berharga Rp30.750.

Spreadsheet buatan keponakannya menghitung 4,1 × 7500 lalu menanyakan apakah hasilnya sama dengan 30750. Jawabannya FALSE: hasil kali tampil sebagai 30749.999999999996.

Enam puluh pesanan sehari, lebih dari dua puluh ribu struk setahun. Di tablet murah yang menyimpan jumlah uang dalam format lebih pendek, total setahun **kurang Rp826**.

Tidak ada yang salah ketik. Mesin tidak dapat menyimpan 4,1 secara tepat, dan selisih kecil itu bertahan melewati setiap operasi.

Galat kecil yang sama hidup di model cuaca BMKG, di tabel angsuran bank, dan di penghitung jarak aplikasi ojek. Tahun 1991 sebuah baterai Patriot di Dhahran menghitung waktu dalam persepuluhan detik dengan format yang tidak dapat menyimpan 0,1; setelah sekitar 100 jam jamnya meleset 0,34 detik dan intersepsi gagal. Galat per detak sangat kecil. Yang mematikan adalah akumulasinya. Buku: Subbab 1.1.

Apa Itu Metode Numerik

Matematika sekolah menyelesaikan $2x+3=11$ dengan rumus. Masalah nyata jarang punya rumus: suku bunga yang tersembunyi dalam cicilan (Bab 2), kedalaman untuk volume tertentu dalam tangki yang berbaring (Bab 3).

Metode numerik mengganti masalah itu dengan barisan terbatas penjumlahan, pengurangan, perkalian, dan pembagian yang jawabannya sedekat mungkin dengan kebenaran sesuai kebutuhan kita.

Setiap metode dinilai dengan tiga pertanyaan: **Apakah konvergen? Seberapa cepat? Apakah stabil** — apakah galat kecil tetap kecil?

> **WAWASAN KUNCI:** Jawaban metode numerik tidak pernah tepat, tetapi kita tahu *seberapa* tidak tepatnya, dan kita dapat membeli ketelitian lebih dengan kerja lebih. Biseksi yang dibagi dua 20 kali mengetahui akar sampai satu per sejuta.

Ketiga pertanyaan ini adalah pertanyaan buku-buku klasik, Chapra dan Canale serta Burden dan Faires, dan akan kembali di setiap bab mata kuliah ini. Buku: Subbab 1.2.

Meja Kerja

Tabel 1.1 Alat-alat meja kerja dan versi yang dipakai di seluruh buku.

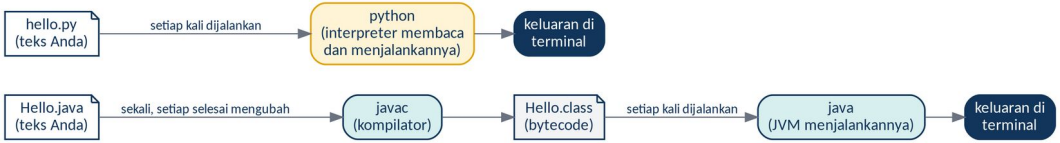
Alat	Peran dalam mata kuliah
JDK 25 (LTS)	`javac` dan `java`: mengompilasi dan menjalankan program Java
Python 3.14	interpreter, di dalam lingkungan virtual `.venv`
NumPy 2.x, SciPy	jawaban pustaka untuk memeriksa program kita sendiri
VS Code + ekstensi Java dan Python	editor, tombol Run, terminal terpadu

Windows — Temurin 25 (LTS) dari adoptium.net atau `winget`, Python install manager (`py install 3.14`), VS Code. **macOS** — berkas `.pkg` JDK, installer dari python.org. **Linux** — `openjdk-25-jdk` dan `python3-venv` dari distribusi.

Periksa di terminal *baru*: `javac -version`, `python3 --version` (di Windows `py --version`), `code --version`.

Tabel 1.1: rilis tambalan yang lebih baru bekerja persis sama. Ubuntu 26.04 LTS membawa OpenJDK 25 dan Python 3.14 secara bawaan, tetapi tidak memasang `javac` maupun modul `venv` kecuali diminta. Buku: Subbab 1.3–1.6.

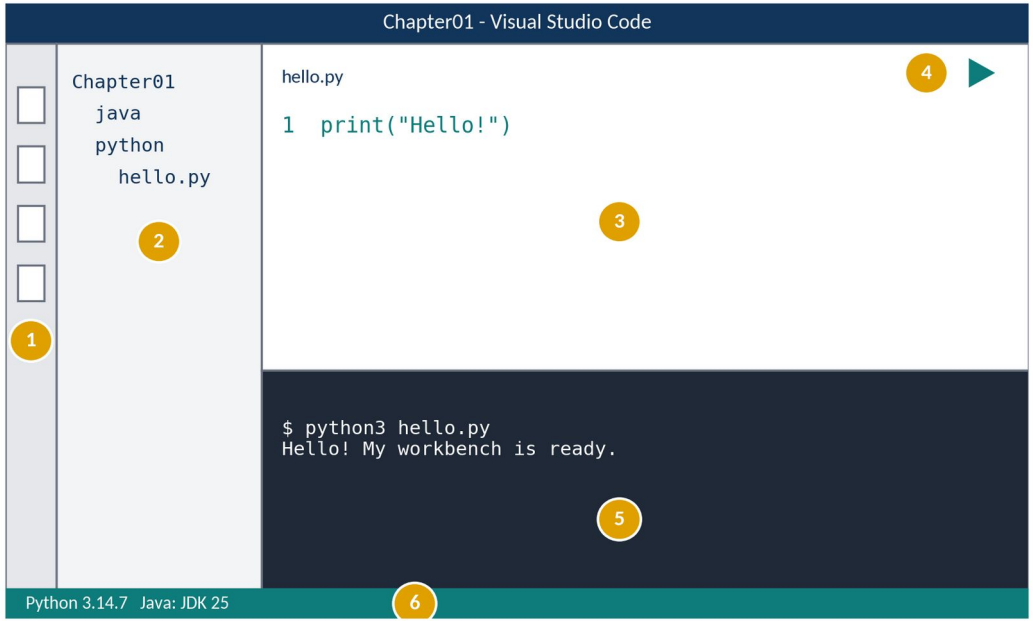
Java Dikompilasi, Python Diinterpretasi



Gambar 1.1 Bagaimana program Java dan program Python dijalankan. Java dikompilasi sekali oleh `javac` menjadi *bytecode* yang dijalankan Java Virtual Machine; Python dibaca dan dijalankan interpreter setiap kali.

Gambar 1.1: `javac Hello.java` menghasilkan `Hello.class`; `java Hello` menjalankannya di JVM. `python3 hello.py` membaca sumbernya setiap kali. Kecepatan bukan topik hari ini; ketepatanlah topiknya. Buku: Subbab 1.3.

VS Code, Lingkungan Virtual, NumPy, dan SciPy



Gambar 1.2 Bagian-bagian Visual Studio Code yang dipakai buku ini: (1) Activity Bar, (2) Explorer, (3) editor, (4) tombol Run, (5) terminal terpadu, (6) Status Bar.

Satu folder per bab; buka *folder*-nya di VS Code. Di terminalnya: `python3 -m venv .venv`, aktifkan, lalu `pip install numpy scipy`.

Gambar 1.2: dua ekstensi Microsoft, Extension Pack for Java dan Python, memberi tombol Run dan pemilih interpreter di Status Bar. `check_setup.py` dan `CheckSetup.java` memastikan Python, lingkungan virtual, NumPy, SciPy, dan Java dalam sekali jalan. Buku: Subbab 1.7–1.8.

Mengukur Galat: Sejati dan Relatif

Jika p nilai sejati dan $\tilde{p}$ hampirannya, **galat sejati** adalah

$$E_t = p - \tilde{p} \tag{1.1}$$

Galat 1 cm sangat besar untuk sekrup dan tidak berarti untuk jalan raya, maka kita menskalakannya menjadi **galat relatif**:

$$\varepsilon_t = \frac{p - \tilde{p}}{p} \times 100\ \% \tag{1.2}$$

Persamaan (1.1) bersatuan; Persamaan (1.2) tidak, dan galat relatif sekitar 0.5×10^{-n} *berarti kira-kira* n *angka signifikan yang benar. Untuk* $\pi \approx 3.1416$ *galat relatifnya sekitar* 2.3×10^{-6}*: lima sampai enam angka benar. Buku: Subbab 1.9.1.*

Galat Hampiran dan Aturan Berhenti

Dalam praktik kebenaran tidak diketahui. Iterasi membandingkan jawaban terbarunya dengan jawaban sebelumnya:

$$\varepsilon_a = \frac{\tilde{p}_{\text{new}} - \tilde{p}_{\text{old}}}{\tilde{p}_{\text{new}}} \times 100\ \% \tag{1.3}$$

Berhenti ketika nilai mutlak ε_a turun di bawah toleransi untuk n angka signifikan (Scarborough):

$$\varepsilon_s = 0.5 \times 10^{2-n}\ \% \tag{1.4}$$

Persamaan (1.3) dan Persamaan (1.4) adalah aturan berhenti setiap metode iteratif di Bab 2, 3, 5, dan 13. Untuk tiga angka toleransinya 0,05%. Buku: Subbab 1.9.2.

Dua Jenis Galat

> **DEFINISI. Galat pembulatan** (*round-off*) berasal dari penyimpanan bilangan dengan digit terbatas. **Galat pemotongan** (*truncation*) berasal dari menghentikan proses matematika tak hingga terlalu dini: deret, limit, integral.

Galat pembulatan adalah ulah mesin; galat pemotongan adalah pilihan metode. Sebagian besar mata kuliah ini membahas cara mengendalikan galat pemotongan tanpa disergap galat pembulatan.

Di sampingnya ada kekeliruan manusia, galat model, dan galat data, yang tidak dapat diperbaiki algoritma apa pun.

Dilema ukuran langkah nanti menunjukkan dua jenis galat ini menarik ke arah berlawanan. Buku: Subbab 1.9.3.

Mengapa 0,1 Tidak Dapat Disimpan

Dalam basis 2, sepersepuluh punya ekspansi berulang, sama seperti sepertiga dalam basis 10:

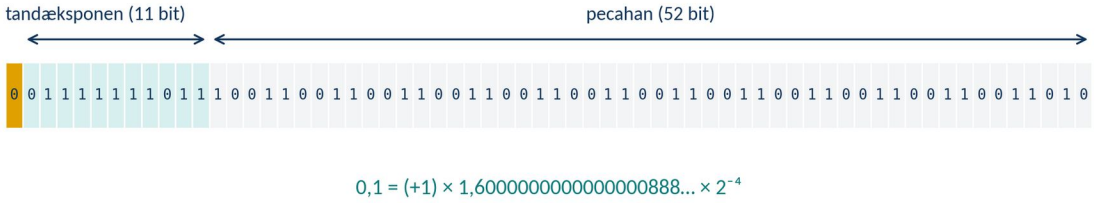
$$0.1 = 0.0\,0011\,0011\,0011\ldots_2 \tag{1.5}$$

Sebuah **double** (IEEE 754 binary64) menyimpan satu bit tanda, eksponen 11 bit e, dan pecahan 52 bit:

$$x = (-1)^s \times (1.f_1 f_2 \ldots f_{52})_2 \times 2^{e-1023} \tag{1.6}$$

Persamaan (1.5): pola 0011 tidak pernah berakhir, sehingga harus dipotong dan dibulatkan. Persamaan (1.6): angka 1 di depan tidak disimpan, sehingga presisinya 53 bit, sekitar 15 sampai 16 digit desimal. Buku: Subbab 1.10.1–1.10.2.

64 Bit dari 0,1



Gambar 1.3 Ke-64 bit double yang terdekat ke 0,1: tanda 0, eksponen 01111111011 (1019, berarti 2^{-4}), dan 52 bit pecahan tempat pola 1001 berulang sampai dipotong dan dibulatkan.

Gambar 1.3: bit-bit terakhir dibulatkan ke atas, sehingga nilai tersimpan sedikit lebih besar daripada 0,1. Dua format mata kuliah ini adalah presisi tunggal (float, 32 bit, sekitar 7 digit) dan presisi ganda (64 bit, sekitar 16 digit). Buku: Subbab 1.10.2.

Epsilon Mesin dan Batas Pembulatan

Epsilon mesin adalah jarak antara 1 dan double berikutnya:

$$\varepsilon_{\text{mach}} = 2^{-52} \approx 2.22 \times 10^{-16} \tag{1.7}$$

Setiap bilangan tersimpan, dan setiap hasil $+, -, \times, \div$, adalah nilai sejati dikali $(1+\delta)$ dengan galat relatif δ paling besar **satuan pembulatan** $u = 2^{-53} \approx 1.11 \times 10^{-16}$.

> **WAWASAN KUNCI:** Satu operasi hampir sempurna. Bahayanya ada pada *banyak* operasi, yang galatnya menumpuk, dan pada *pengurangan* dua bilangan yang hampir sama, yang menyingkap galat yang sudah ada.

Persamaan (1.7): dalam presisi tunggal epsilon bernilai $2^{-23} \approx 1.19 \times 10^{-7}$. Model "setiap operasi tepat, lalu dibulatkan" adalah dasar semua analisis galat di mata kuliah ini. Buku: Subbab 1.10.3.

Lima Percobaan dengan FloatLab

Kode 1.1 Dari `float_lab.py`: menjumlahkan 0,1 dan 0,2, nilai tepat yang tersimpan untuk 0,1, dan epsilon mesin yang ditemukan dengan membagi dua.

```
    print("1. Adding 0.1 and 0.2")
    s = 0.1 + 0.2
    print(f"   0.1 + 0.2        = {s!r}")
    print(f"   0.1 + 0.2 == 0.3 : {s == 0.3}")

    print("2. What is really stored for 0.1")
    print(f"   {Decimal(0.1)}")        # the exact value of the double

    print("3. Machine epsilon, found by halving")
    eps = 1.0
    while 1.0 + eps / 2 > 1.0:     # stop when half of eps is no longer seen
        eps /= 2
```

```
1. Adding 0.1 and 0.2
   0.1 + 0.2        = 0.30000000000000004
   0.1 + 0.2 == 0.3 : False
2. What is really stored for 0.1
   0.1000000000000000055511151231257827021181583404541015625
3. Machine epsilon, found by halving
   eps              = 2.220446e-16
   1 + eps/2 == 1   : True
4. Adding 0.1 again and again
        10 times: 0.9999999999999999     error 1.110e-16
      1000 times: 99.9999999999986       error 1.407e-12
   1000000 times: 100000.00000133288     error 1.333e-06
```

Kode 1.1, dan kembarannya `FloatLab.java`, mencetak persis baris-baris ini. Jangan pernah membandingkan double hasil hitungan dengan `==`; pakailah toleransi. Percobaan 4 adalah jam Patriot dalam skala kecil: setiap penjumlahan membulatkan dan galatnya tidak saling menghapus. Buku: Subbab 1.11.

Pembatalan Katastrofik

$x^2 - 10^8\,x + 1 = 0$ punya akar dekat 10^{-8}. Rumus sekolah mengurangkan dua bilangan yang sama pada 16 digit pertamanya:

$$x_2 = \frac{-b - \sqrt{b^2 - 4\,a\,c}}{2\,a} \tag{1.8}$$

Hasil kali kedua akar adalah c/a (Vieta), jadi hitung akar besar dengan aman, lalu

$$x_2 = \frac{c}{a\,x_1} \tag{1.9}$$

```
5. Subtracting nearly equal numbers: x^2 - 1e8 x + 1 = 0
   textbook formula : 7.4505805969e-09
   stable formula   : 1.0000000000e-08
   relative error of the textbook formula: 2.549e-01
```

Persamaan (1.8) dan Persamaan (1.9) sama secara aljabar dan sangat berbeda di komputer: galat 25% lawan ketelitian penuh. Pengurangannya sendiri tepat; ia hanya menyingkap sampah yang ditaruh pembulatan di digit-digit terakhir. Itulah arti kestabilan numerik. Buku: Subbab 1.11.

Galat Pemotongan: Deret Taylor

$$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n+R_n \tag{1.10}$$

Untuk e^x di sekitar 0 setiap turunan bernilai 1, dan setiap suku baru adalah suku sebelumnya dikali x/k:

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots \tag{1.11}$$

```
e^0.5 to 3 significant digits (tolerance 0.05 %):
terms           sum   true err %  approx err %
    1    1.000000000    39.346934
    2    1.500000000     9.020401     33.333333
    3    1.625000000     1.438768      7.692308
    4    1.645833333     0.175162      1.265823
    5    1.648437500     0.017212      0.157978
```

Persamaan (1.10) adalah alat di balik hampir setiap rumus galat mata kuliah ini; sisa R_n adalah galat pemotongan. `exp_series.py` *menjumlahkan Persamaan (1.11) dengan aturan berhenti Persamaan (1.4); galat hampiran selalu di atas galat sejati, sehingga aturan itu aman. Pada $x=-20$ suku-suku yang berganti tanda saling membatalkan dengan buruk, dan $1/e^{20}$ adalah jalan yang stabil. Buku: Subbab 1.12.1.*

Dilema Ukuran Langkah

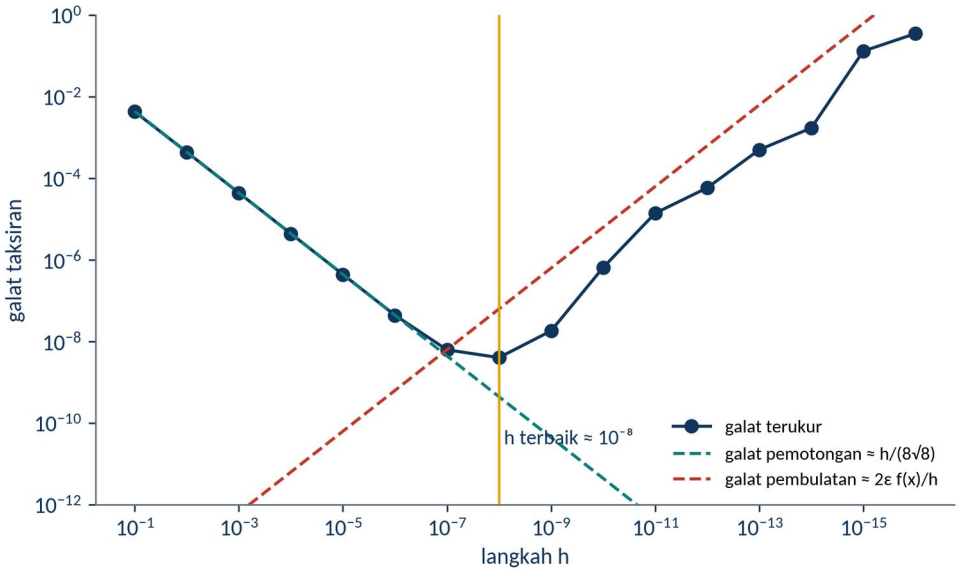
Selisih maju, yang diturunkan dari deret Taylor, punya galat pemotongan yang sebanding dengan h:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(\xi)}{2} h \tag{1.12}$$

h	estimate	error
1e-01	0.349241122458488	4.312e-03
1e-02	0.353112550268730	4.408e-04
1e-03	0.353509207464464	4.418e-05
1e-04	0.353548971285811	4.419e-06
1e-05	0.353552948650915	4.419e-07
1e-06	0.353553346377211	4.422e-08

Persamaan (1.12) untuk $\sqrt{x}$ di $x = 2$: setiap langkah sepuluh kali lebih kecil membuat galat sepuluh kali lebih kecil, sesuai janji, tetapi hanya sampai sekitar $h = 10^{-8}$. Di bawah itu galat membesar lagi. Buku: Subbab 1.12.2.

Tempat Kedua Galat Seimbang



Gambar 1.4 Galat selisih maju untuk $\sqrt{x}$ di $x=2$ terhadap langkah h: galat pemotongan turun bersama h, galat pembulatan tumbuh seperti $1/h$, dan totalnya terkecil di dekat $h=10^{-8}$.

$$E(h) \approx \frac{M}{2} h + \frac{2\varepsilon_{\text{mach}} |f(x)|}{h} \tag{1.13}$$

$$h_{\text{opt}} = 2\sqrt{\frac{\varepsilon_{\text{mach}} |f(x)|}{M}} \tag{1.14}$$

Gambar 1.4: bentuk V dari Persamaan (1.13). Meminimumkannya memberi Persamaan (1.14), sekitar akar kuadrat epsilon mesin, 10^{-8}. Lebih kecil tidak selalu lebih baik; Bab 9 kembali ke dilema ini dengan rumus yang lebih baik. Buku: Subbab 1.12.2.

Buku Kas Laundry: Menjumlahkan Uang dengan Aman

365 hari dengan 60 pesanan, berat 1,0 sampai 8,0 kg, struk pseudo-acak yang sama di Java dan Python, dijumlahkan dengan empat cara:

```
Orders in the year     : 21,900
Exact total (integer)  : Rp 739,304,250
double, simple sum     : Rp 739,304,250.00  (off by +0.00)
float, simple sum      : Rp 739,303,424.00  (off by -826.00)
float, Kahan sum       : Rp 739,304,256.00  (off by +6.00)
```

Penjumlahan terkompensasi Kahan menyimpan bagian kecil yang hilang dari setiap penjumlahan di dalam koreksi c:

$$c = ((t + y) - t) - y \tag{1.15}$$

> **PERINGATAN:** Hitung uang dalam satuan utuh (rupiah, sen, satuan koin terkecil) dengan bilangan bulat. Pakai double untuk hasil ukur, dan penjumlahan terkompensasi atau berpasangan untuk jumlah yang panjang.

Inilah Rp826 milik Bu Wati. Di dekat Rp739 juta jarak antara dua float bertetangga adalah Rp64, sehingga setiap satu dari 21.900 penjumlahan membulatkan sampai Rp32. Persamaan (1.15) menyelamatkan sebagian besarnya bahkan dalam presisi tunggal. Buku: Subbab 1.13.

Membandingkan dengan Pustaka

```
float64 eps (np.finfo)  : 2.220446e-16
float32 eps (np.finfo)  : 1.192093e-07
math.ulp(1.0)           : 2.220446e-16
next double after 1.0   : 1.0000000000000002
0.1 stored as float32   : 0.100000001490116119384765625
loop sum of 10^6 x 0.1  : 100000.00000133288
np.sum (pairwise)       : 100000.00000000003
```

NumPy 2.x membenarkan epsilon mesin yang kita temukan lewat percobaan, dan `np.sum` yang berpasangan jauh lebih teliti daripada perulangan biasa. Java memberi fakta yang sama lewat `Math.ulp` dan `Math.nextUp`, dan `BigDecimal` menjumlahkan secara tepat. Buku: Subbab 1.14.

Metode Numerik dalam Praktik: Profitina

Profitina (app.profitina.com) adalah platform visibilitas AI yang mengukur seberapa siap situs web sebuah merek untuk dikutip mesin jawab AI.

Uang dalam satuan utuh: langganan Pro Rp50.000 untuk tiga puluh hari; koin PROFIT punya 6 desimal, sehingga setiap saldo adalah bilangan bulat dalam sepersejuta koin — aturan buku kas dalam skala besar.

Skor yang dijumlahkan: menggabungkan banyak sinyal adalah jumlah panjang dari hasil kali; menormalkan dengan rentang yang sangat kecil memperbesar galat masukan, efek yang sama dengan suku $1/h$.

Konversi antara PROFIT dan rupiah adalah satu-satunya tempat sebuah kurs bertemu jumlah uang; hasilnya dibulatkan sekali, di titik yang jelas, dengan aturan yang dinyatakan. Buku: Subbab 1.15.

Kesalahan Umum

Membandingkan double dengan `==`. Bandingkan dengan toleransi: relatif untuk bilangan besar, mutlak di dekat nol.

Uang dalam floating point. Pakai satuan bilangan bulat; bulatkan sekali, dengan aturan yang dinyatakan.

Mengurangkan bilangan yang hampir sama. Susun ulang rumusnya, seperti bentuk Vieta.

Membuat langkah sekecil mungkin. Di bawah langkah optimal, galat pembulatan menang.

Menjumlahkan daftar panjang dalam presisi tunggal. Pakai double, penjumlahan Kahan, atau berpasangan.

Di spreadsheet aturannya sama: simpan rupiah sebagai bilangan bulat, dan bulatkan hanya di akhir dengan fungsi ROUND. Buku: Subbab 1.16–1.17.

Ringkasan

1. Metode numerik menjawab dengan galat yang diketahui dan dapat diterima; metode dinilai dari konvergensi, kecepatan, dan kestabilan.

2. Galat diukur sebagai galat sejati, relatif, dan hampiran; iterasi berhenti pada ε_s (Persamaan (1.4)).

3. Double punya presisi 53 bit; setiap operasi meleset paling banyak $u = 2^{-53}$, tetapi galat menumpuk dan pembatalan menyingkapnya.

4. Galat pemotongan turun ketika langkah mengecil, galat pembulatan naik; langkah terbaik menyeimbangkan keduanya (Persamaan (1.14)).

5. Uang dihitung dengan bilangan bulat; jumlah panjang memakai double dan penjumlahan terkompensasi. Program kita sesuai dengan NumPy dan pustaka Java.

Semua yang menyusul di mata kuliah ini, dari biseksi sampai gradient descent, dinilai dengan gagasan-gagasan ini. Buku: Subbab 1.18.

Pertemuan Berikutnya

Bab 2 — Mencari Akar I: Biseksi dan Regula Falsi. Berapa suku bunga bulanan yang tersembunyi dalam sebuah cicilan?

Metode pengurung yang tidak pernah gagal: bagi dua selangnya, atau potong di tempat garis lurus memotong nol.

Sebelumnya: pasang dan periksa meja kerja, jalankan FloatLab dan buku kas di kedua bahasa, dan kerjakan soal latihan tentang galat relatif dan jumlah presisi tunggal.

Jumlah harmonik dalam presisi tunggal, maju dan mundur, sekali lagi menunjukkan bahwa urutan penjumlahan penting; penyelesaiannya ada di buku. Buku: Subbab 1.19 dan 1.20.

Kuliah 2 — Mencari Akar I: Biseksi dan Regula Falsi

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 2

Dosen: Irfan Subakti — yifana@gmail.com

Banyak pertanyaan sehari-hari berakhir pada persamaan yang tidak dapat diselesaikan rumus mana pun. Hari ini kita membangun dua cara tertua dan teraman untuk mencari akar: mulai dari selang tempat fungsi berganti tanda, lalu persempit selang itu sampai akarnya terkunci. Buku: Bab 2, Subbab 2.1–2.13.

Cakupan Pertemuan Ini

1. Pertanyaannya — suku bunga bulanan sebenarnya yang tersembunyi dalam cicilan ponsel.

2. Dari pertanyaan ke akar — rumus anuitas dan fungsi pinjaman $f(r)$.

3. Biseksi — algoritmanya, batas galatnya, dan banyaknya langkah yang diketahui sebelumnya.

4. Regula falsi dan Illinois — memotong di tempat tali busur memotong nol, dan memperbaiki langkah yang merayap.

5. Kode, uji, dan SciPy — satu modul dalam Java dan Python, uji unit, dan tiga jawaban pustaka.

Fungsi-fungsi pengurung yang sama dipakai ulang di bab-bab berikutnya. Semua tabel iterasi Java dan Python sama sampai digit terakhir. Buku: Bab 2.

Mas Dimas dan Ponsel “1,67% Sebulan”

Mas Dimas pengemudi ojek daring di Yogyakarta. Ponselnya adalah kantornya, dan layarnya retak untuk ketiga kalinya.

Sebuah konter menawarkan ponsel Rp6.000.000 dengan **dua belas angsuran Rp600.000**, “bunganya cuma 1,67% sebulan”.

Angka 1,67% itu bunga **flat**: bunga Rp1.200.000 dibagi dua belas bulan dan dibagi harga penuh, seolah seluruh utang tetap terutang sepanjang tahun.

Padahal utangnya turun setiap bulan. Berapa suku bunga bulanan **efektif**, yang akan disebut bank untuk pembayaran yang sama?

Pertanyaan “masukan mana yang memberi keluaran ini?” adalah bentuk setiap masalah hari ini: suku bunga yang membuat angsuran ini adil, kedalaman air saat tangki berisi 3.000 liter, bulan ketika harga yang turun mencapai tingkat tertentu. Buku: Subbab 2.1.

Uang Hari Ini dan Uang Nanti

Pada suku bunga bulanan r, angsuran A yang dibayar k bulan lagi bernilai $A/(1+r)^k$ hari ini. Rencana yang adil membuat nilai-nilai kini itu berjumlah sama dengan harga:

$$P = \frac{A}{1+r} + \frac{A}{(1+r)^2} + \cdots + \frac{A}{(1+r)^n} \tag{2.1}$$

Deret geometri itu berjumlah **rumus anuitas**:

$$P = A\,\frac{1-(1+r)^{-n}}{r} \tag{2.2}$$

Persamaan (2.1) punya suku pertama $A/(1+r)$ dan rasio $1/(1+r)$, yang menghasilkan Persamaan (2.2). Bank memakainya ke dua arah: jika r diketahui, rumus ini memberi angsuran. Dimas mengetahui P, A, dan n dan mencari r, yang duduk di pangkat sekaligus di penyebut; tidak ada langkah aljabar yang dapat memisahkannya. Buku: Subbab 2.2.1.

Mengubah Pertanyaan Menjadi Akar

Pindahkan semuanya ke satu ruas dan tanyakan di mana hasilnya nol:

$$f(r) = A\,\frac{1-(1+r)^{-n}}{r} - P \tag{2.3}$$

Tabel 2.1 Fungsi pinjaman Persamaan (2.3) pada lima suku bunga bulanan: tandanya berganti antara 2% dan 3%.

Suku bunga r	Faktor anuitas	$f(r)$ (rupiah)
1%	11,2551	+753.046
2%	10,5753	+345.205
3%	9,9540	−27.598
4%	9,3851	−368.956
5%	8,8633	−682.049

Tabel 2.1: beberapa tebakan sudah banyak bercerita. Akarnya terletak antara 2% dan 3% — jauh di atas 1,67% yang diiklankan. Mentabulasi atau memplot f sebelum menjalankan metode apa pun adalah kebiasaan pertama bab ini. Buku: Subbab 2.2.2.

Mengapa Pergantian Tanda Menjamin Adanya Akar

DEFINISI. Kurung (*bracket*) adalah selang $[a, b]$ tempat f kontinu dan $f(a)f(b)<0$. Menurut Teorema Nilai Antara, f punya paling sedikit satu akar di dalamnya.

Kurva kontinu yang berawal di atas sumbu dan berakhir di bawahnya pasti memotong sumbu di suatu tempat di antaranya.

Kedua syarat itu penting: tanpa kekontinuan, $1/x$ berganti tanda pada $[-1, 1]$ tetapi sama sekali tidak punya akar.

WAWASAN KUNCI: Metode pengurung mengubah pertanyaan tanpa rumus menjadi barisan pertanyaan ya-atau-tidak: apakah akarnya di kiri atau di kanan titik ini? Tanda $f(c)$ hampir selalu dihitung dengan benar, bahkan ketika nilainya tidak teliti. Itulah sebabnya metode ini sangat tangguh.

Jika akal sehat tidak memberi kurung, melangkahlah dari sebuah titik awal sampai dua nilai bertetangga berbeda tanda, atau, lebih baik, plot fungsinya lebih dulu. Buku: Subbab 2.2.2 dan 2.3.4.

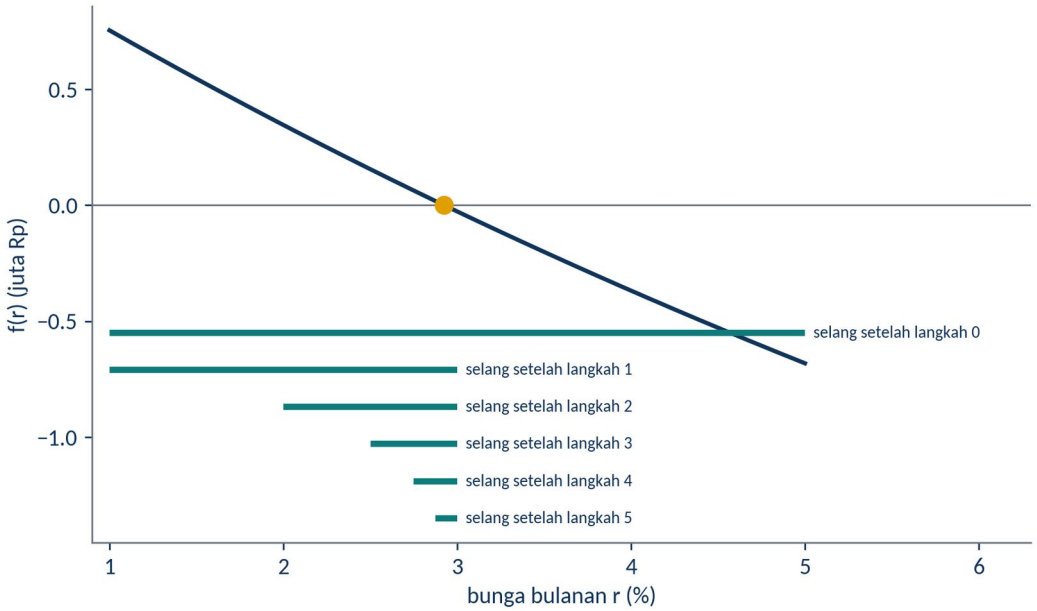
Algoritma Biseksi

Pseudokode 2.1 Metode biseksi dengan toleransi pada perubahan titik tengah.

```
BISECTION(f, a, b, tol, maxIter)
 1  fa = f(a);  fb = f(b)
 2  if fa * fb > 0: error "no sign change: [a, b] is not a bracket"
 3  c_old = a
 4  for k = 1 to maxIter
 5      c = a + (b - a) / 2              // the midpoint
 6      fc = f(c)
 7      if fc == 0 or |c - c_old| < tol: return c
 8      if fa * fc < 0: b = c; fb = fc   // root in [a, c]
 9      else:           a = c; fa = fc   // root in [c, b]
10      c_old = c
11  return c                             // tolerance not reached
```

Pseudokode 2.1: dengan tangan pada $[0.01, 0.05]$*, langkah 1 memberi* $c = 0.03$ *dengan* $f(c) = -27.598$*, jadi akarnya di* $[0.01, 0.03]$*; langkah 2 memberi* $c = 0.02$ *dengan* $f(c) = +345.205$*, jadi akarnya di* $[0.02, 0.03]$*. Hanya tanda* $f(c)$ *yang menentukan. Perhatikan juga baris 5: titik tengah dihitung sebagai* $a + (b - a)/2$*, bukan* $(a + b)/2$*, karena bentuk pertama tidak mungkin keluar dari kurung akibat pembulatan. Baris 2 menolak selang tanpa pergantian tanda alih-alih mengembalikan jawaban yang tidak bermakna. Buku: Subbab 2.3.1–2.3.2.*

Biseksi pada Fungsi Pinjaman



Gambar 2.1 Fungsi pinjaman $f(r)$ dari Persamaan (2.3) dan kurung setelah setiap langkah biseksi, mulai dari $[1\ \%, 5\ \%]$: setiap langkah menyimpan separuh yang memuat pergantian tanda (titik jingga).

Gambar 2.1: setiap batang selebar separuh batang di atasnya. Kurvanya mulus dan turun terus, sehingga akarnya tunggal di dalam kurung. Buku: Subbab 2.3.

Banyaknya Langkah Diketahui Sebelumnya

Titik tengah ke-k terletak di tengah kurung selebar $(b_0 - a_0)/2^{k-1}$:

$$\left|c_k - r^*\right| \le \frac{b_0 - a_0}{2^k} \tag{2.4}$$

Agar galat di bawah toleransi τ, cukup

$$k \ge \log_2 \frac{b_0 - a_0}{\tau} \tag{2.5}$$

Untuk Dimas: $b_0 - a_0 = 0.04$, $\tau = 10^{-7}$, jadi $k \ge \log_2 400.000 = 18,6$: **19 langkah**, bahkan sebelum program dijalankan.

*Persamaan (2.4): satu digit biner per langkah, sekitar 0,3 digit desimal, apa pun bentuk fungsinya. Persamaan (2.5): setiap tambahan satu digit desimal berharga sedikit lebih dari tiga langkah. Inilah **konvergensi linear** dengan laju setengah. Buku: Subbab 2.3.3.*

Regula Falsi: Potong di Tempat Tali Busur Memotong Nol

Biseksi mengabaikan *nilai* f. Jika $f(a)$ besar dan $f(b)$ kecil, akarnya mungkin dekat b.

Tarik garis lurus melalui $(a, f(a))$ dan $(b, f(b))$, lalu potong kurung di tempat garis itu memotong nol:

$$c = b - f(b)\frac{a-b}{f(a)-f(b)} \tag{2.6}$$

Untuk Dimas pada $[0.01, 0.05]$ potongan pertama jatuh di $c = 0.030989$, sudah dekat akar, sedangkan potongan pertama biseksi di 0,03.

Persamaan (2.6) mempertahankan kurung, sehingga regula falsi mewarisi jaminan biseksi bahwa akarnya tidak dapat lolos. Namanya bahasa Latin untuk "aturan posisi palsu": garis itu adalah fungsi palsu yang mewakili fungsi sebenarnya. Buku: Subbab 2.4.1.

Ketika Satu Ujung Macet

Pada fungsi yang melengkung, satu ujung kurung mungkin tidak pernah bergerak. Galatnya lalu menyusut secara linear dengan laju

$$\lambda = 1 - \frac{f'(r^*)(a - r^*)}{f(a)} \tag{2.7}$$

Perbaikan **Illinois**: jika ujung yang sama bertahan dua kali, bagi dua nilai fungsi yang tersimpan di ujung itu, sehingga tali busur berikutnya condong ke arahnya:

$$f(a) \leftarrow \frac{1}{2} f(a) \tag{2.8}$$

Persamaan (2.7) dapat mendekati 1, dan saat itu regula falsi merayap. Persamaan (2.8) menarik potongan berikutnya melewati akar, ujung yang macet bergerak, dan orde konvergensi naik menjadi sekitar 1,44. Di buku satu algoritma melayani keduanya, dengan modifikasi yang dinyalakan lewat sebuah penanda. Buku: Subbab 2.4.2–2.4.3.

Satu Modul, Dua Bahasa

Kode 2.1 Dari `roots.py`: biseksi, tanpa bagian yang mencetak jejak iterasi.

```python
def bisection(f, a, b, tol=1e-7, max_iter=100, trace=False):
    """Halve [a, b] until two midpoints differ by less than tol.
    Returns (root, number of iterations)."""
    fa, fb = f(a), f(b)
    _check(fa, fb)
    c_old = a
    for k in range(1, max_iter + 1):
        c = a + (b - a) / 2                 # safer than (a + b) / 2
        fc = f(c)
        ...
        if fc == 0 or abs(c - c_old) < tol:
            return c, k
        if fa * fc < 0:                     # the root is in [a, c]
            b, fb = c, fc
        else:                               # the root is in [c, b]
            a, fa = c, fc
        c_old = c
    return c, max_iter
```

Kode 2.1: fungsi f adalah parameter. Di Python fungsi adalah nilai biasa; di Java parameternya bertipe `DoubleUnaryOperator`, dan kembarannya dalam Java mengembalikan record `Result` berisi akar dan banyaknya iterasi. `_check` menolak selang yang bukan kurung. Buku: Subbab 2.5.

Menjalankannya: Biseksi

```
Price Rp 6,000,000, 12 instalments of Rp 600,000
Flat rate the seller quotes: 1.6667 % per month

Bisection on [0.01, 0.05]
  k          a          b          c          f(c)  |c - c_old|
  1  0.01000000  0.05000000  0.03000000  -27597.6039   2.000e-02
  2  0.01000000  0.03000000  0.02000000  345204.7326   1.000e-02
  3  0.02000000  0.03000000  0.02500000  154658.7589   5.000e-03
  4  0.02500000  0.03000000  0.02750000   62522.1949   2.500e-03
  5  0.02750000  0.03000000  0.02875000   17213.5840   1.250e-03
  6  0.02875000  0.03000000  0.02937500   -5253.7705   6.250e-04
```

Baris-baris pertama keluaran mengulang hitungan tangan kita. Tabelnya berlanjut sampai tepat 19 langkah seperti diramalkan Persamaan (2.5), dan kolom terakhir terbagi dua di setiap baris. Pada langkah 12 titik tengahnya kebetulan berjarak kurang dari satu rupiah dari akar, tetapi biseksi tidak menyadarinya; ia berhenti berdasarkan batas. Buku: Subbab 2.6.

Menjalankannya: Regula Falsi dan Illinois

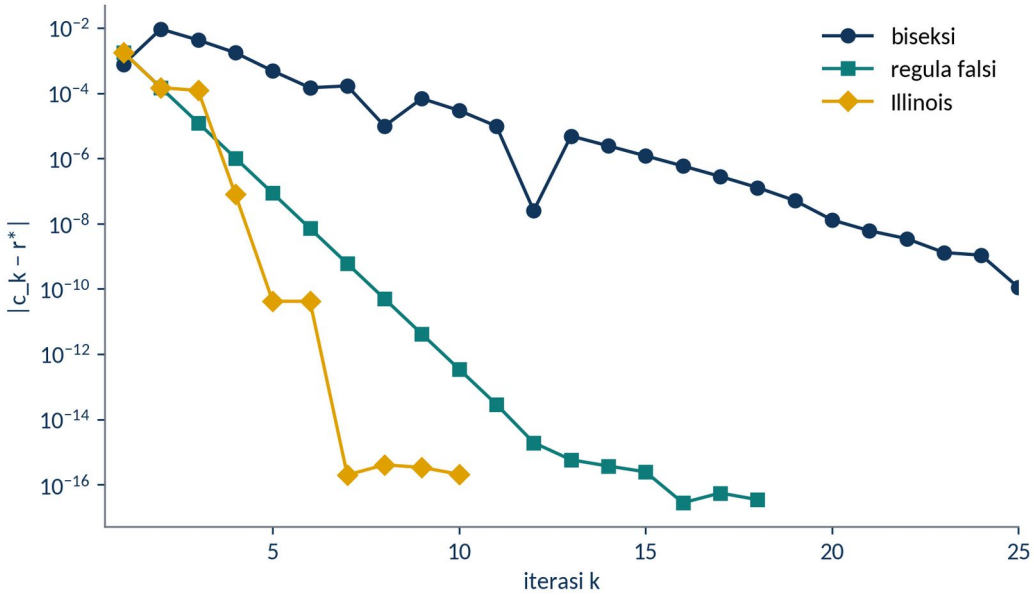
```
Regula falsi on [0.01, 0.05]
  k           a           b           c           f(c)  |c - c_old|
  1  0.01000000  0.05000000  0.03098945     -62719.8812    2.099e-02
  2  0.01000000  0.03098945  0.02937568      -5278.1637    1.614e-03
  3  0.01000000  0.02937568  0.02924082       -440.7396    1.349e-04
  4  0.01000000  0.02924082  0.02922957        -36.7789    1.125e-05
  5  0.01000000  0.02922957  0.02922863         -3.0690    9.391e-07
  6  0.01000000  0.02922863  0.02922855         -0.2561    7.836e-08

Illinois on [0.01, 0.05]
  k           a           b           c           f(c)  |c - c_old|
  1  0.01000000  0.05000000  0.03098945     -62719.8812    2.099e-02
  2  0.01000000  0.03098945  0.02937568      -5278.1637    1.614e-03
  3  0.01000000  0.02937568  0.02910782       4335.4506    2.679e-04
  4  0.02910782  0.02937568  0.02922862         -2.8128    1.208e-04
  5  0.02910782  0.02922862  0.02922854         -0.0015    7.832e-08

True monthly rate : 2.9229 %
Per year, compound: 41.30 %
```

Regula falsi butuh 6 langkah, tetapi lihat kolom a: nilainya tetap 0,01 selamanya — ujung yang macet. Illinois membagi dua $f(a)$ pada langkah 3, potongannya melompat melewati akar, a akhirnya bergerak, dan metode selesai dalam 5 langkah. Bunga efektifnya 2,9229% sebulan, 41,30% setahun, dibanding 1,67% yang diiklankan. Buku: Subbab 2.6.

Tiga Kecepatan Konvergensi



Gambar 2.2 Galat $\left|c_k - r^*\right|$ ketiga metode pengurung pada fungsi pinjaman, dilanjutkan melewati toleransi sampai galatnya mencapai batas presisi ganda.

Gambar 2.2: pada skala logaritmik metode yang konvergen linear turun mengikuti garis lurus; galat biseksi naik-turun di sekitar garisnya yang berlaju setengah, sedangkan Illinois turun makin lama makin cepat, ciri konvergensi superlinear. Buku: Subbab 2.6.

Utang Bulan demi Bulan

Tabel 2.2 Utang Dimas bulan demi bulan pada bunga efektif: bunganya turun ketika utangnya turun, dan angsuran kedua belas melunasi utang dengan tepat.

Bulan	Utang awal	Bunga (2,9229%)	Pokok dibayar	Utang akhir
1	6.000.000	175.371	424.629	5.575.371
2	5.575.371	162.960	437.040	5.138.331
6	3.749.062	109.580	490.420	3.258.642
12	582.961	17.039	582.961	0

Tabel 2.2 adalah pemeriksaan yang berdiri sendiri: pada akar Persamaan (2.3) angsuran kedua belas melunasi utang dengan tepat. Bunga setahunnya tetap Rp1.200.000; hanya cara menyebutnya yang berbeda. Pada bulan 1 Dimas membayar bunga Rp175.371, bukan Rp100.000 — bunga flat meratakan bunga di atas utang yang tidak rata. Buku: Subbab 2.6.

Tempat Regula Falsi Merayap

Kasus terburuk: $f(x) = x^{10} - 1$ pada $[0, 1.3]$, akarnya 1. Kurvanya landai di kiri dan sangat curam di kanan.

```
bisection    : root 1.000000018 after  24 iterations
regula falsi : root 0.999999702 after  65 iterations
Illinois     : root 1.000000000 after  14 iterations
```

> **PERINGATAN:** Regula falsi biasa dapat berhenti karena perubahan c kecil padahal masih jauh dari akar: barisan yang merayap bergerak sedikit per langkah. Jawabannya di sini salah di digit ketujuh meskipun toleransi meminta lebih.

`stagnation.py` dan `Stagnation.java` mencetak tiga baris yang sama. Di sini regula falsi lebih lambat daripada biseksi, dan uji berhentinya menyesatkan; Illinois paling cepat dan benar. Buku: Subbab 2.7.

Mengujinya

```
ok      bisection finds sqrt 2
ok      bisection needs 21 iterations for tol 1e-6
ok      regula falsi and Illinois find sqrt 2
ok      Illinois is faster
ok      no bracket is refused
all tests passed
```

Uji memakai x^2-2, yang akarnya $\sqrt{2}$ kita ketahui. Untuk kurung $[0,2]$ dan $\tau=10^{-6}$, Persamaan (2.5) memberi $\log_2(2\times10^6)=20,9$, jadi tepat 21 iterasi — uji yang memeriksa teori, bukan hanya kode. Uji Java mencetak enam baris yang sama. Buku: Subbab 2.8.

Membandingkan dengan SciPy

```
scipy bisect : 2.92285919 %
scipy brentq : 2.92285408 %  (7 iterations)
toms748      : 2.92285408 %  (3 iterations)
```

`scipy.optimize` *sepakat dengan 2,9229% kita.* `brentq` *menggabungkan biseksi dengan interpolasi dan mempertahankan jaminan kurung; TOMS 748 memakai interpolasi berorde lebih tinggi dan hanya butuh tiga iterasi. Pustaka adalah alat profesional; kode kita memperlihatkan apa yang dikerjakannya di dalam. Buku: Subbab 2.9.*

Metode Numerik dalam Praktik: Profitina

Koin utilitas **PROFIT** milik Profitina dirancang deflasioner: koin baru dikeluarkan menurut jadwal yang *meluruh*, dan koin yang dipakai di platform dibakar.

Kapan emisi kumulatif mencapai separuh batas atas? Pada bulan ke berapa emisi turun di bawah pembakaran? Masing-masing menanyakan di mana sebuah fungsi bernilai nol.

Fungsinya monoton dengan kurung yang diketahui — tepat di sini jaminan biseksi Persamaan (2.5) paling berharga: tidak mungkin gagal, dan biayanya diketahui sebelum dimulai.

Bentuk yang sama muncul di sisi harga: "pada harga koin berapa sebuah saldo cukup untuk dua belas perpanjangan Pro?" lagi-lagi fungsi monoton dan sebuah kurung. Buku: Subbab 2.10.

Memakainya Sungguhan, dan Kesalahan Umum

Rencana Anda sendiri: `python3 loan_rate.py 18000000 950000 24` untuk motor Rp18.000.000 dengan 24 angsuran Rp950.000. Bandingkan tawaran dengan bunga efektifnya.

Tanpa kurung — tabulasi dulu. **Kutub, bukan akar** — periksa bahwa $f(c)$ menjadi kecil. **Titik tak terdefinisi** — fungsi pinjaman membagi dengan r, jadi jauhkan $r = 0$ dari kurung.

Berhenti hanya karena $f(c)$ kecil — besarnya bergantung pada satuan. **Titik tengah sebagai $(a+b)/2$** — pakai $a + (b-a)/2$. **Bunga flat dibaca sebagai bunga efektif** — bedanya hampir dua kali lipat.

Program yang sama menjawab pertanyaan "masukan mana yang memberi keluaran ini?" apa pun setelah f diganti: kedalaman tangki, harga impas, tanggal target. Buku: Subbab 2.11–2.12.

Ringkasan

1. “Masukan mana yang memberi keluaran ini?” menjadi $f(x)=0$; untuk pinjaman, nilai kini dikurangi harga (Persamaan (2.2), Persamaan (2.3)).

2. Fungsi kontinu dengan $f(a)f(b)<0$ punya akar di $[a,b]$: sebuah kurung.

3. Biseksi membagi dua kurung (Pseudokode 2.1); batasnya (Persamaan (2.4)) memberi banyaknya langkah sebelumnya.

4. Regula falsi (Persamaan (2.6)) sering lebih cepat tetapi dapat merayap; Illinois (Persamaan (2.8)) memperbaikinya, orde sekitar 1,44.

5. Dimas: 19, 6, dan 5 langkah, bunga efektif 2,9229% sebulan (41,30% setahun) dibanding 1,67% flat; SciPy sepakat.

Metode pengurung lambat tetapi aman. Pertemuan berikutnya kita menukar sebagian keamanan dengan kecepatan yang jauh lebih besar. Buku: Subbab 2.13.

Pertemuan Berikutnya

Bab 3 — Mencari Akar II: Newton-Raphson, Secant, dan Iterasi Titik Tetap. Menandai tongkat ukur truk air yang tangkinya berbaring.

Metode terbuka yang tidak butuh kurung: memakai kemiringan, dan menggandakan digit yang benar di setiap langkah — ketika berhasil.

Sebelumnya: empat langkah biseksi dengan tangan untuk $x^2 - 2$ pada $[1, 2]$, dan kedalaman air di tangki bola untuk 500 sampai 4.000 liter.

Tangki bola butuh 21 iterasi biseksi untuk setiap volume, tepat seperti diramalkan Persamaan (2.5); penyelesaiannya dalam kedua bahasa ada di buku. Buku: Subbab 2.14–2.15.

Kuliah 3 — Mencari Akar II: Newton-Raphson, Secant, dan Iterasi Titik Tetap

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 3

Dosen: Irfan Subakti — yifana@gmail.com

Metode pengurung aman tetapi lambat: satu digit biner per langkah. Hari ini kita melepas kurung dan mengikuti bentuk fungsinya. Hadiahnya kecepatan — digit yang benar berlipat dua di setiap langkah — dan harganya, titik awal yang buruk dapat berujung ke mana-mana. Buku: Bab 3, Subbab 3.1–3.16.

Cakupan Pertemuan Ini

1. Masalahnya — tanda tongkat ukur untuk truk air yang tangkinya berbaring.

2. Newton-Raphson — ikuti garis singgung; mengapa galat dikuadratkan di setiap langkah.

3. Secant dan titik tetap — Newton tanpa turunan, dan iterasi $x = g(x)$.

4. Mengukur orde konvergensi — 2, 1,618, dan 1, terlihat di angka-angkanya.

5. Kegagalan, uji, SciPy — divergensi, siklus, dan garis singgung datar; serta cara memilih metode.

Ketiga metode terbuka masuk ke satu modul di samping metode pengurung Bab 2, dalam Java dan Python, dengan jejak iterasi yang identik. Buku: Bab 3.

Truk Air Pak Harun

Setiap kemarau sumur-sumur di lereng **Gunungkidul** menyusut, dan rumah tangga membeli air per tangki.

Tangki baja Pak Harun berbaring: silinder **panjang 3,20 m**, **jari-jari 0,90 m**, sedikit di atas 8.000 liter bila penuh.

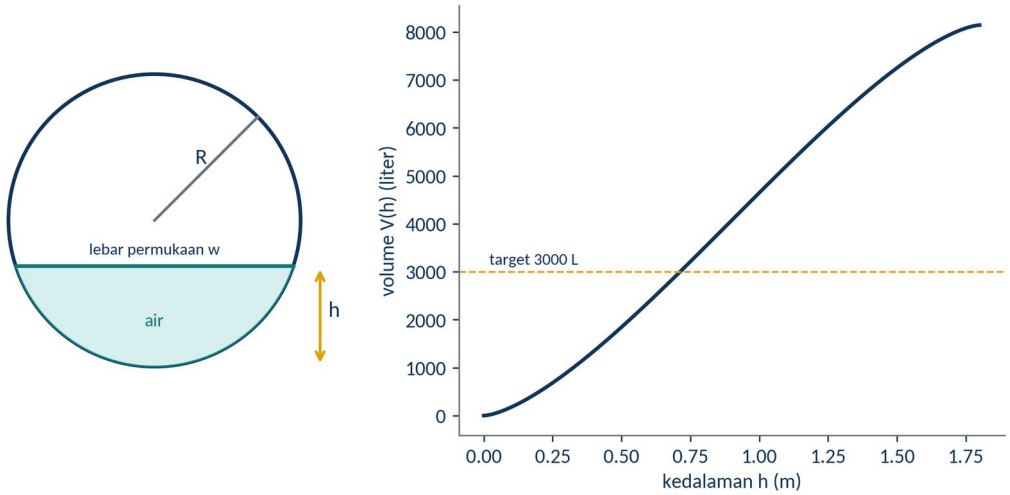
Pelanggan membeli 500, 1.000, 2.000, atau 3.000 liter. Di setiap perhentian ia harus tahu berapa yang sudah dikeluarkan — dengan **tongkat ukur** bertanda setiap 1.000 liter.

Untuk silinder berbaring tanda-tandanya tidak berjarak sama, dan rumusnya memberi volume dari kedalaman, tidak pernah kedalaman dari volume.

Tanda yang berjarak sama akan membuat sebagian pelanggan membayar lebih dan sebagian lagi dirugikan. Pertanyaannya, lagi-lagi, menjalankan rumus ke belakang: kedalaman mana yang memberi volume ini? Buku: Subbab 3.1.

Volume Tangki yang Berbaring

$$V(h) = L\left[R^2 \arccos\left(\frac{R-h}{R}\right) - (R-h)\sqrt{2Rh-h^2}\right] \tag{3.1}$$



Gambar 3.1 Tangki truk air dilihat dari ujungnya (kiri): air mengisi tembereng lingkaran dengan kedalaman h dan lebar permukaan w. Volume Persamaan (3.1) sebagai fungsi kedalaman (kanan) berbentuk S, paling curam pada setengah kedalaman.

Persamaan (3.1) adalah luas tembereng lingkaran dikali panjang. Gambar 3.1: kurvanya berbentuk S, sehingga seribu liter pertama dan terakhir memakan kedalaman lebih banyak daripada yang di tengah. Buku: Subbab 3.2.1.

Kemiringannya, lewat Jalan Pintas

Naikkan air sebanyak Δh yang sangat kecil: air tambahannya berupa lempeng tipis seluas permukaan, $w \times L$. Jadi

$$V'(h) = L\,w = 2L\sqrt{2Rh - h^2} \tag{3.2}$$

Terbesar pada setengah kedalaman, tempat permukaan paling lebar; nol di dasar dan di puncak, tempat permukaan menyusut menjadi garis.

Persamaan (3.2) dapat diperiksa dengan menurunkan Persamaan (3.1) suku demi suku, tetapi argumen lempeng menjelaskan mengapa jawabannya lebar permukaan. Nol-nolnya di dasar dan puncak akan penting bagi pengaman algoritma. Buku: Subbab 3.2.1.

Mengikuti Garis Singgung, dengan Tangan

Tebakan kotak untuk 3.000 L: 3.000 dari 8.143 liter adalah 36,8%, dan 36,8% dari 1,80 m adalah $h_0 = 0.6631$ m.

$V(0.6631) = 2.7231$ m^3: **kurang 277 liter**. Kemiringan di sana $V'(0.6631) = 5.5570$ m^3 per meter.

Kekurangan 0.2769 m^3 butuh kira-kira $0.2769/5.5570 = 0.0498$ m lagi: $h_1 = 0.7130$ m — sudah dalam **0,4 mm** dari tanda sebenarnya.

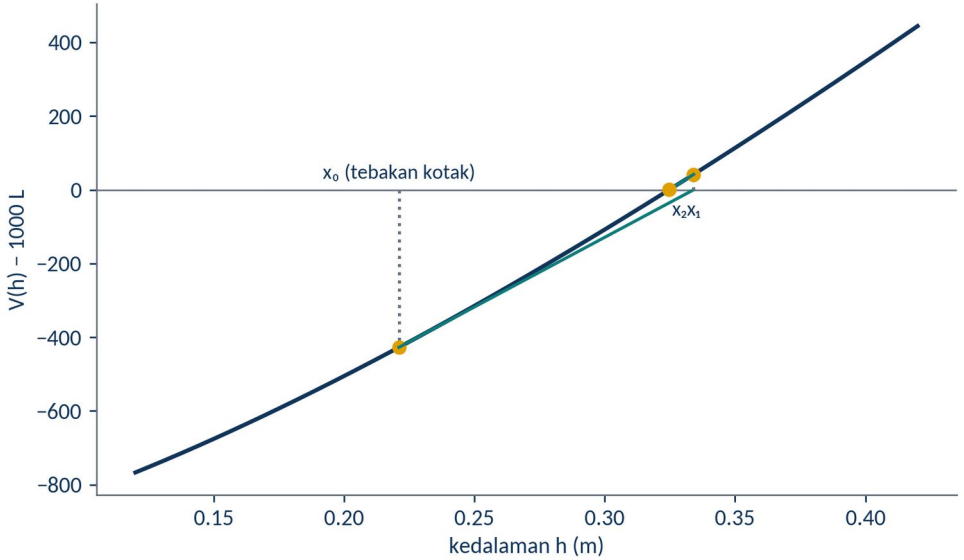
> **WAWASAN KUNCI:** Gagasan Newton dalam satu kalimat: di tebakan sekarang, anggap fungsinya adalah garis singgungnya, lalu lompat ke tempat garis itu mencapai target.

Tebakan kotak adalah jawaban yang diberikan model garis lurus; itu titik awal yang masuk akal, dan memilih titik awal yang masuk akal adalah separuh dari memakai metode terbuka dengan baik. Buku: Subbab 3.2.2.

Newton-Raphson

Ambil dua suku pertama rumus Taylor di sekitar x_k, samakan dengan nol, dan selesaikan untuk x:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{3.3}$$



Gambar 3.2 Metode Newton untuk tanda 1.000 liter: dari tebakan kotak x_0 garis singgung sedikit melampaui ke x_1, dan garis singgung kedua mendarat hampir tepat di akar x_2.

Persamaan (3.3) membuang suku yang memuat f''; suku yang dibuang itulah galatnya, seperti ditunjukkan bagian argumen berikutnya. Gambar 3.2: dua garis singgung hampir cukup. Buku: Subbab 3.3.1.

Algoritma dan Pengamannya

Pseudokode 3.1 Metode Newton-Raphson dengan tiga pengamannya.

```
NEWTON(f, df, x0, tol, maxIter)
 1  x = x0
 2  for k = 1 to maxIter
 3      d = df(x)
 4      if d == 0: error "f'(x) = 0: the tangent is flat"
 5      x_new = x - f(x) / d
 6      if |x_new - x| < tol: return x_new
 7      x = x_new
 8  error "no convergence in maxIter iterations"
```

Pseudokode 3.1: garis singgung datar ditolak, perubahan antariterasi menjadi uji berhenti, dan kehabisan iterasi adalah galat. Tebakan terakhir biseksi selalu di dalam kurung; tebakan terakhir Newton bisa di mana saja. Untuk metode kuadratik perubahan itu adalah taksiran yang sangat baik bagi galat iterasi yang lebih lama, sehingga iterasi yang dikembalikan jauh lebih baik lagi. Buku: Subbab 3.3.1.

Mengapa Begitu Cepat

Uraikan f di sekitar x_k dengan sisa eksak di akar r lalu kurangkan Persamaan (3.3). Dengan $e_k = x_k - r$:

$$e_{k+1} = \frac{f''(\xi_k)}{2f'(x_k)} e_k^2 \approx C\, e_k^2, C = \frac{f''(r)}{2f'(r)} \tag{3.4}$$

Sebuah metode punya **orde** p bila

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^p} = |C| \tag{3.5}$$

Persamaan (3.4): galat 10^{-3} menjadi sekitar 10^{-6}, lalu 10^{-12}: banyaknya digit benar berlipat dua. Dalam bahasa Persamaan (3.5), Newton berorde 2 di dekat akar sederhana; biseksi berorde 1. Konstanta C menjelaskan juga kapan Newton bermasalah: jika $f'(r)$ kecil, C besar, dan galat harus sudah kecil sebelum pengkuadratan mulai menolong. Itulah sebabnya titik awal yang baik begitu penting. Buku: Subbab 3.3.2.

Metode Secant

Tidak ada turunan? Pakai kemiringan garis melalui dua tebakan terakhir:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \tag{3.6}$$

Ordenya adalah rasio emas:

$$p = \frac{1+\sqrt{5}}{2} \approx 1.618 \tag{3.7}$$

Persamaan (3.6) mirip regula falsi, tetapi menyimpan dua titik terbaru, bukan kurung. Per langkah ia lebih lambat daripada Newton (Persamaan (3.7)), tetapi setiap langkah hanya butuh satu nilai fungsi, sehingga per satuan kerja sering lebih cepat. Dua langkah secant berharga sama dengan satu langkah Newton yang menghitung nilai dan turunan, dan dua langkah itu bersama-sama berorde $1.618^2 \approx 2.6$. *Secant butuh dua titik awal, dan tidak menjamin akarnya tetap terkurung. Buku: Subbab 3.4.*

Iterasi Titik Tetap

Tulis ulang $f(x)=0$ sebagai $x=g(x)$ lalu iterasikan:

$$x_{k+1}=g(x_k) \tag{3.8}$$

Galatnya dikali g' di setiap langkah. Jika nilai mutlak g' paling besar $K<1$ di dekat akar, maka

$$|x_k-r|\le\frac{K^k}{1-K}|x_1-x_0| \tag{3.9}$$

Persamaan (3.8) konvergen linear dengan laju sebesar ukuran $g'(r)$ (Persamaan (3.9)). Persamaan yang sama punya bentuk baik dan buruk: $x=2/x$ dan $x=x^2+x-2$ sama-sama bentuk lain dari $x^2=2$ dan keduanya gagal. Newton adalah iterasi titik tetap yang g'-nya nol di akar. Buku: Subbab 3.5.

Newton dalam Kode

Kode 3.1 Dari `roots2.py`: awal Newton-Raphson, dengan pengaman garis singgung datar.

```
def newton(f, df, x0, tol=1e-10, max_iter=50, trace=False):
    """x_new = x - f(x) / f'(x) until two iterates differ by less than tol.
    Returns (root, iterations); raises ArithmeticError if f'(x) is 0."""
    x = x0
    for k in range(1, max_iter + 1):
        d = df(x)
        if d == 0:
            raise ArithmeticError(f"f'(x) = 0 at x = {x}")
        x_new = x - f(x) / d
        if trace:
            _row(k, x_new, f(x_new), abs(x_new - x))
        if abs(x_new - x) < tol:
```

Kode 3.1: fungsi dan turunannya adalah parameter; kembarannya `Roots2.java` memakai dua `DoubleUnaryOperator`. `dipstick.py` mengirim $V(h) - \text{target}$ dan $V'(h)$ dari Persamaan (3.1) dan Persamaan (3.2). Buku: Subbab 3.6.

Menjalankannya: Tongkat Ukur

```
Tank: radius 0.9 m, length 3.2 m, full 8143.0 L

Newton-Raphson for 3000 L, starting from the box guess
  k                  x           f(x)  |x - x_old|
  1     0.712968084725   2.005405e-03    4.982e-02
  2     0.712612153675   8.618449e-08    3.559e-04
  3     0.712612138378   0.000000e+00    1.530e-08
  4     0.712612138378   0.000000e+00    0.000e+00

 volume     depth  Newton  secant
 1000 L  32.47 cm       5       6
 2000 L  52.96 cm       4       5
 3000 L  71.26 cm       4       4
 4000 L  88.76 cm       3       3
 5000 L 106.21 cm       4       4
 6000 L 124.33 cm       4       5
 7000 L 144.36 cm       4       6
 8000 L 171.37 cm       5       7
```

Perubahan 4.98×10^{-2}, 3.56×10^{-4}, 1.53×10^{-8} *memperlihatkan digit yang berlipat dua. Tanda-tandanya tidak rata: 1.000 liter pertama mengisi 32,47 cm, 1.000 liter di tengah hanya 17,45 cm. Newton butuh 3 sampai 5 langkah per tanda, secant 3 sampai 7; biseksi pada* $[0, 1.8]$ *akan butuh 31. Buku: Subbab 3.7.*

Mengukur Orde Konvergensi

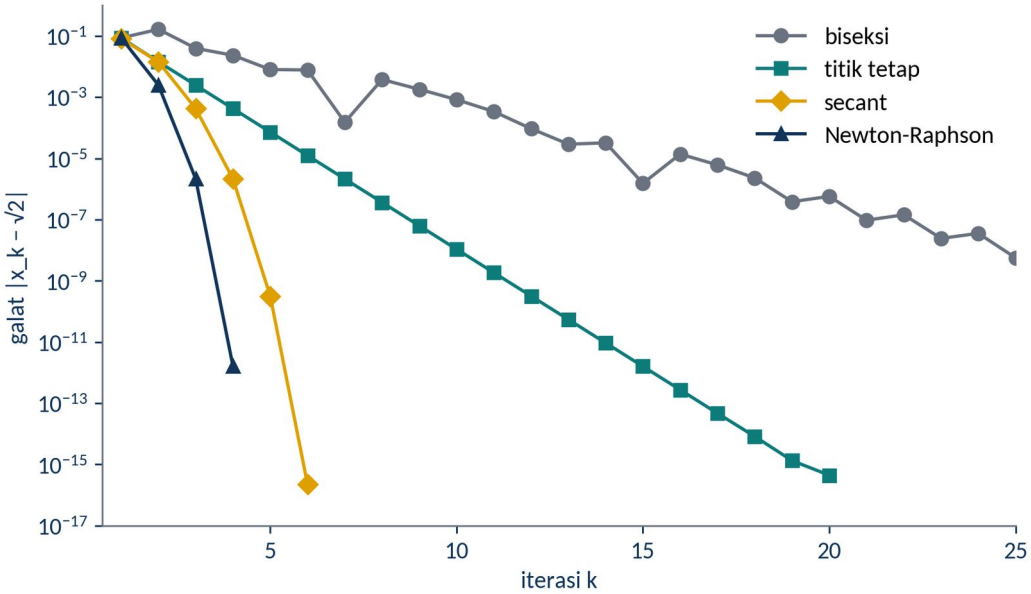
Bagi dua contoh berurutan dari Persamaan (3.5) lalu ambil logaritmanya; C lenyap:

$$p \approx \frac{\ln\left(|e_{k+1}|/|e_k|\right)}{\ln\left(|e_k|/|e_{k-1}|\right)} \tag{3.10}$$

```
Newton-Raphson, x0 = 1
  k              x_k        error    ratio  order p
  1   1.50000000000000   8.579e-02
  2   1.41666666666667   2.453e-03   0.0286
  3   1.41421568627451   2.124e-06   0.0009    1.984
  4   1.41421356237469   1.595e-12   0.0000    2.000
  5   1.41421356237310   0.000e+00
```

Persamaan (3.10) pada $\sqrt{2}$: orde terukur Newton 1,984, lalu 2,000. Program yang sama mengukur sekitar 1,6 untuk secant dan 1 untuk iterasi titik tetap, dengan rasio 0,1716. Teori menjadi pengukuran yang dapat Anda ulangi. Kolom rasio Newton turun menuju nol, ciri setiap metode yang lebih cepat daripada linear; pada secant orde terukurnya mula-mula berisik (2,025, 1,502, 1,667) lalu mengendap menuju 1,607, dekat rasio emas. Galat sejati di sini dapat dihitung karena $\sqrt{2}$ diketahui; pada masalah nyata, pakailah perubahan antariterasi sebagai penggantinya. Buku: Subbab 3.8.

Empat Metode dalam Satu Plot



Gambar 3.3 Galat empat metode yang menghitung $\sqrt{2}$: biseksi dan iterasi titik tetap turun sepanjang garis lurus (konvergensi linear), kurva secant dan Newton melengkung ke bawah makin curam (orde 1,618 dan 2). Sebuah garis berhenti ketika iterasinya sama dengan *double* terdekat dengan $\sqrt{2}$.

Gambar 3.3: pada sumbu galat logaritmik, garis lurus berarti konvergensi linear; kurva yang melengkung ke bawah berarti superlinear. Newton mencapai presisi penuh dalam 5 langkah, secant dalam 6, dan iterasi titik tetap akan butuh sekitar 20. Rasio titik tetap mengendap di 0,1716, tepat sebesar ukuran $g'(\sqrt{2})$ untuk bentuk $x = 1 + 1/(1+x)$: galatnya turun dengan faktor yang sama di setiap langkah, selamanya, sementara galat Newton runtuh. Buku: Subbab 3.8.

Ketika Newton Keliru

```
1. f(x) = cube root of x, x0 = 0.1: each step doubles the distance
   x1 = -0.200000
   x2 = 0.400000
   x3 = -0.800000
   x4 = 1.600000
   x5 = -3.200000
2. f(x) = x^3 - 2x + 2, x0 = 0: a cycle 0, 1, 0, 1, ...
   x1 = 1.000000
   x2 = 0.000000
   x3 = 1.000000
3. f(x) = x^2 - 4x + 5 has no real root; f'(2) = 0
   stopped: f'(x) = 0 at x = 2.0
```

> **PERINGATAN:** Newton tidak punya jaminan. Ia dapat divergen (terlalu curam di akar), berputar selamanya, berhenti di garis singgung datar, atau konvergen ke akar lain. Di akar ganda ia melambat menjadi konvergensi linear, membagi dua galatnya per langkah.

`failures.py` dan `Failures.java` *mencetak kasus-kasus yang sama; dari* $x_0 = -2$ *kubik yang sama konvergen dengan cepat. Selalu batasi iterasi, dan periksa bahwa hasilnya berada di rentang yang diharapkan dan* f *kecil di sana. Buku: Subbab 3.3.3 dan 3.9.*

Menguji, dan SciPy

```
ok      Newton finds sqrt 2 in at most 6 steps
ok      secant finds sqrt 2
ok      fixed point finds sqrt 2
ok      zero derivative is refused
ok      half-full tank: depth = radius
all tests passed
newton with fprime : 0.712612138378 m
newton, secant     : 0.712612138378 m
brentq on [0, 2R]  : 0.712612138378 m
```

Tangki setengah penuh harus punya kedalaman tepat sama dengan jari-jarinya, 0,90 m — pemeriksaan yang dapat kita pastikan tanpa menghitung. `newton` *dari SciPy memakai Newton bila diberi* `fprime` *dan metode secant bila tidak;* `brentq` *adalah metode pengurung. Ketiganya sepakat sampai dua belas digit. Buku: Subbab 3.10–3.11.*

Memilih Metode

Tabel 3.1 Pencari akar Bab 2 dan 3 dibandingkan: apa yang diperlukan masing-masing, seberapa cepat konvergen di dekat akar sederhana, dan apakah dijamin berhasil.

Metode	Butuh	Orde	Dijamin?
Biseksi	kurung	1 (laju 1/2)	ya, dengan banyak langkah yang diketahui
Regula falsi	kurung	1 (laju berubah)	ya, tetapi dapat merayap
Illinois	kurung	sekitar 1,44	ya
Secant	dua titik awal	sekitar 1,618	tidak
Newton-Raphson	titik awal dan f'	2	tidak
Titik tetap	bentuk $x = g(x)$	1	hanya jika g' kecil

Tabel 3.1: dalam praktik yang terbaik dari dua dunia adalah metode hibrida — pertahankan kurung demi keamanan dan ambil langkah terbuka selama tetap di dalamnya. Itulah yang dilakukan metode Brent di `brentq`. Buku: Subbab 3.12.

Metode Numerik dalam Praktik: Profitina

Banyak besaran dalam sebuah produk mengikuti kurva mulus yang naik, dan orang mengajukan pertanyaan yang menjalankan kurva itu **ke belakang**.

Langganan Pro Rp50.000 untuk tiga puluh hari dan dapat dibayar dengan PROFIT lewat oracle harga. Begitu biaya atau diskon bergantung tak linear pada jumlahnya, “berapa koin untuk rupiah sebanyak ini?” membalik sebuah kurva — masalah Pak Harun.

AI Visibility Engine melaporkan skor 0–100; “nilai mentah berapa yang menaikkan skor ke 70?” membalik kurva normalisasi.

Untuk kurva monoton yang mulus dengan turunan yang diketahui, Newton menjawab dalam segelintir langkah dari tebakan awal garis lurus; bila kurvanya hanya diketahui dari data, metode secant tidak butuh turunan. Buku: Subbab 3.13.

Kesalahan Umum

Mulai di tempat turunannya nol. $h=0$ memberi $V'(0)=0$. Mulailah dari tebakan kotak.

Tanpa batas banyaknya iterasi. Siklus berjalan selamanya; batasi dan bangkitkan galat.

Turunan yang salah. Newton tetap konvergen, tetapi hanya linear — yang menyembunyikan kesalahannya. Periksa f' dengan selisih hingga.

Bentuk titik tetap dipilih tanpa memeriksa g'. Hitung dulu di akar yang diharapkan.

Mengharapkan konvergensi kuadratik di akar ganda. Jika Newton lambat tanpa sebab, periksa apakah $f'(r)$ juga dekat nol.

Untuk silinder berbaring mana pun, program tongkat ukur hanya butuh R dan L yang baru. Ukur bagian dalam, bukan luar: tangki baja berdinding 4 mm punya jari-jari dalam 4 mm lebih kecil, dan tanda-tandanya ikut bergeser. Periksa tongkat yang sudah jadi sekali dengan volume yang diketahui, misalnya dengan mengisi tangki dari keran bermeter. Buku: Subbab 3.14–3.15.

Ringkasan

1. Metode terbuka mengikuti bentuk fungsi; cepat di dekat akar, tetapi butuh titik awal yang baik.

2. Newton mengikuti garis singgung (Persamaan (3.3)); galatnya dikuadratkan setiap langkah (Persamaan (3.4)).

3. Secant mengganti f' dengan kemiringan tali busur (Persamaan (3.6)), orde 1,618; titik tetap konvergen linear bila g' kecil.

4. Orde dapat diukur dari tiga galat berurutan (Persamaan (3.10)).

5. Kedelapan tanda tongkat ukur dalam 3 sampai 5 langkah Newton masing-masing, dibanding 31 untuk biseksi; SciPy sepakat.

Dua bab, enam pencari akar, satu pelajaran: kecepatan dan keamanan saling bertukar, dan titik awal yang baik membeli keduanya. Buku: Subbab 3.16.

Pertemuan Berikutnya

Bab 4 — Sistem Linear I: Eliminasi Gauss dan Pivoting. Berapa banyak setiap pupuk yang dibutuhkan sebatang pohon durian?

Banyak peubah sekaligus: eliminasi, substitusi mundur, dan mengapa urutan baris penting.

Sebelumnya: dua langkah Newton dengan tangan untuk $x^2 - 5$ dari $x_0 = 2$, dan suku bunga pinjaman Dimas dengan metode secant.

Metode secant menemukan 2,9229% milik Dimas dalam enam langkah tanpa turunan; penyelesaiannya ada di buku dalam kedua bahasa. Buku: Subbab 3.17–3.18.

Kuliah 4 — Sistem Linear I: Eliminasi Gauss dan Pivoting

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 4

Dosen: Irfan Subakti — yifana@gmail.com

Sampai sekarang kita punya satu peubah. Hari ini kita punya banyak, yang terikat oleh persamaan linear: harga yang tersembunyi dalam struk belanja dan gram empat pupuk untuk sebatang pohon durian. Kita membangun eliminasi Gauss, menghitung biayanya, mengamankannya dengan pivoting, dan belajar mengapa residu kecil tidak menjanjikan galat kecil. Buku: Bab 4, Subbab 4.1–4.15.

Cakupan Pertemuan Ini

1. Masalahnya — empat unsur hara dari empat pupuk untuk pohon durian Bu Rahma.

2. Eliminasi dan substitusi mundur — algoritmanya pada tiga struk, dan biayanya sekitar $\frac{2}{3}n^3$.

3. Pivoting — pivot nol, pivot sangat kecil, dan *partial pivoting*.

4. Menjalankannya — struk dan rencana pupuk, dalam Java dan Python.

5. Kondisi — bilangan kondisi, matriks Hilbert, dan jawaban NumPy.

Modul `gauss.py` / `Gauss.java` yang dibangun hari ini dipakai ulang di bab-bab metode iteratif, dekomposisi, interpolasi, dan kuadrat terkecil. Buku: Bab 4.

Pohon Durian Bu Rahma

Kebun buah dan durian dekat **Jember**: rambutan dan manggis di sepanjang jalan setapak, empat puluh pohon durian yang sudah berbuah enam tahun.

Setelah uji tanah, penyuluh pertanian (PPL) menuliskan kebutuhan setiap pohon produktif per tahun: **480 g N, 490 g P_2O_5, 660 g K_2O, 320 g S**.

Toko tani desa tidak menjual unsur hara murni, hanya karung: **NPK 15-15-15** (dengan 10% S), **DAP 18-46-0**, **KCl 0-0-60**, **ZA** (21% N, 24% S).

Berapa gram masing-masing, supaya keempat unsur hara pas semuanya?

Setiap pupuk membawa beberapa unsur hara sekaligus, sehingga mengubah satu karung mengubah beberapa jumlah. Empat peubah, empat syarat: sebuah sistem linear. Buku: Subbab 4.1.

Tiga Struk, Tiga Harga

Pemanasan yang lebih sederhana: tiga struk dari warung yang sama hanya mencantumkan total (harga dalam ribuan rupiah):

$$\begin{aligned} 2x_1 + x_2 + 2x_3 &= 82 \\ 5x_1 + 2x_2 + x_3 &= 128 \\ 3x_1 + 2x_2 + 3x_3 &= 132 \end{aligned} \tag{4.1}$$

Pisahkan angka dari peubah:

$$Ax = b,\ A = \begin{pmatrix} 2 & 1 & 2 \\ 5 & 2 & 1 \\ 3 & 2 & 3 \end{pmatrix},\ x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},\ b = \begin{pmatrix} 82 \\ 128 \\ 132 \end{pmatrix} \tag{4.2}$$

Persamaan (4.1): beras, minyak goreng, dan gula. Setiap persamaan ***linear****: setiap peubah hanya dikali bilangan lalu dijumlahkan. Persamaan (4.2): baris ke-i dari $Ax = b$ adalah struk ke-i; untuk hitungan tangan kita menulis matriks diperbesar $[A \mid b]$ dan bekerja pada baris-barisnya. Tiga operasi tidak pernah mengubah penyelesaian: menukar dua persamaan, mengalikan satu persamaan dengan bilangan tak nol, dan menambahkan kelipatan satu persamaan ke persamaan lain. Eliminasi hanya memakai yang pertama dan terakhir. Buku: Subbab 4.2.1.*

Eliminasi: Bersihkan Kolom di Bawah Pivot

Pada tahap k **pivot**-nya a_{kk}. Untuk setiap baris i di bawahnya, **pengali**-nya adalah

$$m_{ik} = \frac{a_{ik}}{a_{kk}} \tag{4.3}$$

dan baris i dikurangi m_{ik} kali baris k:

$$a_{ij} \leftarrow a_{ij} - m_{ik} a_{kj} \, (j = k+1, \ldots, n), \; b_i \leftarrow b_i - m_{ik} b_k \tag{4.4}$$

Persamaan (4.3) dan Persamaan (4.4): untuk struk, $m_{21} = 5/2$ dan $m_{31} = 3/2$ membersihkan kolom 1; lalu kolom 2 dibersihkan di bawah pivotnya, -0.5. Mengurangkan kelipatan satu persamaan dari persamaan lain tidak pernah mengubah penyelesaian. Buku: Subbab 4.3.1.

Tangga



Gambar 4.1 Eliminasi Gauss pada empat persamaan: di setiap tahap pivot (jingga) membersihkan kolomnya di bawahnya dan hanya entri berarsir yang berubah; substitusi mundur kemudian mengubah kolom kanan menjadi penyelesaiannya.

Setelah matriksnya segitiga atas, selesaikan dari peubah terakhir ke yang pertama — **substitusi mundur**:

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}\,x_j\right), i = n, n-1, \ldots, 1 \tag{4.5}$$

Gambar 4.1: pekerjaan menyusut di setiap tahap, satu baris dan satu kolom lebih sedikit. Persamaan (4.5): untuk struk, $-4\,x_3 = -68$ *memberi* $x_3 = 17$*, lalu* $x_2 = 18$ *dan* $x_1 = 15$*. Buku: Subbab 4.3.2.*

Algoritmanya

Pseudokode 4.1 Eliminasi Gauss dengan *partial pivoting* opsional (baris 2 sampai 4) dan substitusi mundur.

```
GAUSS(A, b, pivot)                  // A is n x n, b has n entries
 1  for k = 1 to n - 1                          // forward elimination
 2      if pivot:
 3          p = the row i >= k with the largest |a_ik|
 4          swap rows k and p of A and b
 5      if a_kk == 0: error "zero pivot"
 6      for i = k + 1 to n
 7          m = a_ik / a_kk                     // the multiplier
 8          a_ik = 0
 9          for j = k + 1 to n: a_ij = a_ij - m * a_kj
10          b_i = b_i - m * b_k
11  if a_nn == 0: error "zero pivot"
12  for i = n downto 1                          // back substitution
13      s = b_i
14      for j = i + 1 to n: s = s - a_ij * x_j
15      x_i = s / a_ii
16  return x
```

Pseudokode 4.1: tiga perulangan bersarang pada eliminasi, dua pada substitusi mundur. Perulangan itu memberi tahu biayanya sebelum kita menjalankan apa pun. Buku: Subbab 4.3.2.

Menghitung Pekerjaan

$$\text{flops}(n) = \frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n \approx \frac{2}{3}n^3 \tag{4.6}$$

n	flops counted	formula	2n^3/3
3	28	28	18
10	805	805	667
50	87,025	87,025	83,333
100	681,550	681,550	666,667
200	5,393,100	5,393,100	5,333,333

Persamaan (4.6): `op_count.py` *menghitung setiap operasi floating point dan cocok persis dengan rumusnya. Menggandakan* n *mengalikan pekerjaan delapan kali. Laptop menyelesaikan 1.000 persamaan dalam jauh kurang dari satu detik; substitusi mundur hanya berbiaya sekitar* n^2*. Buku: Subbab 4.3.3.*

Tanpa Penyelesaian Tunggal, dan Determinan

Pivot nol dengan hanya nol di bawahnya: matriksnya **singular** — tanpa penyelesaian, atau tak hingga banyaknya. “1 kg beras dan 2 L minyak Rp51.000” dan “2 kg beras dan 4 L minyak Rp102.000” mengatakan hal yang sama dua kali. Eliminasi memberi determinan secara cuma-cuma, dengan s banyaknya pertukaran baris:

$$\det A = (-1)^{s} \prod_{k=1}^{n} a_{kk}^{(\text{final})} \tag{4.7}$$

Persamaan (4.7): untuk struk, pivot $2, -0.5, -4$ *tanpa pertukaran memberi* $\det A = 4$*. Nol berarti singular, tetapi besarnya determinan yang tak nol hampir tidak berkata apa-apa tentang seberapa dapat dipercaya penyelesaiannya. Bila hanya* b *yang berubah, seperti untuk pohon beda umur, eliminasi sekali dan simpan pengalinya — gagasan LU di Bab 6. Buku: Subbab 4.3.4–4.3.5.*

Pivot Nol dan Pivot Sangat Kecil

Susun ulang struk sehingga yang pertama tanpa beras; eliminasi biasa langsung membagi dengan nol:

$$\begin{pmatrix} 0 & 1 & 2 \\ 5 & 2 & 1 \\ 3 & 2 & 3 \end{pmatrix} x = \begin{pmatrix} 52 \\ 128 \\ 132 \end{pmatrix} \tag{4.8}$$

Lebih buruk, pivot yang *sangat kecil* memberi jawaban salah tanpa peringatan:

$$10^{-20} x + y = 1 \,, \; x + y = 2 \tag{4.9}$$

Persamaan (4.8) punya harga-harga yang sama sebagai penyelesaiannya; manusia cukup memakai persamaan lain lebih dulu. Persamaan (4.9): pengali 10^{20} *menelan angka 1 dan 2, persis seperti struk kecil lenyap ke dalam jumlah float yang besar di Bab 1; substitusi mundur lalu mengembalikan* $x = 0$*, bukan 1. Buku: Subbab 4.4.1–4.4.2.*

Partial Pivoting

Sebelum mengeliminasi kolom k, naikkan baris yang entrinya terbesar di kolom itu:

$$p = \arg\max_{i \geq k} |a_{ik}|, \text{swap rows } k \text{ and } p \tag{4.10}$$

WAWASAN KUNCI: Dengan *partial pivoting* setiap pengali paling besar 1, sehingga tidak ada baris yang diperbesar sebelum dikurangkan. Pertukaran itu tidak berbiaya flop, dan menghilangkan pivot nol sekaligus pivot sangat kecil.

Persamaan (4.10) adalah baris 2 sampai 4 dari Pseudokode 4.1. Wilkinson menunjukkan bahwa entri masih dapat tumbuh, dalam kasus terburuk 2^{n-1} kali, tetapi matriks seperti itu begitu langka sehingga eliminasi dengan partial pivoting dianggap stabil dalam praktik; LAPACK, dan karenanya NumPy, memakainya untuk sistem umum. Buku: Subbab 4.4.3.

Pivoting dalam Kode

```
1. The receipts with rice missing from the first one
   without pivoting: zero pivot in column 1
   with pivoting   : 15.000000, 18.000000, 17.000000
2. A tiny pivot: 1e-20 x + y = 1, x + y = 2 (answer near 1, 1)
   without pivoting: x = 0.000000, y = 1.000000
   with pivoting   : x = 1.000000, y = 1.000000
```

`pivot_demo.py` dan `PivotDemo.java` mencetak baris-baris yang sama. Kegagalan pertama mengumumkan dirinya; yang kedua tidak — $x = 0$ jelas salah, dan tidak ada apa pun di keluaran yang mengatakannya. Buku: Subbab 4.9.

Menjalankannya: Struk

```
start
      2.0000     1.0000     2.0000 |    82.0000
      5.0000     2.0000     1.0000 |   128.0000
      3.0000     2.0000     3.0000 |   132.0000
after eliminating column 1
      2.0000     1.0000     2.0000 |    82.0000
      0.0000    -0.5000    -4.0000 |   -77.0000
      0.0000     0.5000     0.0000 |     9.0000
after eliminating column 2
      2.0000     1.0000     2.0000 |    82.0000
      0.0000    -0.5000    -4.0000 |   -77.0000
      0.0000     0.0000    -4.0000 |   -68.0000
rice  : Rp 15,000
oil   : Rp 18,000
sugar : Rp 17,000
largest residual: 0.000e+00
```

`receipts.py` *membaca* `data/receipts.csv` *dan mencetak setiap tahap, yang cocok dengan hitungan tangan.* `gauss.py` *menyalin matriks dan ruas kanan lebih dulu, sehingga data pemanggil tidak pernah berubah. Residu* $b - A\,x$ *di sini tepat nol. Buku: Subbab 4.5–4.6.*

Pupuk Kebun Durian

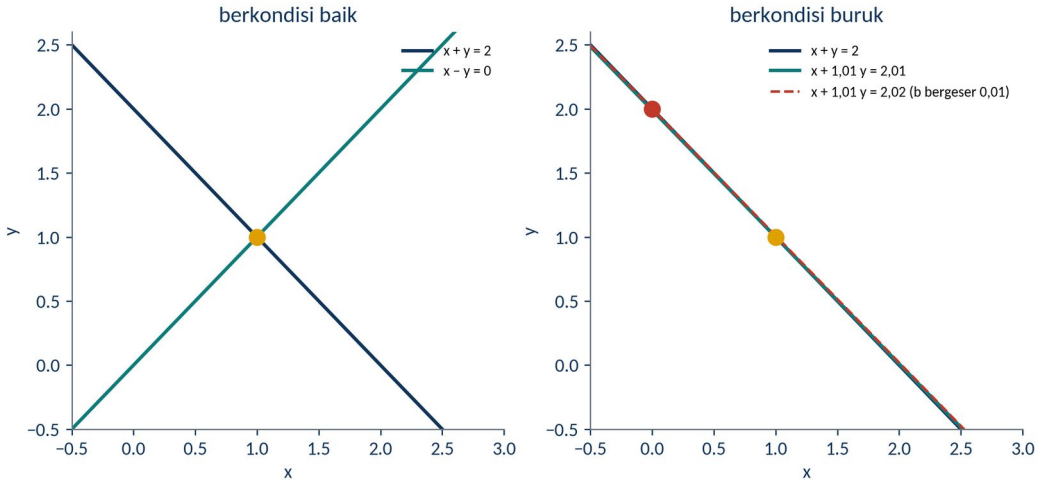
Satu baris per unsur hara, satu kolom per pupuk (fraksi massa), target dalam gram per pohon per tahun:

$$\begin{pmatrix} 0.15 & 0.18 & 0 & 0.21 \\ 0.15 & 0.46 & 0 & 0 \\ 0.15 & 0 & 0.60 & 0 \\ 0.10 & 0 & 0 & 0.24 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 480 \\ 490 \\ 660 \\ 320 \end{pmatrix} \tag{4.11}$$

```
fertilizer      g per tree   kg for 40
NPK 15-15-15        2171.4       86.86
DAP 18-46-0          357.1       14.29
KCl 0-0-60           557.1       22.29
ZA 21-0-0-24S        428.6       17.14
largest residual: 1.421e-14 g
```

Persamaan (4.11): berkasnya berisi satu pupuk per baris, tetapi matriksnya butuh satu unsur hara per baris — tuliskan arti baris dan kolom sebelum menyusunnya. Program juga memeriksa setiap unsur: N diinginkan 480,0 g, didapat 480,000 g. Untuk empat puluh pohon, Bu Rahma membeli sekitar 87 kg NPK. Buku: Subbab 4.7.

Residu Kecil Bukan Galat Kecil



Gambar 4.2 Sistem berkondisi baik (kiri) dan sistem berkondisi buruk (kanan): bila garis-garisnya hampir sejajar, perubahan 0,01 pada data menggeser penyelesaian dari $(1,1)$ ke $(0,2)$.

Gambar 4.2: dua persamaan adalah dua garis, dan penyelesaiannya titik potongnya. Bila garis-garisnya hampir sejajar, perubahan kecil pada data memindahkan titik potong jauh sekali, sementara setiap titik di sekitarnya masih hampir memenuhi kedua persamaan. Buku: Subbab 4.8.

Bilangan Kondisi

$$\kappa(A) = \|A\|\,\|A^{-1}\| \tag{4.12}$$

$$\frac{\|\delta x\|}{\|x\|} \le \kappa(A)\,\frac{\|\delta b\|}{\|b\|} \tag{4.13}$$

```
 n     cond(H)    residual  max error
 2   2.70e+01   1.11e-16   6.66e-16
 4   2.84e+04   1.67e-16   6.11e-13
 6   2.91e+07   8.33e-17   5.26e-10
 8   3.39e+10   1.67e-16   4.19e-07
10   3.54e+13   1.80e-16   3.18e-04
12   3.83e+16   4.30e-16   3.56e-01
```

Persamaan (4.12), Persamaan (4.13): galat relatif pada data diperbesar sampai κ kali; harapkan kehilangan sekitar $\log_{10}\kappa$ digit. Matriks Hilbert memperlihatkannya: residu tetap di tingkat pembulatan untuk setiap ukuran, tetapi pada $n=12$ galatnya 36% — tidak ada lagi yang benar. Buku: Subbab 4.8.

Menguji, dan NumPy

```
ok      receipts give 15, 18, 17
ok      the input is not changed
ok      pivoting handles a zero pivot
ok      a singular matrix is refused
ok      28 flops for n = 3
all tests passed
np.linalg.solve : [2171.4  357.1  557.1  428.6]
cond(A), 2-norm : 265.7
cond(A), inf    : 446.8
Hilbert n =  6: cond 2.91e+07, max error 2.60e-10
Hilbert n = 12: cond 3.99e+16, max error 5.16e-01
```

Uji flop memeriksa Persamaan (4.6) pada $n=3$*:* $18+13.5-3.5=28$*.* `solve` *dari NumPy memanggil* `gesv` *dari LAPACK, yaitu eliminasi dengan partial pivoting dalam bentuk faktorisasi LU, dan memberi rencana yang sama. Matriks pupuk punya* $\kappa_\infty = 447$*: berkondisi agak buruk, sehingga target yang dibulatkan cukup banyak mengubah rencana. Bahkan LAPACK tidak dapat menyelamatkan Hilbert 12. Buku: Subbab 4.10–4.11.*

Metode Numerik dalam Praktik: Profitina

AI Visibility Engine milik Profitina memberi skor sebuah situs dari banyak sinyal — konten yang mudah diekstrak, data terstruktur, kesehatan teknis — dan menjelaskan setiap sinyal.

Skor dari banyak sinyal adalah jumlah berbobot. Platform tidak menerbitkan bobotnya, tetapi “bobot mana yang mereproduksi penilaian rujukan ini?” berbentuk sistem linear: satu persamaan per situs rujukan, satu peubah per bobot.

Dua sinyal yang mengukur hal yang hampir sama membuat sistemnya berkondisi buruk: residu kecil, bobot besar dan tidak stabil, kadang negatif.

Obatnya sama dengan di kebun: ukur bilangan kondisinya, gabungkan atau buang sinyal yang saling menduplikasi, dan pakai metode yang stabil. Dengan rujukan lebih banyak daripada bobot, masalahnya menjadi kuadrat terkecil di Bab 8. Buku: Subbab 4.12.

Memakainya Sungguhan

Pupuk Anda sendiri. Ganti baris-baris `fertilizer.csv` dengan karung yang dijual toko Anda dan target dari uji tanah Anda. Jumlah negatif berarti target tidak dapat dipenuhi dengan karung-karung itu: tambahkan pupuk lain (urea, “46-0-0”), atau terima kecocokan hampiran (Bab 8).

Periksa kepekaannya. Ubah setiap target sebesar ketidakpastiannya, mungkin 5%, lalu selesaikan lagi. Ayunan besar berarti pupuk-pupuknya saling tumpang tindih; pupuk berunsur tunggal membuat rencananya kokoh.

Campuran di mana-mana. Pakan ayam atau lele dari kandungan protein, energi, dan serat, atau campuran beberapa mutu beras menuju harga sasaran: semuanya $A\,x = b$.

Struk Anda sendiri. Sejumlah n struk dengan n barang yang harganya tetap memberi harga-harganya, asalkan struk-struk itu sungguh berbeda: dua struk yang sebanding (yang satu tepat dua kali yang lain) membuat matriksnya singular.

Program kita cukup sampai beberapa ratus peubah. Untuk ribuan, `solve` dari NumPy memakai metode yang sama dengan perulangan dalam yang jauh lebih cepat; untuk sistem jarang dengan jutaan peubah, lihat Bab 5. Buku: Subbab 4.13.

Kesalahan Umum

Eliminasi tanpa pivoting. Gagal pada pivot nol, diam-diam salah pada pivot sangat kecil.

Tertukar baris dan kolom. Putuskan dulu arti baris i dan kolom j.

Menilai penyelesaian dari residunya. Taksirlah bilangan kondisinya.

Menghitung invers untuk menyelesaikan sistem. Tiga kali pekerjaan dan kurang teliti; selesaikan $Ax = b$ secara langsung.

Memercayai besarnya determinan. 10^{-3} kali matriks identitas punya determinan sangat kecil, padahal kondisinya sempurna.

Dan jangan menimpa matriks pemanggil diam-diam: salin dulu, seperti `solve` *kita, atau nyatakan bahwa matriksnya dirusak. Untuk kebun, bulatkan rencananya ke yang dapat ditimbang lalu hitung ulang unsur hara yang sungguh diterima pohon. Buku: Subbab 4.13–4.14.*

Ringkasan

1. n persamaan linear dengan n peubah ditulis $A\,x = b$ (Persamaan (4.2)).

2. Eliminasi (Persamaan (4.3), Persamaan (4.4)) membuat A segitiga atas; substitusi mundur (Persamaan (4.5)) menyelesaikannya; Pseudokode 4.1 menggabungkan keduanya dengan sekitar $\frac{2}{3}n^3$ flop.

3. *Partial pivoting* (Persamaan (4.10)) menjaga setiap pengali paling besar 1 dan stabil dalam praktik; determinan adalah hasil kali pivot bertanda.

4. $\kappa(A)$ membatasi pembesaran galat data; sekitar $\log_{10}\kappa$ digit hilang; residu kecil bukan galat kecil.

5. Struk: Rp15.000, Rp18.000, Rp17.000. Kebun: 2.171, 357, 557, dan 429 g per pohon, sesuai NumPy.

Eliminasi adalah metode langsung: sejumlah langkah yang tetap memberi jawabannya. Pertemuan berikutnya kita bertemu metode yang memperbaiki sebuah tebakan. Buku: Subbab 4.15.

Pertemuan Berikutnya

Bab 5 — Sistem Linear II: Jacobi, Gauss-Seidel, dan SOR. Sistem besar dan jarang, tempat eliminasi terlalu mahal.

Mulai dari tebakan dan perbaiki persamaan demi persamaan; kapan konvergen, dan seberapa cepat?

Sebelumnya: selesaikan struk beras, telur, dan bawang merah dengan tangan, dan jalankan sendiri percobaan Hilbert.

Mengerjakan satu eliminasi dengan tangan, dengan setiap pengali dituliskan, adalah cara terbaik untuk menguasai algoritmanya; penyelesaian semua soal latihan ada di buku. Buku: Subbab 4.16–4.17.

Kuliah 5 — Sistem Linear II: Jacobi, Gauss-Seidel, dan SOR

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 5

Dosen: Irfan Subakti — yifana@gmail.com

Eliminasi Gauss berbiaya sekitar $\frac{2}{3}n^3$ operasi dan mengisi nol-nol sebuah matriks. Banyak sistem nyata besar tetapi jarang: setiap persamaan hanya melibatkan sedikit peubah. Hari ini kita menyelesaikannya dengan memperbaiki tebakan, sapuan demi sapuan, dan mempelajari dengan tepat kapan dan seberapa cepat cara itu berhasil. Buku: Bab 5, Subbab 5.1–5.12.

Cakupan Pertemuan Ini

1. Masalahnya — suhu tunak wajan martabak.

2. Tiga sapuan — Jacobi, Gauss-Seidel, dan *successive over-relaxation* (SOR), dengan tangan dan secara tepat.

3. Konvergensi — pemisahan matriks, jari-jari spektral, dominansi diagonal, dan ω optimal Young.

4. Menjalankannya — 400 peubah tanpa pernah membangun matriksnya; banyak sapuan dan peta suhu.

5. Memeriksa dan berhenti — pemecah jarang SciPy, dan mengapa pergeseran kecil bukan galat kecil.

Kedua program, `iterative_solvers` dan `griddle`, ada dalam Java dan Python dan mencetak teks yang persis sama. Buku: Bab 5.

Wajan Pak Joni

Gerobak martabak dan roti bakar Pak Joni punya wajan besi datar berbentuk persegi, **40 cm setiap sisinya**.

Kompor panjang berada di bawah tepi **kiri**, kompor lebih kecil menjaga tepi **depan** tetap hangat; tepi belakang dan kanan tidak kena api.

Setelah seperempat jam pelatnya berhenti bertambah panas: **keadaan tunak**, panas yang masuk sama dengan panas yang keluar.

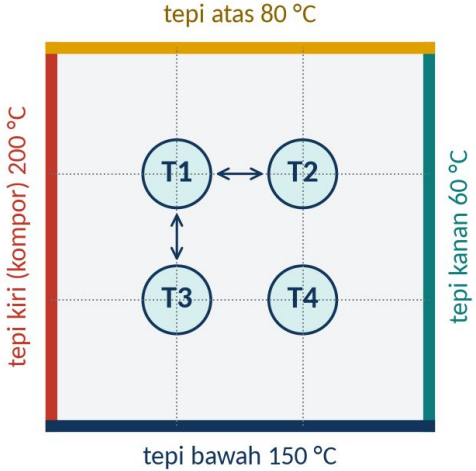
Di bagian mana pelat bersuhu **140 °C sampai 170 °C**, zona tempat martabak matang paling baik?

Suhu tepi diketahui dari kompor dan udara; bagian dalam tidak diketahui. Pertanyaannya adalah peta bagian dalam, dan peta itu adalah penyelesaian sebuah sistem linear yang besar. Buku: Subbab 5.1.

Gagasannya: Setiap Titik adalah Rata-rata Tetangganya

Dalam keadaan tunak panas yang mengalir dari keempat tetangga berjumlah nol, sehingga setiap suhu titik dalam adalah rata-ratanya:

$$T = \frac{T_{\text{left}} + T_{\text{right}} + T_{\text{up}} + T_{\text{down}}}{4} \tag{5.1}$$



Gambar 5.1 Wajan paling kasar: empat suhu titik dalam yang belum diketahui, masing-masing rata-rata keempat tetangganya; dua tetangga selalu berupa tepi yang suhunya diketahui.

Persamaan (5.1) adalah bentuk diskret persamaan panas dalam keadaan tunak; Bab 11 menurunkannya dengan lebih cermat. Gambar 5.1: dengan hanya empat titik dalam, dua tetangga setiap titik adalah tepi yang suhunya kita ketahui. Buku: Subbab 5.2.1.

Empat Persamaan

$$\begin{aligned} 4T_1 - T_2 - T_3 &= 280 \\ -T_1 + 4T_2 - T_4 &= 140 \\ -T_1 + 4T_3 - T_4 &= 350 \\ -T_2 - T_3 + 4T_4 &= 210 \end{aligned} \tag{5.2}$$

Setiap persamaan punya koefisien diagonal besar, 4, dan paling banyak dua tetangga dengan koefisien -1.

Pada kisi 20×20 ada 400 peubah tetapi tetap paling banyak **5 entri tak nol per baris**: 1.920 dari 160.000 entri.

Persamaan (5.2) cukup kecil untuk Bab 4, tetapi saudara-saudaranya yang lebih besar tidak: eliminasi akan mengisi nol dengan bilangan tak nol dan berbiaya $\frac{2}{3}n^3$. Metode iteratif hanya pernah menyentuh entri tak nol. Satu sapuan menyentuh setiap entri tak nol sekali, jadi pada wajan biayanya sekitar $5N$ perkalian untuk $N = n^2$ peubah, dan yang perlu disimpan hanya peubahnya sendiri: matriksnya tidak pernah dibangun, karena Persamaan (5.1) adalah matriksnya. Buku: Subbab 5.2.2.

Jacobi dengan Tangan: Semua Diperbarui dari Gambar Lama

Tebak 0 °C di seluruh bagian dalam. Terapkan Persamaan (5.1) ke setiap titik, dengan tebakan *lama* untuk setiap tetangga:

$T_1 = (280+0+0)/4 = 70$, $T_2 = 140/4 = 35$, $T_3 = 350/4 = 87.5$, $T_4 = 210/4 = 52.5$.

Sapuan kedua dengan nilai-nilai itu: $T_1 = (280+35+87.5)/4 = 100.625$, dan seterusnya. Tepi-tepi "membocorkan" panasnya ke dalam.

Gauss-Seidel langsung memakai setiap nilai baru begitu dihitung. **SOR** melangkah sedikit lebih jauh daripada saran Gauss-Seidel, dengan faktor ω.

Tebakannya buruk sekali, tetapi tidak apa-apa. Setiap sapuan lebih sedikit salahnya daripada sebelumnya, karena suhu tepi yang diketahui masuk di setiap langkah. Buku: Subbab 5.2.3–5.2.5.

Metode-metodenya secara Tepat

Selesaikan persamaan ke-*i* untuk peubahnya sendiri. **Jacobi** hanya memakai sapuan sebelumnya:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}\right), i = 1, \ldots, n \tag{5.3}$$

Gauss-Seidel memakai nilai baru dari peubah yang sudah diperbarui:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}\right) \tag{5.4}$$

Persamaan (5.3) butuh dua larik, lama dan baru; Persamaan (5.4) menimpa satu larik di tempat dan butuh memori lebih sedikit. Menulis Jacobi ke larik yang sedang dibaca tanpa sengaja mengubahnya menjadi Gauss-Seidel. Sebaliknya, Jacobi punya satu keunggulan: setiap peubah dalam satu sapuan dapat dihitung terpisah dari yang lain, sehingga pekerjaannya mudah dibagi ke banyak inti prosesor, gagasan yang kembali di Bab 15. Buku: Subbab 5.3.1.

Successive Over-Relaxation

$$x_i^{(k+1)} = (1-\omega)\,x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\,x_j^{(k+1)} - \sum_{j>i} a_{ij}\,x_j^{(k)}\right) \tag{5.5}$$

Pseudokode 5.1 *Successive over-relaxation*; dengan omega = 1 ia adalah metode Gauss-Seidel.

```
SOR(A, b, x, omega, tol, maxSweeps)
 1  for k = 1 to maxSweeps
 2      change = 0
 3      for i = 1 to n
 4          s = b[i] - (sum over j != i of A[i][j] * x[j])
 5          new = (1 - omega) * x[i] + omega * s / A[i][i]
 6          change = max(change, |new - x[i]|)
 7          x[i] = new
 8      if change < tol
 9          return x, k
10  return x, maxSweeps        // not converged: report it
```

Persamaan (5.5) mencampur nilai lama dengan nilai Gauss-Seidel; dengan $\omega > 1$ ia sengaja melampaui, karena koreksi Gauss-Seidel pada masalah semacam ini selalu terlalu malu-malu. Pseudokode 5.1 berhenti bila tidak ada peubah yang bergeser sebesar `tol` dalam satu sapuan, dan melaporkan kegagalan dengan jujur. Buku: Subbab 5.3.1.

Pemisahan Matriks

Tulis $A = D - L - U$ (diagonal, segitiga bawah murni, segitiga atas murni). Ketiga metode itu adalah

$$\begin{aligned} \text{Jacobi:} \quad & D\,x^{(k+1)} = (L+U)\,x^{(k)} + b \\ \text{Gauss-Seidel:} \quad & (D-L)\,x^{(k+1)} = U\,x^{(k)} + b \\ \text{SOR:} \quad & (D-\omega L)\,x^{(k+1)} = ((1-\omega)D + \omega U)\,x^{(k)} + \omega b \end{aligned} \tag{5.6}$$

Semuanya berbentuk **iterasi stasioner**:

$$x^{(k+1)} = T\,x^{(k)} + c\,, T = M^{-1}N \tag{5.7}$$

Persamaan (5.6): setiap baris berbentuk $M\,x^{(k+1)} = N\,x^{(k)} + c$ dengan $A = M - N$. Pemisahan yang baik membuat M mudah diselesaikan: diagonal, atau segitiga bawah (substitusi maju). Persamaan (5.7) menaruh ketiganya dalam satu bentuk, sehingga satu teori mencakup semuanya. Buku: Subbab 5.3.2.

Kapan Konvergen

Kurangkan penyelesaian eksak: galatnya dikali T setiap sapuan:

$$e^{(k+1)} = T\,e^{(k)}, \text{so } e^{(k)} = T^k e^{(0)} \tag{5.8}$$

Konvergen dari setiap titik awal tepat bila **jari-jari spektral** di bawah 1:

$$\rho(T) = \max_i |\lambda_i| \tag{5.9}$$

Uji yang dapat dilakukan dengan mata — **dominansi diagonal tegas**:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \text{ for every } i \tag{5.10}$$

Persamaan (5.8) dan Persamaan (5.9): ρ adalah faktor penyusutan galat per sapuan dalam jangka panjang. Persamaan (5.10) menjamin Jacobi dan Gauss-Seidel; matriks simetris definit positif menjamin Gauss-Seidel dan SOR untuk $0 < \omega < 2$. Buku: Subbab 5.4.1–5.4.2.

Seberapa Cepat: Teori Young

Untuk wajan $n \times n$ jari-jari Jacobi diketahui dengan tepat:

$$\rho_J = \cos\frac{\pi}{n+1} \tag{5.11}$$

$$\rho_{GS} = \rho_J^2, \omega_{\text{opt}} = \frac{2}{1+\sqrt{1-\rho_J^2}}, \rho\left(T_{\omega_{\text{opt}}}\right) = \omega_{\text{opt}} - 1 \tag{5.12}$$

Banyak sapuan untuk menyusutkan galat awal E_0 menjadi ε:

$$k \approx \frac{\ln\left(\varepsilon / E_0\right)}{\ln \rho} \tag{5.13}$$

Persamaan (5.11): dekat 1 untuk kisi halus, sehingga Jacobi lambat. Persamaan (5.12): satu sapuan Gauss-Seidel senilai dua sapuan Jacobi, dan SOR optimal jauh lebih baik. Persamaan (5.13): banyak sapuan tumbuh seperti n^2 untuk Jacobi dan Gauss-Seidel tetapi hanya seperti n untuk SOR optimal. Buku: Subbab 5.4.3–5.4.4.

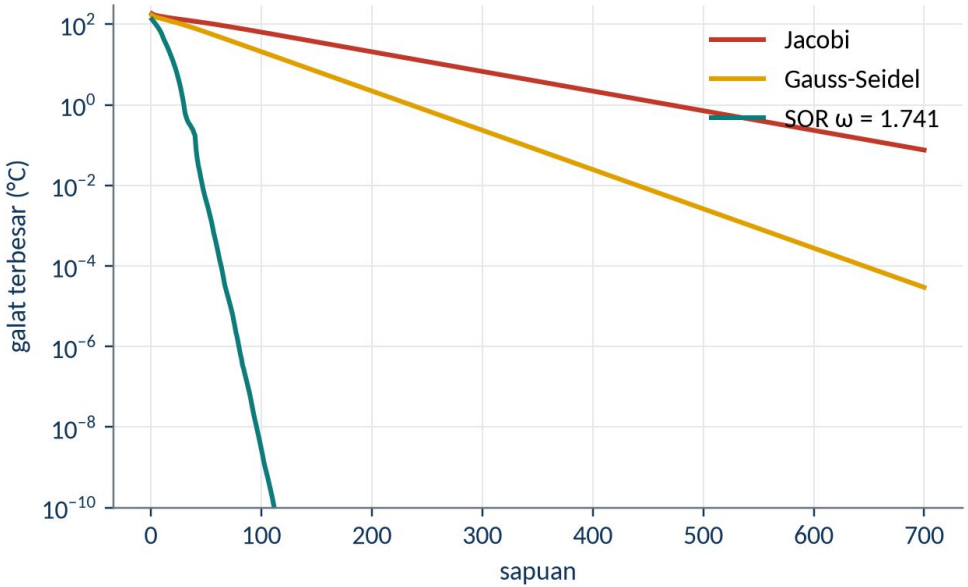
Menjalankannya: Sistem Kecil

```
diagonally dominant: True
Jacobi
   k       x1        x2        x3        x4     change
   1   70.0000   35.0000   87.5000   52.5000    87.5000
   2  100.6250   65.6250  118.1250   83.1250    30.6250
   3  115.9375   80.9375  133.4375   98.4375    15.3125
   4  123.5938   88.5938  141.0938  106.0938     7.6562
   5  127.4219   92.4219  144.9219  109.9219     3.8281
   6  129.3359   94.3359  146.8359  111.8359     1.9141
  converged after 27 iterations, residual 1.825e-06
  T = 131.249999 96.249999 148.749999 113.749999
```

Dua baris pertama adalah hitungan tangan kita; mulai baris 3 pergeserannya terbagi dua setiap kali, seperti diramalkan $\rho(T_J)=0.5$ *untuk sistem ini. Gauss-Seidel butuh 15 sapuan, SOR dengan* $\omega=1.1$ *(dekat optimum 1,072) butuh 10, dan* $\omega=1.5$ *kelewatan dan butuh 29. Sistem yang urutannya diubah tidak dominan diagonal dan Jacobi divergen: pergeseran* 1.073×10^{12} *pada iterasi 16. Buku: Subbab 5.6.*

Menjalankannya: 400 Peubah

```
Griddle with 20 x 20 inner points (400 unknowns)
stop when no point moves by 1e-06 or more in a sweep
method           omega  sweeps  max error
Jacobi           1.000    1302   8.76e-05
Gauss-Seidel     1.000     683   4.31e-05
SOR              1.500     235   1.36e-05
SOR (optimal)    1.741      80   8.85e-07
```



Gambar 5.2 Galat terbesar wajan 20×20 setelah setiap sapuan, terhadap solusi acuan. Garis lurus pada skala ini berarti faktor tetap per sapuan: $\rho_J = 0.989$, $\rho_{GS} = 0.978$, dan sekitar 0.74 untuk SOR dengan $\omega = 1.741$.

`griddle.py` *tidak pernah membangun matriksnya: ia menyapu kisi yang berbingkai. Gambar 5.2: garis lurus pada skala log, dengan kemiringan yang ditentukan* ρ*. SOR optimal butuh 80 sapuan, sedangkan Jacobi butuh 1.302. Buku: Subbab 5.6.*

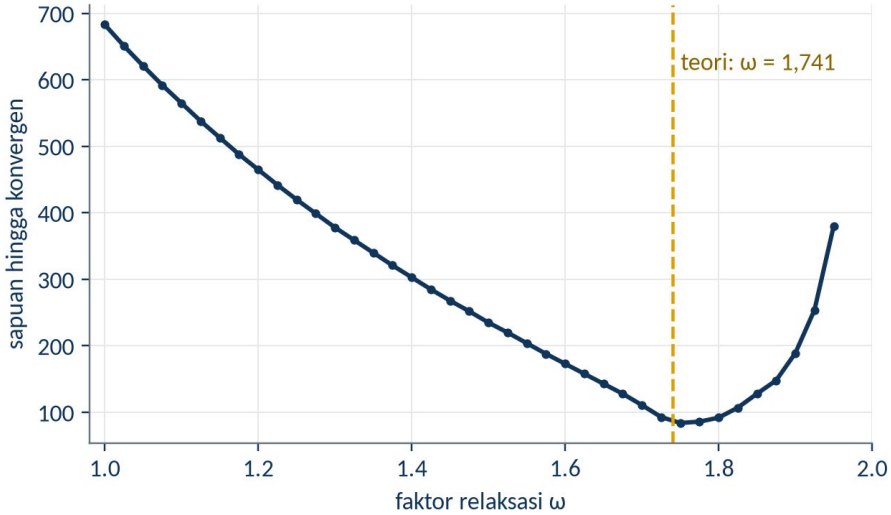
Banyak Sapuan terhadap Ukuran Kisi

Tabel 5.1 Banyak sapuan yang diperlukan agar pergeseran terbesar pada wajan $n \times n$ turun di bawah 10^{-6}. Gauss-Seidel memerlukan sekitar separuh sapuan Jacobi, sesuai ramalan $\rho_{GS} = \rho_J^2$; SOR optimal hanya tumbuh linear terhadap n.

n	Peubah	ρ_J	Jacobi	Gauss-Seidel	SOR	ω_{opt}
10	100	0.95949	385	202	42	1.560
20	400	0.98883	1302	683	80	1.741
40	1600	0.99707	4518	2377	155	1.858

Tabel 5.1: menggandakan n kira-kira mengempatkalikan sapuan Jacobi dan Gauss-Seidel tetapi hanya menggandakan sapuan SOR. Pada $n = 40$ itu berarti 4.518 lawan 155. Buku: Subbab 5.6.

Memilih Omega

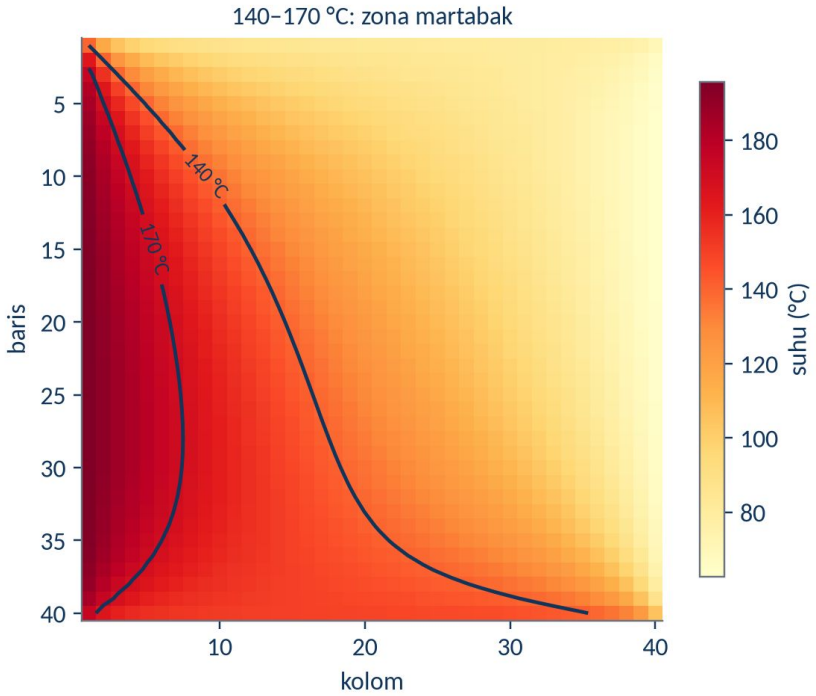


Gambar 5.3 Banyak sapuan yang diperlukan SOR pada wajan 20×20 untuk faktor relaksasi dari 1 sampai 1.95. Titik terendahnya berada pada nilai Young $\omega_{\text{opt}} = 1.741$.

WAWASAN KUNCI: Kurvanya timpang: sedikit di atas optimum itu murah (84 sapuan pada 1,75, 92 pada 1,80), jauh di bawahnya membuang sebagian besar manfaatnya (683 pada 1,00). Bila ragu, condonglah ke sisi atas — tetapi tetap di bawah 2.

Gambar 5.3: pemindaian ω dari 1 sampai 1,95 dengan langkah 0,05 menemukan lembahnya tepat di tempat yang ditunjuk Persamaan (5.12). Optimumnya bergantung pada n: 1,560 untuk $n=10$, 1,858 untuk $n=40$. Buku: Subbab 5.6.

Peta Suhu



Gambar 5.4 Suhu tunak wajan pada kisi 40×40 (kompor di kiri, tepi depan yang hangat di bawah), dengan garis 140 °C dan 170 °C yang membatasi zona martabak.

Gambar 5.4: jawaban bagi pertanyaan Pak Joni. Zona martabak adalah pita selebar delapan sampai sepuluh sentimeter yang dimulai beberapa sentimeter dari kompor dan melebar ke arah tepi depan yang hangat. Pada kisi 20×20*, 94 dari 400 titik, hampir seperempat pelat, berada di dalamnya: martabak telur di pita itu, roti bakar tepat di sebelah kanannya, dan pojok kanan belakang untuk menjaga makanan tetap hangat. Buku: Subbab 5.6.*

Pergeseran Kecil Bukan Galat Kecil

$$\| x^{(k)} - x \| \leq \frac{\rho}{1-\rho} \| x^{(k)} - x^{(k-1)} \| \tag{5.14}$$

PERINGATAN: Untuk Jacobi pada wajan 20×20, $\rho/(1-\rho)$ sekitar 88: ketika pergeseran terakhir 10^{-6}, galatnya masih dapat mendekati 10^{-4}. Galat 8.76×10^{-5} di tabel membenarkannya.

Persamaan (5.14) adalah batas titik tetap dari Bab 3 dalam bentuk matriks. Perketat toleransinya dengan faktor $(1-\rho)/\rho$, atau berhentilah berdasarkan residu disertai taksiran κ. Buku: Subbab 5.4.5.

Memeriksa dengan Pustaka

```
NumPy 2.5.3, SciPy 1.18.1
4 x 4 system, numpy.linalg.solve: [131.25  96.25 148.75 113.75]
griddle matrix: 400 x 400, 1920 nonzeros of 160000
SOR (126 sweeps) vs spsolve: max difference 1.4e-12
conjugate gradient: 64 iterations, max difference 6.9e-09
```

`compare_scipy.py` membangun matriks wajan secara eksplisit di `scipy.sparse`, menyelesaikannya dengan pemecah langsung jarang `spsolve` dan dengan metode gradien konjugat, lalu membandingkan keduanya dengan SOR kita pada toleransi yang lebih ketat. Gradien konjugat, kuda beban modern untuk sistem simetris definit positif, hanya butuh 64 iterasi. Buku: Subbab 5.7–5.8.

Metode Numerik dalam Praktik: Profitina

AI Visibility Engine milik Profitina melihat sebuah situs halaman demi halaman. Halaman-halaman saling bertaut, dan nilai yang ditentukan oleh nilai halaman tetangganya adalah **sistem linear jarang** yang besar, satu peubah per halaman.

Awal hangat. Merek dipindai ulang secara otomatis dengan irama sesuai tingkat langganannya, lengkap dengan tren skor mingguan. Dari satu pemindaian ke berikutnya hanya sedikit halaman yang berubah, sehingga nilai minggu lalu adalah tebakan awal yang sangat baik.

Berhenti dengan jujur. “Angkanya sudah berhenti bergerak” dapat menyembunyikan galat sembilan puluh kali lebih besar daripada pergeseran terakhir.

Bab 12 bertemu contoh paling terkenal, PageRank, iterasi jenis Jacobi pada graf tautan web. Dasbor yang ditunjukkan kepada pemilik merek harus berhenti dengan kriteria yang membatasi galat sebenarnya. Aplikasi memberi saran dan pemilik yang memutuskan. Buku: Subbab 5.10.

Kesalahan Umum

Beriterasi pada sistem yang urutannya buruk. Periksa Persamaan (5.10), atau paling tidak taruh koefisien terbesar setiap persamaan di diagonal.

Menganggap pergeseran kecil sebagai galat kecil. Pakai Persamaan (5.14).

Mencampur Jacobi dan Gauss-Seidel tanpa sengaja. Di Python, `new = x` alih-alih `new = x[:]` melakukannya.

Memilih ω di luar $(0, 2)$. SOR lalu divergen untuk setiap matriks.

Memakai ω optimal suatu kisi untuk kisi lain. Hitung ulang dengan Persamaan (5.12).

Untuk memetakan wajan, dinding oven, atau lantai jemur Anda sendiri, ubah empat hal di `griddle.py`: ukur tepi-tepinya dengan termometer inframerah, pilih kisinya (satu titik per sentimeter sudah cukup), tangani tepi yang terisolasi (tidak ada panas yang melintasinya), dan periksa modelnya terhadap tiga atau empat titik dalam yang diukur. Peta hasil hitungan menunjukkan bagian yang lebih panas atau lebih dingin, bukan apakah sebuah alat aman. Buku: Subbab 5.9 dan 5.11.

Ringkasan

1. Sistem jarang yang besar muncul secara alami: setiap titik wajan adalah rata-rata tetangganya (Persamaan (5.1)).

2. Jacobi (Persamaan (5.3)), Gauss-Seidel (Persamaan (5.4)), dan SOR (Persamaan (5.5)) memperbaiki tebakan sapuan demi sapuan.

3. Semuanya iterasi stasioner (Persamaan (5.7)); konvergen tepat bila $\rho(T)<1$; dominansi diagonal adalah uji yang cukup.

4. Pada wajan, banyak sapuan tumbuh seperti n^2 untuk Jacobi dan Gauss-Seidel, seperti n untuk SOR optimal: 4.518, 2.377, dan 155 pada $n=40$.

5. Berhentilah dengan mengingat Persamaan (5.14); `spsolve` dan gradien konjugat SciPy membenarkan petanya.

Metode iteratif menukar ketepatan dalam jumlah langkah tetap dengan memori kecil dan biaya kecil per langkah. Untuk matriks jarang, itu pertukaran yang sangat baik. Buku: Subbab 5.12.

Pertemuan Berikutnya

Bab 6 — Dekomposisi Matriks: LU, Cholesky, dan QR. Faktorkan matriks sekali, lalu selesaikan dengannya berkali-kali.

Eliminasi ditinjau ulang sebagai faktorisasi, versi yang lebih cepat untuk matriks simetris definit positif, dan versi ortogonal yang tidak pernah kehilangan ketelitian.

Sebelumnya: lanjutkan tabel Jacobi untuk sapuan 7 dan 8 dengan tangan, dan ukur seberapa besar pergeserannya menyusut.

Soal latihan itu menanyakan seberapa jauh T_1 dari 131,25 setelah sapuan 8; bandingkan jawaban Anda dengan batas Persamaan (5.14). Penyelesaiannya ada di buku. Buku: Subbab 5.13–5.14.

Kuliah 6 — Dekomposisi Matriks: LU, Cholesky, dan QR

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 6

Dosen: Irfan Subakti — yifana@gmail.com

Eliminasi Gauss membuang pengali-pengalinya. Simpan pengali itu, dan eliminasi menjadi faktorisasi matriks yang dapat dipakai ulang untuk setiap ruas kanan baru. Hari ini: LU untuk sistem persegi, Cholesky untuk sistem simetris definit positif dengan separuh biaya, dan QR untuk persamaan yang lebih banyak daripada peubah — kuadrat terkecil tanpa kehilangan digit. Buku: Bab 6, Subbab 6.1–6.13.

Cakupan Pertemuan Ini

1. Masalahnya — empat blok pohon durian, satu set karung pupuk, empat rekomendasi.

2. LU — $PA = LU$ dengan tangan, rumus Doolittle, dan biaya faktorkan-sekali-selesaikan-berkali-kali.

3. Kuadrat terkecil — unsur hara keempat, persamaan normal, dan Cholesky.

4. QR — mengapa persamaan normal mengkuadratkan bilangan kondisi, dan refleksi Householder.

5. Menjalankan dan memeriksa — dua program dalam Java dan Python, dan faktorisasi SciPy.

LU, Cholesky, dan QR adalah apa yang sungguh dikerjakan pustaka ketika Anda memanggil `solve` atau `lstsq`. Mengenalnya memberi tahu panggilan mana yang tepat dan kapan harus khawatir. Buku: Bab 6.

Pertanyaan Pupuk Pak Harun

Kebun buah dan durian dalam **empat blok**: pohon muda, sesudah panen, menjelang berbunga, dan pengisian buah.

Dua kali setahun penyuluh pertanian menuliskan gram **N, P_2O_5, dan K_2O** per pohon untuk setiap blok. Pohon muda musim ini: 150, 60, dan 70 g.

Toko tani menjual tiga karung: **NPK 12-12-17**, **NPK 20-10-10**, dan **urea 46-0-0**.

Empat sistem musim ini, empat lagi musim depan — tetapi **matriks kadarnya tetap sama bertahun-tahun**; hanya ruas kanannya yang berubah.

Bab 4 dapat menyelesaikan setiap sistem dari awal. Itu mengulang bagian yang mahal setiap kali. Bab ini mengerjakan bagian yang mahal hanya sekali. Pola yang sama muncul di mana-mana: satu jaringan pipa dengan banyak pola pemakaian, satu rangka bangunan dengan banyak kombinasi beban, satu model skor dengan banyak merek. Matriksnya tetap, ruas kanannya berganti. Buku: Subbab 6.1.

Persamaannya

Tabel 6.1 Gram setiap hara dalam 100 g setiap pupuk, dan rekomendasi untuk satu pohon muda.

Unsur hara	NPK 12-12-17	NPK 20-10-10	Urea 46-0-0	Kebutuhan pohon muda
N	12	20	46	150 g
P_2O_5	12	10	0	60 g
K_2O	17	10	0	70 g

$$\begin{pmatrix} 12 & 20 & 46 \\ 12 & 10 & 0 \\ 17 & 10 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 150 \\ 60 \\ 70 \end{pmatrix} \tag{6.1}$$

Tabel 6.1: ukur pupuk dalam satuan 100 g; setiap kolom satu karung, setiap baris satu unsur hara. Persamaan (6.1) adalah sistem untuk pohon muda. Buku: Subbab 6.2.1.

Simpan Pengalinya: $PA = LU$

Pivoting menaikkan baris kalium, dengan 17, ke atas; kedua baris lain mendapat pengali $12/17 = 0.7059$. Di kolom 2 baris nitrogen menang (12,9412 lawan 2,9412), dan pengali terakhir $5/22 = 0.2273$.

L = pengali-pengali di bawah diagonal satuan; U = segitiga atas yang ditinggalkan eliminasi, dengan diagonal 17, 12.9412 , − 10.4545.

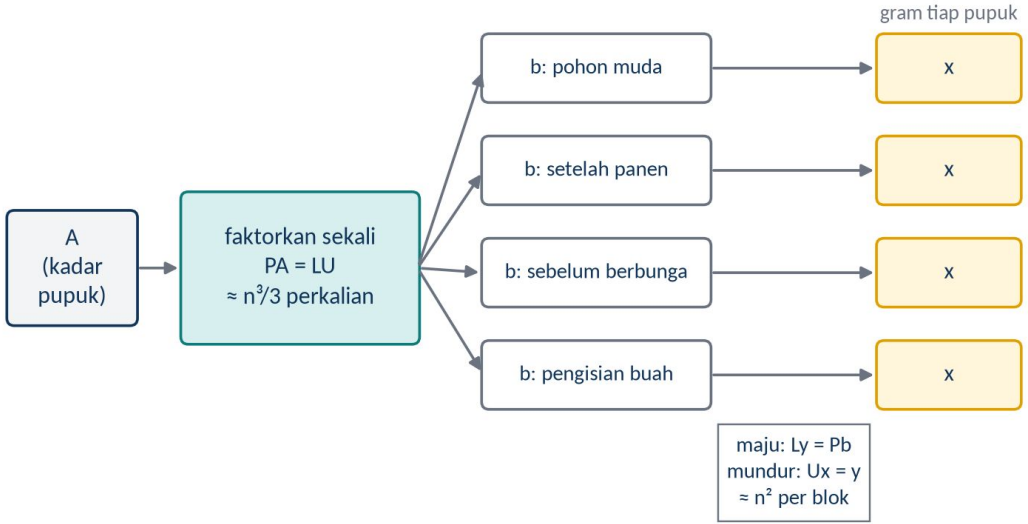
$$PA = LU \tag{6.2}$$

Lalu setiap ruas kanan hanya butuh dua penyelesaian segitiga:

$$Ly = Pb \text{ \(forward substitution\)}, Ux = y \text{ \(back substitution\)} \tag{6.3}$$

Persamaan (6.2): P mencatat pertukaran baris. Persamaan (6.3): substitusi maju memutar ulang eliminasi pada b, substitusi mundur sama dengan Bab 4. Untuk pohon muda hasilnya $x = (2, 3.6, 1.174)$: 200 g, 360 g, dan 117,4 g. Periksa satu baris dengan mengalikan: $0.7059 \times (17, 10, 0) + (0, 12.9412, 46) = (12, 20, 46)$, baris nitrogen. Tidak ada yang baru dihitung; L dan U hanyalah kerja eliminasi yang dicatat alih-alih dibuang. Buku: Subbab 6.2.2–6.2.3.

Faktorkan Sekali, Selesaikan Berkali-kali



Gambar 6.1 Faktorkan sekali, selesaikan berkali-kali: matriks kadar difaktorkan sebagai $PA = LU$ sekali, dan rekomendasi setiap blok hanya memerlukan satu substitusi maju dan satu substitusi mundur.

Gambar 6.1: faktorisasinya hanya bergantung pada karung-karungnya; keempat blok, dan blok-blok musim depan, memakainya ulang. Buku: Subbab 6.2.

Rumus Doolittle dan Algoritmanya

$$u_{kj} = a_{kj} - \sum_{m<k} l_{km} u_{mj} \,(j \geq k), \quad l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{m<k} l_{im} u_{mk} \right) (i > k) \tag{6.4}$$

Pseudokode 6.1 Dekomposisi LU dengan *pivoting* parsial, disimpan di tempat, dan dua penyelesaian segitiga.

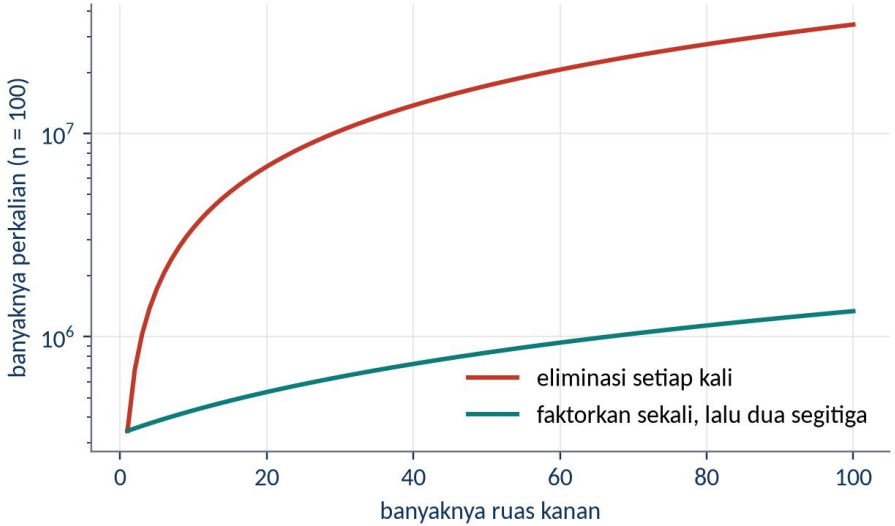
```
LU-DECOMPOSE(A)                    // A is n x n, overwritten
 1  perm = (1, 2, ..., n); sign = 1
 2  for k = 1 to n - 1
 3      p = the row i >= k with the largest |A[i][k]|
 4      if A[p][k] == 0: error "singular"
 5      if p != k: swap rows k and p of A and of perm; sign = -sign
 6      for i = k + 1 to n
 7          A[i][k] = A[i][k] / A[k][k]            // multiplier l_ik
 8          for j = k + 1 to n
 9              A[i][j] = A[i][j] - A[i][k] * A[k][j]
10  return A, perm, sign

LU-SOLVE(A, perm, b)
11  for i = 1 to n: y[i] = b[perm[i]] - sum_{j<i} A[i][j] * y[j]
12  for i = n downto 1: x[i] = (y[i] - sum_{j>i} A[i][j] * x[j]) / A[i][i]
13  return x
```

Persamaan (6.4) muncul dari membaca $A = LU$ entri demi entri: baris ke-i dari A adalah kombinasi baris-baris U dengan bobot dari baris ke-i matriks L. Pseudokode 6.1 menyimpan L di bawah diagonal dan U pada dan di atasnya, dalam larik yang sama — tanpa memori tambahan. Buku: Subbab 6.3.1–6.3.2.

Biayanya

$$\sum_{k=1}^{n-1} (n-k)^2 \approx \frac{n^3}{3}, \text{while each solve costs} \approx n^2 \tag{6.5}$$



Gambar 6.2 Banyak perkalian untuk K ruas kanan dengan matriks 100×100: eliminasi setiap kali tumbuh seperti $K\,n^3/3$, faktorisasi sekali seperti $n^3/3 + K\,n^2$.

Persamaan (6.5) menghitung perkalian. Gambar 6.2: untuk 100 ruas kanan dan $n=1000$, eliminasi setiap kali berbiaya 33 miliar perkalian, faktorisasi sekali 0,43 miliar. Determinan didapat cuma-cuma sebagai hasil kali bertanda diagonal U, dan invers adalah n penyelesaian, jarang diperlukan. Buku: Subbab 6.3.3–6.3.4.

Determinan

$$\det A = (\pm 1)\, u_{11} u_{22} \cdots u_{nn} \tag{6.6}$$

Untuk matriks kadar: dua pertukaran baris (17 ke atas pada langkah 1, lalu 12,9412 di atas 2,9412 pada langkah 2), jadi tandanya $+1$ dan $\det A = 17 \times 12.9412 \times (-10.4545) = -2300$.

> **PERINGATAN:** Pivoting juga tidak boleh dilewatkan pada LU. Tanpanya pivot yang sangat kecil menghasilkan pengali raksasa yang menghapus informasi di baris-baris lain — kegagalan yang sama dengan Bab 4, kini tersimpan di dalam L.

Persamaan (6.6): tandanya (-1) pangkat banyaknya pertukaran, yang dicatat Pseudokode 6.1 dalam `sign`*. Buku: Subbab 6.3.4–6.3.5.*

Unsur Hara Keempat: Persamaan Lebih Banyak daripada Peubah

Daun menguning: penyuluh menambahkan **6 g MgO** per pohon muda. NPK 12-12-17 dijual sebagai 12-12-17-2 (2% MgO); pupuk lain tidak mengandungnya.

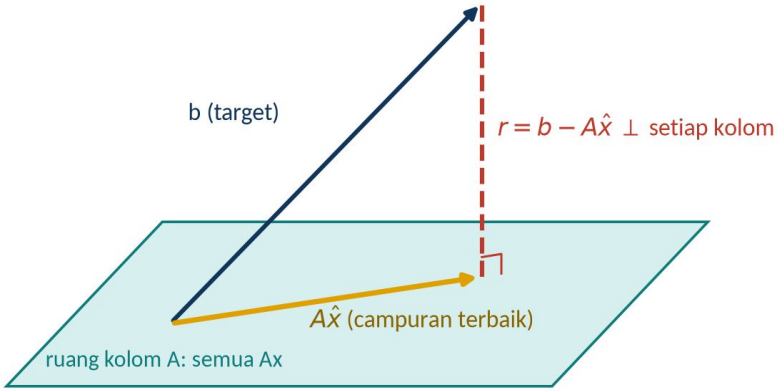
$$\begin{pmatrix} 12 & 20 & 46 \\ 12 & 10 & 0 \\ 17 & 10 & 0 \\ 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \approx \begin{pmatrix} 150 \\ 60 \\ 70 \\ 6 \end{pmatrix} \tag{6.7}$$

Tiga baris pertama memaksa $x_1 = 2$, yang memberi 4 g MgO, bukan 6. Tidak ada campuran yang memenuhi keempatnya. Mana yang **paling dekat**?

$$\min_x \| b - A\,x \|_2^2 = \min_x \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)^2 \tag{6.8}$$

Persamaan (6.7) ***tertentu lebih*** *(overdetermined). Persamaan (6.8) mengukur kedekatan dengan jumlah kuadrat meleset —* ***masalah kuadrat terkecil****, yang juga menjadi jantung Bab 8. Buku: Subbab 6.4.1.*

Persamaan Normal



Gambar 6.3 Campuran kuadrat terkecil: $A\,\hat{x}$ adalah titik ruang kolom yang paling dekat ke target b, dan melesetnya $r = b - A\,\hat{x}$ tegak lurus terhadap setiap kolom.

$$A^T A\,\hat{x} = A^T b \tag{6.9}$$

Gambar 6.3: setiap campuran $A\,x$ terletak pada bidang yang direntang kolom-kolomnya; titik terdekat ke b adalah kaki garis tegak lurus. "Tegak lurus terhadap setiap kolom" berarti $A^T(b - A\,\hat{x}) = 0$, yaitu Persamaan (6.9): sistem persegi 3×3. Buku: Subbab 6.4.2.

Cholesky: Separuh Pekerjaan

$S = A^T A$ **simetris definit positif** (SPD). Ia terfaktor sebagai $S = L L^T$:

$$l_{jj} = \sqrt{s_{jj} - \sum_{k<j} l_{jk}^2}, \quad l_{ij} = \frac{1}{l_{jj}} \left(s_{ij} - \sum_{k<j} l_{ik} l_{jk} \right) \quad (i > j) \tag{6.10}$$

Untuk kebun, $l_{11} = \sqrt{581} = 24.1039$, $l_{21} = 530 / 24.1039 = 21.9881$, dan seterusnya.

Persamaan (6.10) berbiaya $n^3/6$ perkalian, separuh LU, dan tidak butuh pivoting pada matriks SPD karena setiap entrinya dibatasi akar kuadrat diagonal. Bilangan tak positif di bawah tanda akar membuktikan matriksnya bukan SPD — Cholesky juga uji SPD yang termurah. Setelah L didapat, campurannya menyusul dari dua substitusi, $L\,y = A^T b$ lalu $L^T \hat{x} = y$, persis seperti pada LU. Buku: Subbab 6.4.3–6.4.4.

QR: Kuadrat Terkecil tanpa Mengkuadratkan

Persamaan normal punya biaya tersembunyi:

$$\kappa(A^T A) = \kappa(A)^2 \tag{6.11}$$

Dengan $A = Q\,R$ dan Q ortogonal, panjang tidak berubah, dan masalahnya terbelah:

$$\|b - A\,x\|^2 = \|Q^T b - R\,x\|^2 = \|c_1 - R_1\,x\|^2 + \|c_2\|^2 \tag{6.12}$$

Persamaan (6.11): jika A kehilangan 6 digit, $A^T A$ kehilangan 12. Persamaan (6.12): pilih x yang membuat bagian pertama nol, satu penyelesaian segitiga $R_1 x = c_1$; bagian kedua adalah meleset yang tak terhindarkan. Q tidak pernah perlu dibentuk secara eksplisit. Buku: Subbab 6.5.1–6.5.2.

Refleksi Householder

$$H = I - \frac{2\,v\,v^T}{v^T v} \tag{6.13}$$

H mencerminkan setiap vektor terhadap cermin yang tegak lurus v. Memilih v dengan tepat mengirim satu kolom utuh menjadi kelipatan vektor satuan pertama: semua nol di bawah diagonal sekaligus.

Tabel 6.2 Ketiga dekomposisi bab ini, apa yang diperlukannya, berapa biayanya, dan untuk apa.

Dekomposisi	Untuk matriks apa	Biaya (perkalian)	Dipakai untuk
$PA = LU$	sembarang persegi	$n^3/3$	sistem persegi, banyak ruas kanan
$S = L\,L^T$	simetris definit positif	$n^3/6$	sistem SPD, persamaan normal
$A = Q\,R$	sembarang $m \times n$, $m \geq n$	$mn^2 - n^3/3$	kuadrat terkecil tanpa mengkuadratkan κ

Persamaan (6.13): refleksi bersifat ortogonal, sehingga tidak pernah memperbesar galat. Tabel 6.2 adalah seluruh bab dalam satu kartu. Buku: Subbab 6.5.3–6.5.4.

Menjalankannya: LU untuk Empat Blok

```
max |PA - LU| = 0.0e+00
det A = -2300.0000

grams per tree           NPK 12-12-17  NPK 20-10-10   Urea 46-0-0
young trees                     200.0         360.0         117.4
after harvest                   400.0         420.0         104.3
before flowering                600.0         180.0          26.1
fruit filling                   600.0          80.0          26.1
```

`lu_blend.py` dan `LuBlend.java` mencetak P, L, dan U lebih dulu — tepat faktor-faktor hitungan tangan — lalu baris-baris ini. "Menjelang berbunga" dan "pengisian buah" berbeda $(20, 10, 10)$, tepat satu satuan NPK 20-10-10, sehingga campurannya berbeda 100 g karung itu. Sistem linear menghadiahi cara membaca seperti ini. Buku: Subbab 6.6–6.7.

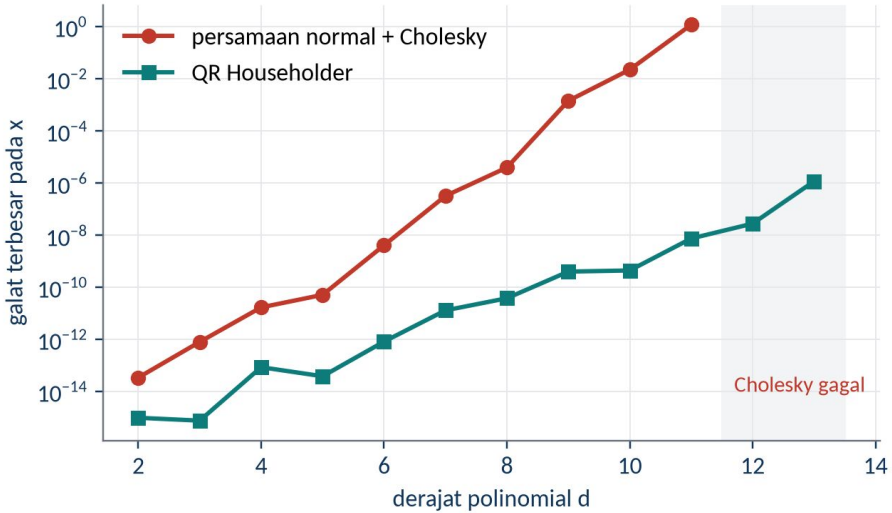
Menjalankannya: Campuran Kuadrat Terkecil

```
grams per tree       normal+Cholesky  Householder QR
  NPK 12-12-17-2            224.242         224.242
  NPK 20-10-10              324.848         324.848
  Urea                      126.350         126.350
nutrient  target  delivered  miss
  N        150.0     150.00  +0.00
  P2O5      60.0      59.39  -0.61
  K2O       70.0      70.61  +0.61
  MgO        6.0       4.48  -1.52
residual norm 1.7408
```

Kedua metode sepakat di sini. Nitrogen dipenuhi tepat, karena urea hanya berisi nitrogen dan dapat membereskan baris itu sendirian; meleset yang tak terhindarkan disebar ke fosfat, kalium, dan magnesium, di tempat biayanya paling kecil dalam kuadrat gram. Apakah 1,5 g MgO lebih penting daripada 0,6 g fosfat adalah pertanyaan agronomi; memberi bobot pada baris-baris memberi tahu programnya. Buku: Subbab 6.7.

Ketika Persamaan Normal Gagal

```
Polynomial columns t^0..t^d at 20 points, true x = all ones
 d   normal+Cholesky error   Householder QR error
 3             8.14e-13              7.77e-16
 6             4.14e-09              8.35e-13
 9             1.45e-03              4.05e-10
12    fails (pivot <= 0)              2.81e-08
```



Gambar 6.4 Galat terbesar solusi kuadrat terkecil untuk kolom polinomial derajat d pada 20 titik: persamaan normal kehilangan sekitar dua kali lebih banyak digit daripada QR Householder, dan mulai derajat 12 Cholesky gagal.

Gambar 6.4: Persamaan (6.11) yang terlihat. Pada derajat 9 QR mempertahankan sepuluh digit sedangkan persamaan normal tiga; pada derajat 12 pembulatan membuat $A^T A$ tampak tidak definit positif dan Cholesky berhenti, sementara QR masih memberi delapan digit. Buku: Subbab 6.7.

Memeriksa dengan Pustaka

```
NumPy 2.5.3, SciPy 1.18.1
scipy.linalg.lu: U diagonal [ 17.      12.9412 -10.4545]
lu_solve vs ours, all four blocks: max difference 6.2e-15
numpy.linalg.lstsq grams: [224.242 324.848 126.35 ]  residual norm 1.7408
cho_solve grams:          [224.242 324.848 126.35 ]
numpy.linalg.qr R diagonal: [-24.1039  10.7946 -10.1559]
degree 9 columns: cond(A) = 3.79e+06, cond(A^T A) = 1.44e+13
```

SciPy mereproduksi setiap angka. Perhatikan konvensinya: `scipy.linalg.lu` *mengembalikan* $A = PLU$*, transpos dari* P *kita.* R *dari NumPy punya diagonal yang sama dengan faktor Cholesky kita sampai tanda — dalam aritmetika eksak* R *dan* L^T *adalah matriks yang sama. Baris terakhir adalah Persamaan (6.11) yang diukur:* 3.79×10^6 *dikuadratkan menjadi* 1.44×10^{13}*. Buku: Subbab 6.8.*

Metode Numerik dalam Praktik: Profitina

AI Visibility Engine milik Profitina mengubah sinyal-sinyal yang diukurnya menjadi skor kesiapan dan menjelaskan setiap sinyal. Memilih bobot agar skor sesuai dengan yang teramati, dalam bentuknya, adalah **masalah kuadrat terkecil tertentu lebih**: satu baris per situs yang diamati, satu kolom per sinyal.

Sinyal-sinyal yang mengukur hal yang hampir sama membuat $\kappa(A)$ besar dan $\kappa(A^T A)$ raksasa; persamaan normal lalu dapat memberi bobot yang hanya berisi derau pembulatan. **QR adalah jalan yang aman**, dan pustaka memakainya secara bawaan.

Bila satu model dievaluasi untuk banyak merek pada setiap pemindaian ulang terjadwal, faktorisasi yang mahal berada di luar perulangan dan hanya substitusi yang murah di dalamnya, seperti pada Gambar 6.1. Buku: Subbab 6.11.

Membuatnya Lebih Baik, dan Kesalahan Umum

Gram negatif. LU dan QR tidak tahu bahwa pupuk tidak mungkin negatif; rekomendasi $(150, 100, 120)$ memberi -4.35 g urea. Pakai kuadrat terkecil nonnegatif, `scipy.optimize.nnls`.

Menghitung invers untuk menyelesaikan sistem. Tiga kali biaya `solve`, dan kurang teliti.

Memfaktorkan ulang untuk setiap ruas kanan. Faktorkan sekali, di luar perulangan.

Persamaan normal pada kolom yang hampir bergantung. Mereka mengkuadratkan κ; pakai QR.

Untuk memakai `lu_blend.py` bagi kebun, sawah, atau kebun sayur Anda sendiri, ubah kadar karung-karung Anda, rekomendasinya, dan nama-namanya. Buku: Subbab 6.9, 6.10, dan 6.12.

Ringkasan

1. Eliminasi dengan pengali yang dicatat adalah $PA = LU$ (Persamaan (6.2), Pseudokode 6.1), dari rumus Doolittle (Persamaan (6.4)).
2. Faktorisasi berbiaya $n^3/3$; setiap ruas kanan lalu berbiaya sekitar n^2 (Persamaan (6.3), Persamaan (6.5)).
3. Syarat yang lebih banyak daripada peubah membawa ke kuadrat terkecil (Persamaan (6.8)); residu terbaik tegak lurus kolom-kolomnya, memberi persamaan normal (Persamaan (6.9)).
4. Cholesky (Persamaan (6.10)) menyelesaikan sistem SPD dengan separuh biaya LU.
5. Persamaan normal mengkuadratkan κ (Persamaan (6.11)); QR Householder (Persamaan (6.13), Persamaan (6.12)) tidak.

Program Java dan Python sama byte demi byte, dan SciPy serta NumPy mereproduksi setiap angka. Buku: Subbab 6.13.

Pertemuan Berikutnya

Bab 7 — Interpolasi: Lagrange, Newton, dan Spline. Kurva melalui titik-titik ukur, dan nilai-nilai di antaranya.

Satu polinom melalui semua titik, bahayanya di ujung-ujung, dan alternatif sepotong-sepotong yang mulus.

Sebelumnya: selesaikan blok “menjelang berbunga” dengan tangan memakai faktor P, L, dan U, dengan $b = (120, 90, 120)$.

Periksa jawaban tangan Anda dengan 600, 180, dan 26,1 g dari program. Penyelesaian semua soal latihan, termasuk invers matriks kadar, ada di buku. Buku: Subbab 6.14–6.15.

Kuliah 7 — Interpolasi: Lagrange, Newton, dan Spline

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 7

Dosen: Irfan Subakti — yifana@gmail.com

Sebuah tabel memberi nilai di beberapa titik; kehidupan menanyakan nilai di antaranya. Hari ini kita membangun fungsi melalui titik-titik tabel — garis lurus, satu polinomial melalui semuanya, dan rantai kubik yang mulus — lalu mengukur dengan jujur mana yang paling baik membaca tangki bahan bakar. Buku: Bab 7, Subbab 7.1–7.12.

Cakupan Pertemuan Ini

1. Masalahnya — tongkat ukur kios bahan bakar dan bagan dinding bertanda setiap 20 cm.

2. Interpolasi polinomial — bentuk Lagrange dan Newton, selisih terbagi, evaluasi Horner.

3. Galat dan jebakannya — rumus galat, fungsi Runge, titik Chebyshev, Hermite.

4. Spline kubik — kelengkungan dari sistem tridiagonal, algoritma Thomas, syarat ujung.

5. Mengukurnya — setiap metode terhadap rumus eksak tangki, dan terhadap NumPy dan SciPy.

Tangki ini guru yang sempurna karena, tidak seperti kebanyakan tabel, fungsi eksaknya diketahui: kita dapat mengukur setiap galat alih-alih menebaknya. Buku: Bab 7.

Tongkat Ukur Pak Darto

Sebuah **kios bahan bakar** kecil: tangki baja yang rebah, panjang 200 cm dan garis tengah 120 cm, dibaca dengan tongkat kayu.

Setiap malam kedalamannya diubah menjadi liter dan dibandingkan dengan penjualan menurut pompa. Selisih berarti kebocoran, pompa rusak, atau penjualan yang lupa dicatat.

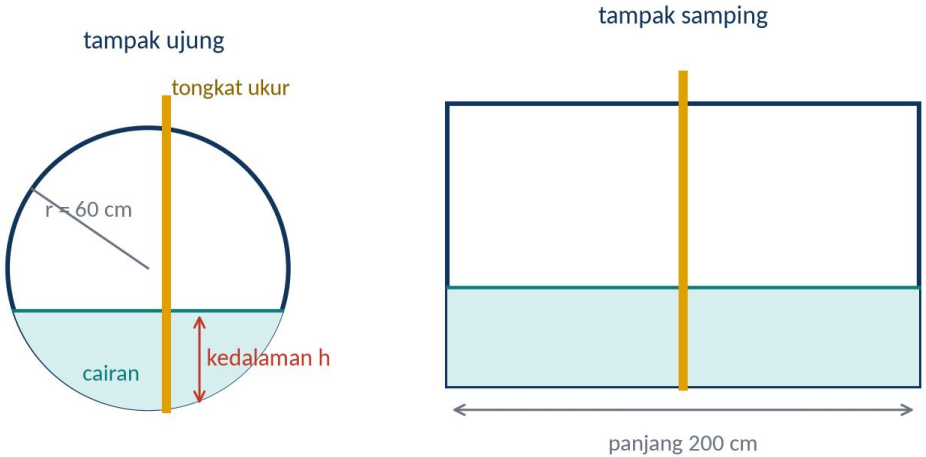
Bagan dinding menunjukkan setiap **20 cm**; tongkatnya menunjukkan 47 cm, atau 13 cm. Bertahun-tahun Pak Darto menarik garis lurus di kepalanya antara dua tanda terdekat.

Di dekat dasar dan puncak, bacaan garis lurus itu meleset **lebih dari 30 liter**.

Tangki rebah bukan ember: 10 cm pertama hanya menampung sedikit karena dasarnya sempit, sedangkan 10 cm di tengah menampung banyak. Keponakannya menduga sebagian liter yang “hilang” berasal dari cara membaca, bukan dari tangkinya. Buku: Subbab 7.1.

Tangki Eksak, Tongkat Ukur Kita

$$V(h) = L\left[r^2 \arccos\left(\frac{r-h}{r}\right) - (r-h)\sqrt{2rh-h^2}\right] \tag{7.1}$$



Gambar 7.1 Tangki rebah di kios bahan bakar: tongkat ukur mengukur kedalaman h, dan volumenya adalah luas tembereng lingkaran dikalikan panjang tangki.

Persamaan (7.1) adalah rumus yang sama dengan truk air Bab 3. Di sini kita berpura-pura hanya mengetahui tabelnya, membangun interpolan darinya, dan memakai Persamaan (7.1) untuk mengukur seberapa salah masing-masing. Gambar 7.1: tembereng lingkaran di bawah permukaan. Buku: Subbab 7.1.

Gagasannya: Garis Lurus di antara Tanda

$$p(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) \tag{7.2}$$

Galatnya paling besar $\frac{h^2}{8}$ kali kelengkungan terbesar f pada potongan itu.

Di tengah tangki kurvanya hampir lurus: galat sekitar 3,4 L. Di dekat ujung ia melengkung tajam: sekitar **34 L**.

Persamaan (7.2) adalah yang dikerjakan Pak Darto di kepalanya. Batas galatnya memberi tahu di mana harus khawatir: di tempat fungsinya melengkung. Ada dua jalan keluar: pakai lebih banyak informasi per potongan (kurva alih-alih garis), atau lebih banyak tanda. Buku: Subbab 7.2.

Satu Polinomial melalui Semua Titik: Lagrange

Melalui $n+1$ titik dengan x_i berbeda ada tepat satu polinomial berderajat paling tinggi n. Lagrange menuliskannya dengan polinomial basis yang bernilai 1 di satu simpul dan 0 di simpul lain:

$$\ell_i(x)=\prod_{j\neq i}\frac{x-x_j}{x_i-x_j},\quad p(x)=\sum_{i=0}^{n} y_i\,\ell_i(x) \tag{7.3}$$

Persamaan (7.3) mudah ditulis dan mudah dibuktikan benar: di simpul x_k setiap suku lenyap kecuali $y_k\cdot 1$. Ia canggung untuk diperluas — satu titik baru mengubah setiap polinomial basis. Buku: Subbab 7.3.1.

Bentuk Newton dan Selisih Terbagi

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n] \prod_{j=0}^{n-1} (x - x_j) \tag{7.4}$$

$$f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i} \tag{7.5}$$

Persamaan (7.4) adalah polinomial yang sama yang ditulis sehingga satu titik tambahan menambah satu suku. Koefisiennya adalah ***selisih terbagi*** *Persamaan (7.5), dihitung kolom demi kolom seperti tabel kemiringan dari kemiringan. Buku: Subbab 7.3.2.*

Tabel Selisih Terbagi dengan Tangan

Tabel 7.1 Tabel selisih terbagi untuk empat tanda pertama; entri teratas setiap kolom adalah koefisien bentuk Newton.

x_i	$f[x_i]$	Pertama	Kedua	Ketiga
0	0,00	12,39	0,205525	−0,00220188
20	247,80	20,611	0,0734125	
40	660,02	23,5475		
60	1130,97			

Evaluasi secara bersarang — **skema Horner**, n perkalian:

$$p(x) = c_0 + (x - x_0)\big(c_1 + (x - x_1)\big(c_2 + \cdots + (x - x_{n-1})\,c_n\big)\big) \tag{7.6}$$

Tabel 7.1: misalnya $247.80/20 = 12.39$ *dan* $(20.611 - 12.39)/40 = 0.205525$. *Kubiknya* $p_3(x) = 12.39\,x + 0.205525\,x(x-20) - 0.00220188\,x(x-20)(x-40)$. *Persamaan (7.6) mengevaluasinya dari dalam ke luar: mulai dari* c_n*, kalikan dengan* $(x - x_{n-1})$*, tambahkan* c_{n-1}*, dan seterusnya. Cara ini hemat dan juga stabil, karena tidak pernah menghitung pangkat tinggi yang besar lalu saling mengurangkannya. Menambah tanda baru ke bagan dinding hanya menambah satu baris miring pada tabel dan satu suku pada polinomialnya; semua koefisien lama tetap berlaku. Buku: Subbab 7.3.2–7.3.3.*

Galat Interpolasi Polinomial

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i) \tag{7.7}$$

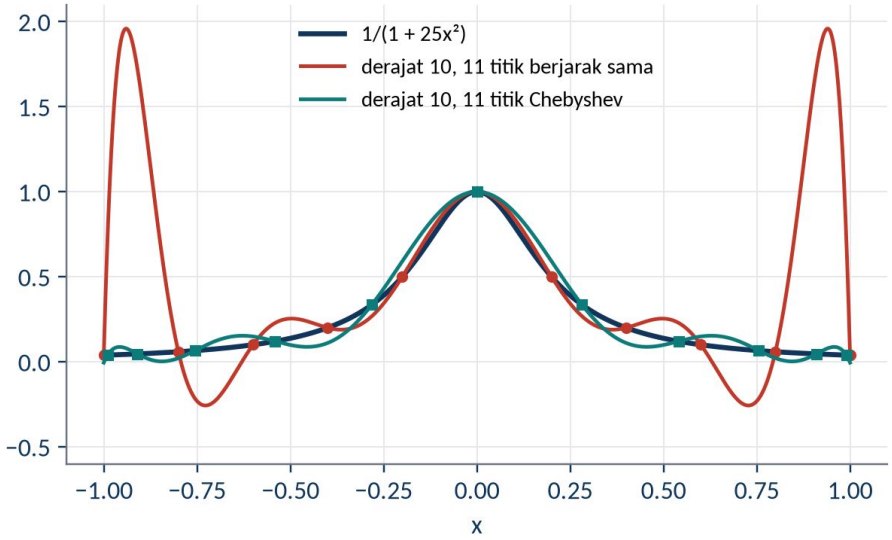
Dua faktor: faktor **turunan** yang milik fungsinya, dan **polinomial simpul** yang milik pilihan titik kita.

Polinomial simpul kecil di tengah simpul berjarak sama dan **besar di dekat ujung**.

Persamaan (7.7) menjelaskan keberhasilan sekaligus bencana. Di luar tabel polinomial simpul tumbuh cepat: itulah sebabnya ekstrapolasi berbahaya. Untuk tangki, turunan-turunan tinggi V menjadi tak hingga di dasar dan puncak, tempat permukaan menyempit menjadi garis; maka faktor turunan pun paling besar di sana. Kedua faktor bersekongkol di tempat yang sama, dan itulah sebabnya semua metode hari ini paling kesulitan di dekat ujung tangki. Buku: Subbab 7.3.4.

Peringatan Runge

$$f(x)=\frac{1}{1+25x^2},\ -1\le x\le 1 \tag{7.8}$$



Gambar 7.2 Fungsi Runge dan interpolan derajat 10-nya: melalui 11 titik berjarak sama polinomialnya berayun liar di dekat ujung; melalui 11 titik Chebyshev ia mengikuti fungsinya dengan rapat.

$$x_i = \cos \frac{(2i+1)\pi}{2n+2}, i = 0, 1, \ldots, n \tag{7.9}$$

Persamaan (7.8) mulus dan tampak tidak berbahaya, tetapi Gambar 7.2 memperlihatkan interpolasi berjarak sama makin buruk ketika derajatnya naik. ***Titik Chebyshev*** *Persamaan (7.9) berkerumun ke arah ujung, tempat polinomial simpul besar, dan menjaga galatnya tetap kecil di mana-mana. Buku: Subbab 7.3.5.*

Hermite: Bila Kemiringan Juga Diketahui

Cocokkan nilai *dan* kemiringan s_0, s_1 di dua simpul berjarak h: sebuah kubik. Di titik tengahnya

$$p\left(\frac{x_0+x_1}{2}\right)=\frac{y_0+y_1}{2}+\frac{h}{8}(s_0-s_1) \tag{7.10}$$

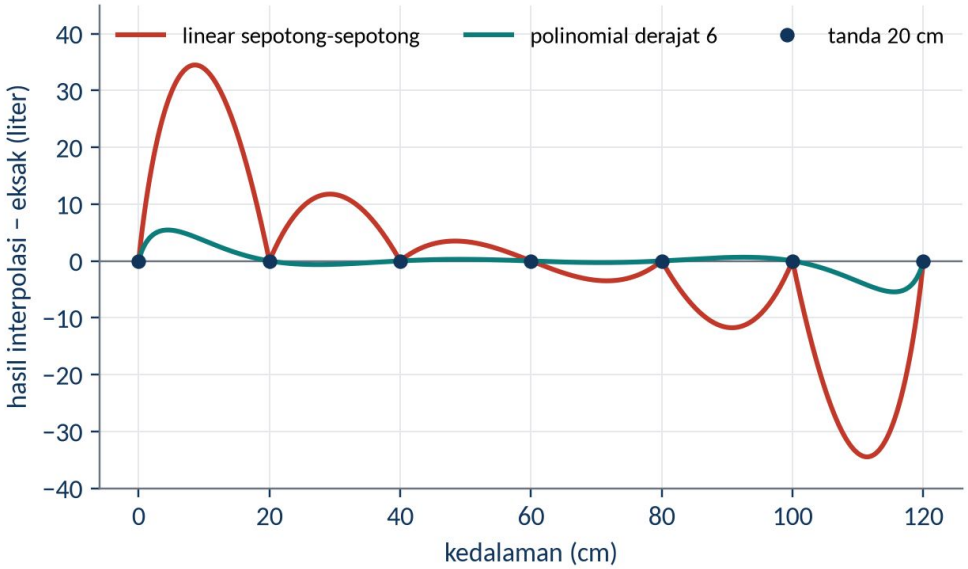
Persamaan (7.10): titik tengah garis lurus ditambah koreksi dari selisih kemiringan. Untuk tangki, kemiringannya adalah lebar permukaan kali panjang, yang kita ketahui dari Bab 3; di dasar tangki Hermite memangkas galat garis lurus sebanyak dua pertiga. Buku: Subbab 7.3.6.

Menjalankannya: Garis lawan Polinomial

```
depth   table   linear   error  Lagrange   Newton   error
   10    90.03   123.90   33.87     93.62    93.62    3.59
   30   442.21   453.91   11.70    441.62   441.62   -0.59
   50   892.09   895.50    3.40    892.37   892.37    0.28
   70  1369.86  1366.45   -3.41   1369.57  1369.57   -0.29
   90  1819.73  1808.04  -11.69   1820.34  1820.34    0.61
  110  2171.92  2138.05  -33.87   2168.32  2168.32   -3.60
largest error on 0..120 cm: linear 34.49 L, degree 6 5.45 L
```

`interp.py` dan `Interp.java` menguji di enam garis 10 cm yang tidak ada di bagan dinding. Lagrange dan Newton mencetak angka yang sama, sebagaimana mestinya: dua cara menulis satu polinomial. Polinomial derajat 6 sepuluh kali lebih baik daripada garis lurus di dekat ujung. Koefisien Newton terakhirnya, 2.17×10^{-12}*, akan nol untuk data eksak; itu adalah pembulatan tabel ke dua desimal yang diperbesar. Buku: Subbab 7.5–7.6.*

Kurva-kurva Galat



Gambar 7.3 Galat interpolasi linear sepotong-sepotong dan polinomial derajat 6 dari tanda-tanda 20 cm, diukur terhadap rumus eksak tangki.

```
Runge's function 1/(1+25x^2) on [-1, 1]: largest error
   n   equally spaced   Chebyshev points
   5           0.4327             0.5559
  10           1.9156             0.1092
  15           2.1076             0.0831
  20          59.8223             0.0153
```

Gambar 7.3: setiap potongan lurus punya lengkungnya sendiri, paling tinggi di dekat ujung; polinomialnya hampir sempurna di tengah dan bergoyang di ujung. Percobaan Runge dalam angka: pada $n = 20$ titik berjarak sama memberi galat 59,8, titik Chebyshev 0,015. Buku: Subbab 7.6.

Spline Kubik: Rantai Kubik yang Mulus

Satu kubik per selang, disambung sehingga nilai, kemiringan, dan **kelengkungan** M_i cocok di setiap simpul. Dengan $a = x_{i+1} - x$, $b = x - x_i$:

$$S(x) = \frac{M_i a^3 + M_{i+1} b^3}{6 h_i} + \left(\frac{y_i}{h_i} - \frac{M_i h_i}{6} \right) a + \left(\frac{y_{i+1}}{h_i} - \frac{M_{i+1} h_i}{6} \right) b \tag{7.11}$$

Mencocokkan kemiringan memberi satu persamaan per simpul dalam, dengan Δ_i kemiringan potongan ke-i:

$$h_{i-1} M_{i-1} + 2(h_{i-1} + h_i) M_i + h_i M_{i+1} = 6(\Delta_i - \Delta_{i-1}), \; i = 1, \ldots, n-1 \tag{7.12}$$

Persamaan (7.11) mencocokkan nilai dan kelengkungan menurut konstruksinya. Persamaan (7.12) tridiagonal dan dominan diagonal tegas — kasus paling ramah di Bab 5. Dua syarat masih kurang: ***natural*** *($M = 0$ di ujung),* ***terjepit*** *(kemiringan ujung diketahui), atau* ***not-a-knot****. Buku: Subbab 7.4.1–7.4.2.*

Algoritma Thomas

$$d_i \leftarrow d_i - \frac{l_i}{d_{i-1}} u_{i-1}, r_i \leftarrow r_i - \frac{l_i}{d_{i-1}} r_{i-1} \, (i = 1, \ldots, n); \; M_n = \frac{r_n}{d_n}, M_i = \frac{r_i - u_i M_{i+1}}{d_i} \tag{7.13}$$

Eliminasi Gauss yang tahu bahwa matriksnya hanya punya tiga diagonal: **waktu linear** dan tanpa pivoting.

Dengan tangan, spline natural di 10 cm: $M_0 = 0$, $M_1 = 0.5990$ memberi $S(10) = 4.99 + (12.39 - 2.00) \times 10 = 108.93$ L.

Persamaan (7.13): satu sapuan maju dan satu sapuan mundur, sekitar $8n$ operasi. Dominansi diagonal menjamin tidak ada d_i yang menjadi nol, sehingga pivoting tidak diperlukan. Untuk tujuh tanda tangki, sistemnya hanya lima persamaan, tetapi algoritma yang sama menangani ribuan titik dalam sekejap. Buku: Subbab 7.4.3.

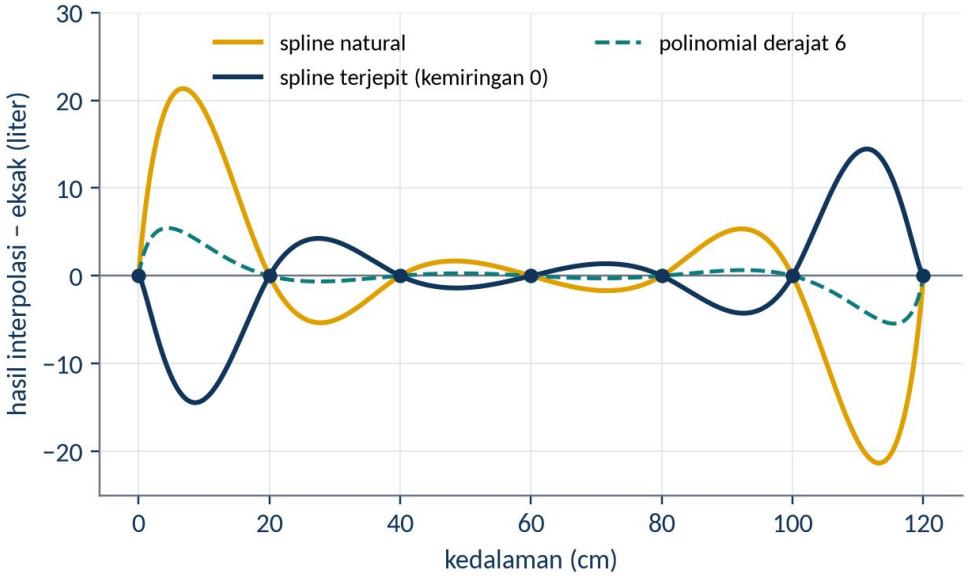
Menjalankannya: Spline

```
curvatures M_i = S''(x_i) at the 20 cm marks
  natural  0.0000  0.5990  0.0705  0.0001 -0.0706 -0.5989  0.0000
  clamped  1.7989  0.1193  0.1904  0.0001 -0.1905 -0.1192 -1.7989

depth   table  natural   error  clamped   error
   10    90.03   108.93   18.90    75.95  -14.08
   30   442.21   437.17   -5.04   446.17    3.96
   50   892.09   893.73    1.64   890.73   -1.36
```

Spline natural punya kelengkungan nol di ujung karena ditetapkan begitu; spline terjepit menemukan 1,80, lebih dekat ke kebenaran — kelengkungan sejati tangki tak hingga di sana. Keduanya punya $M_3 \approx 0$ *di 60 cm, titik beloknya. Spline mengalahkan garis lurus di mana-mana tetapi kalah dari polinomial derajat 6 pada tangki ini. Buku: Subbab 7.6.*

Tempat Spline Menderita



Gambar 7.4 Galat spline natural, spline terjepit (kemiringan ujung 0), dan polinomial derajat 6 melalui tanda-tanda 20 cm. Di dekat ujung, tempat kelengkungan tangki tak hingga, spline paling menderita.

> **WAWASAN KUNCI:** Tidak ada interpolan yang terbaik di mana-mana. Volume tangki sangat mulus di tengah dan singular di ujung; jawaban yang benar adalah mengukur. Tanda yang lebih rapat paling menolong: tanda setiap 10 cm memangkas galat terburuk spline terjepit dari 14,45 L menjadi 5,08 L.

Gambar 7.4: program juga membaca 47 cm sebagai 821,4 L, tepat sampai liternya. Bagi Pak Darto nasihat praktisnya sederhana: cat tanda setiap 10 cm, dan pakai kurva, bukan garis lurus, di dekat dasar dan puncak. Buku: Subbab 7.6.

Memeriksa dengan Pustaka

```
NumPy 2.5.3, SciPy 1.18.1
degree 6: Newton vs BarycentricInterpolator 9.1e-13
numpy.interp largest error 34.49 L
natural: ours vs CubicSpline 4.5e-13, largest error 21.34 L
clamped: ours vs CubicSpline 1.2e-12, largest error 14.45 L
not-a-knot (SciPy default): largest error 7.87 L
```

`BarycentricInterpolator` *adalah polinomial Lagrange dalam bentuk yang stabil secara numerik;* `numpy.interp` *adalah garis lurus;* `CubicSpline` *mengerjakan ketiga syarat ujung. Bawaan SciPy, not-a-knot, adalah spline terbaik di sini dengan 7,87 L. Bila tabelnya berasal dari rumus yang diketahui, seperti yang ini, pilihan terbaik adalah memakai rumusnya. Buku: Subbab 7.7–7.8.*

Membuatnya Lebih Baik

Pakai apa yang Anda ketahui. Bila sebuah tabel berasal dari rumus yang diketahui, pakailah rumusnya. Interpolasi untuk tabel yang asal-usulnya tidak diketahui atau terlalu rumit untuk dimodelkan.

Taruh tanda di tempat fungsinya melengkung. Tujuh tanda pada kedalaman Chebyshev (1,5, 13,1, 34,0, 60,0, 86,0, 106,9, dan 118,5 cm) memberi polinomial derajat 6 dengan galat terburuk 3,95 L, dibanding 5,45 L untuk jarak sama.

Interpolasi monoton. Volume hanya bertambah bersama kedalaman, tetapi spline dapat turun di antara dua titik sehingga satu sentimeter bahan bakar tambahan "kehilangan" liter. PCHIP memilih kemiringan sehingga interpolannya monoton di mana pun datanya monoton.

Evaluasi barisentrik. Bentuk Lagrange yang ditulis ulang dengan bobot yang dihitung sekali dievaluasi dalam $O(n)$ dan stabil secara numerik; pada titik Chebyshev ia metode pilihan untuk derajat berapa pun.

Setiap perbaikan ini hanya satu baris di SciPy: `PchipInterpolator`, `BarycentricInterpolator`, *dan daftar kedalaman Chebyshev untuk tukang cat. Buku: Subbab 7.8.*

Metode Numerik dalam Praktik: Profitina

Dasbor analitik historis Profitina menampilkan skor visibilitas AI setiap merek **minggu demi minggu**, dengan ekspor CSV. Di antara dua pemindaian mingguan tidak ada pengukuran.

Grafik yang menarik garis di antara titik-titik mingguan sedang berinterpolasi. Spline yang mulus dapat **melampaui**: puncak di atas skor nyata tertinggi, nilai yang tidak pernah diukur.

Pilihan yang jujur adalah ruas-ruas lurus atau interpolan monoton yang menjaga bentuk seperti **PCHIP**, yang tidak pernah menciptakan maksimum yang tidak dimiliki data.

Pilihan antara garis lurus, polinomial, dan spline adalah pilihan tentang apa yang akan diceritakan kepada pembaca mengenai yang terjadi di antaranya. Buku: Subbab 7.10.

Kesalahan Umum

Menaikkan derajat demi ketelitian. Dengan jarak sama hasilnya bisa jauh lebih buruk (Runge). Pakai titik Chebyshev, atau spline.

Mengekstrapolasi. Di luar tabel polinomial simpul tumbuh cepat, begitu pula galatnya.

Menyelesaikan sistem Vandermonde. Polinomial yang sama, matriks yang berkondisi buruk; pakai Newton atau barisentrik.

Memercayai syarat ujung natural. Jepit bila kemiringan ujung diketahui; jika tidak, not-a-knot.

Lupa bahwa spline dapat melampaui. Untuk total atau persentase, pakai interpolan monoton.

Untuk tangki, drum, atau jeriken berbentuk aneh milik Anda: tuangkan volume yang diketahui sedikit demi sedikit, jeriken 20 liter cukup praktis, catat kedalaman setelah setiap tuangan, dan ukur lebih sering di dekat dasar dan puncak. Buku: Subbab 7.9 dan 7.11.

Ringkasan

1. Garis lurus (Persamaan (7.2)) sederhana; galatnya mengikuti kelengkungan: 34 L di dekat ujung tangki, 3,4 L di tengah.
2. Satu polinomial derajat n melalui $n+1$ titik: Lagrange (Persamaan (7.3)) atau Newton dengan selisih terbagi (Persamaan (7.4), Persamaan (7.5)).
3. Galatnya (Persamaan (7.7)) meledak di dekat ujung untuk jarak sama (Runge); titik Chebyshev (Persamaan (7.9)) memperbaikinya.
4. Kelengkungan spline kubik menyelesaikan sistem tridiagonal (Persamaan (7.12)) dalam waktu linear (Persamaan (7.13)).
5. Pada tangki ini polinomial derajat 6 mengalahkan spline; tanda yang lebih rapat paling menolong. Ukur sebelum percaya.

Program Java dan Python sama byte demi byte, dan NumPy serta SciPy mereproduksi setiap interpolan sampai sekitar 10^{-12}*. Buku: Subbab 7.12.*

Pertemuan Berikutnya

Bab 8 — Pencocokan Kurva: Regresi Kuadrat Terkecil. Bila datanya berderau, kurva harus lewat *dekat* titik-titiknya, bukan melaluinya.

Regresi garis lurus dan polinomial, model yang dilinearkan, dan cara menilai kecocokan.

Sebelumnya: dari bagan dinding, hitung dengan tangan bacaan garis lurus di 30 cm dan parabola melalui 20, 40, dan 60 cm.

Bandingkan kedua bacaan dengan 442,21 L dari pembuat tangki; penyelesaiannya ada di buku. Buku: Subbab 7.13–7.14.

Kuliah 8 — Pencocokan Kurva: Regresi Kuadrat Terkecil

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 8

Dosen: Irfan Subakti — yifana@gmail.com

Interpolasi melewati setiap titik, dan itu benar bila titik-titiknya eksak. Data hasil ukur tidak eksak: ia berserak. Hari ini kurvanya lewat dekat titik-titik, dipilih agar kuadrat melesetnya sekecil mungkin — kuadrat terkecil — dan kita belajar mengatakan seberapa jauh jawabannya dapat dipercaya serta cara menghindari mencocokkan derau. Buku: Bab 8, Subbab 8.1–8.14.

Cakupan Pertemuan Ini

1. Masalahnya — buku catatan pengemudi ojek: jarak, perjalanan, bensin, kecepatan, dan kesehatan baterai.

2. Garis lurus — jumlah kuadrat galat, persamaan normal, R^2, dan galat baku.

3. Lebih banyak prediktor — matriks desain, diselesaikan dengan QR dari Bab 6.

4. Kurva yang linear dalam koefisiennya — polinomial, dan eksponensial lewat logaritma.

5. Overfitting — menilai model pada data yang disisihkan, dan pemeriksaan dengan pustaka.

Tiga berkas per bahasa: alat bersama `lsq`, `line_fit` untuk garis dan eksponensial, `poly_fit` untuk polinomial. Java dan Python mencetak teks yang identik. Buku: Bab 8.

Buku Catatan Mas Dimas

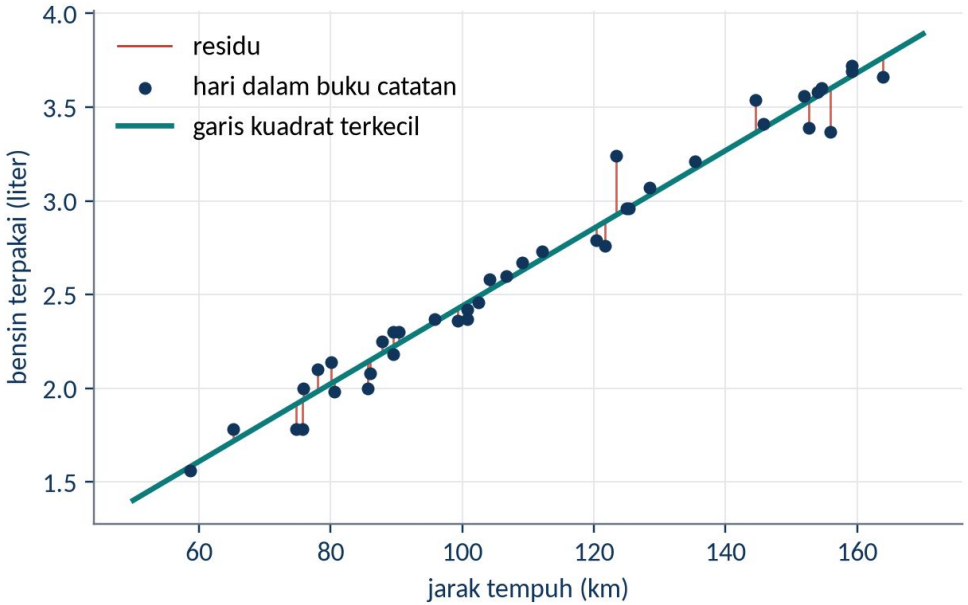
Mas Dimas mengemudikan **ojek** lewat aplikasi transportasi daring. Bensin adalah biaya hariannya yang terbesar.

Selama empat puluh hari ia mencatat **jarak**, banyak **perjalanan**, dan **liter** yang dibeli untuk mengisi ulang; beberapa hari juga **kecepatan rata-rata**; setiap dua bulan **kesehatan baterai** ponselnya.

Empat pertanyaan: Berapa sebenarnya biaya bensin per kilometer? Berapa yang perlu direncanakan untuk 120 km dan 17 perjalanan besok? Pada kecepatan berapa motornya paling irit? Kapan baterainya mencapai **80%**?

Tidak satu pun dari keempat jawaban dapat dibaca langsung dari buku catatan: setiap hari berserak di sekitar suatu kecenderungan, karena lalu lintas, hujan, menunggu, dan tanjakan. Kecenderungan itulah yang kita cari, dan serakannya memberi tahu seberapa yakin kita boleh. Buku: Subbab 8.1.

Empat Puluh Hari dalam Satu Plot



Gambar 8.1 Empat puluh hari dalam buku catatan Mas Dimas: bensin terpakai terhadap jarak tempuh, garis kuadrat terkecil, dan residunya, yaitu meleset tegak yang dibuat sekecil mungkin oleh garis itu dalam jumlah kuadratnya.

Gambar 8.1: titik-titiknya dekat sebuah garis, tetapi tidak di atasnya. Garis melalui dua titik pilihan akan bergantung pada dua titik mana yang dipilih; garis yang mendengarkan keempat puluh titik itulah yang kita bangun. Buku: Subbab 8.1–8.2.

Gagasannya: Buat Kuadrat Meleset Kecil

Untuk garis $a+b\,x$, hari ke-i meleset sebesar **residu** $e_i = y_i - (a+b\,x_i)$. Pilih a dan b yang meminimumkan

$$\mathrm{SSE}(a,b) = \sum_{i=1}^{n} (y_i - a - b\,x_i)^2 \tag{8.1}$$

Mengapa kuadrat? Meleset positif dan negatif tidak dapat saling menghapus, meleset besar lebih berbobot, dan minimumnya ditemukan dengan menyelesaikan sistem *linear*.

Persamaan (8.1): Legendre menerbitkannya pada 1805 untuk orbit komet, dan Gauss memakainya untuk orbit planet. Meminimumkan jumlah meleset mutlak juga masuk akal, tetapi tidak punya rumus dan butuh metode optimasi Bab 13. Buku: Subbab 8.2.

Persamaan Normal dan Garisnya

Samakan kedua turunan parsial SSE dengan nol:

$$\begin{aligned} n\,a + (\sum x_i) b &= \sum y_i \\ (\sum x_i) a + (\sum x_i^2) b &= \sum x_i y_i \end{aligned} \tag{8.2}$$

Diselesaikan dalam bentuk terpusat yang stabil:

$$b = \frac{S_{xy}}{S_{xx}}, a = \acute{y} - b\,\acute{x}, S_{xx} = \sum_i (x_i - \acute{x})^2, S_{xy} = \sum_i (x_i - \acute{x})(y_i - \acute{y}) \tag{8.3}$$

Persamaan (8.2) adalah kasus 2×2 *dari persamaan normal Bab 6. Persamaan (8.3) mengurangkan rata-rata lebih dulu, yang menghindari pembatalan dari Bab 1, dan memperlihatkan bahwa garisnya selalu melalui titik rata-rata* $(\acute{x}, \acute{y})$*. Buku: Subbab 8.3.1.*

Dengan Tangan: Lima Hari Pertama

Tabel 8.1 Garis kuadrat terkecil melalui lima hari pertama, dengan tangan: simpangan dari rata-rata, kuadratnya, dan hasil kalinya.

Hari	x_i (km)	y_i (L)	$x_i - x$	$y_i - y$	$(x_i - x)^2$	Hasil kali
1	74,7	1,78	−32,86	−0,812	1079,78	26,6823
2	95,8	2,37	−11,76	−0,222	138,30	2,6107
3	154,6	3,60	47,04	1,008	2212,76	47,4163
4	87,8	2,25	−19,76	−0,342	390,46	6,7579
5	124,9	2,96	17,34	0,368	300,68	6,3811
jumlah	537,8	12,96	0	0	4121,97	89,848

Tabel 8.1: $\acute{x} = 107.56$, $\acute{y} = 2.592$, *jadi* $b = 89.848 / 4121.97 = 0.0218$ *L/km dan* $a = 2.592 - 0.0218 \times 107.56 = 0.248$ *L. Lima hari memberi taksiran awal; empat puluh hari memberi yang lebih baik. Perhatikan bahwa kolom simpangan berjumlah nol, sebagaimana mestinya, dan bahwa seluruh hitungan hanya butuh jumlah-jumlah, satu kali lewat buku catatan setelah rata-ratanya diketahui. Buku: Subbab 8.2.*

Seberapa Baik Garisnya?

Bagian variasi yang dijelaskan, dengan SST jumlah kuadrat $y_i - \acute{y}$:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} \tag{8.4}$$

Meleset yang khas, dan ketidakpastian kemiringannya:

$$s_e = \sqrt{\frac{\text{SSE}}{n-2}} \tag{8.5}$$

$$\text{SE}(b) = \frac{s_e}{\sqrt{S_{xx}}} \tag{8.6}$$

Persamaan (8.4): 1 berarti cocok sempurna, 0 berarti garisnya tidak menjelaskan apa-apa. Persamaan (8.5) membagi dengan $n-2$ karena dua bilangan ditaksir dari data. Persamaan (8.6): kemiringan lebih terkenali bila nilai x tersebar lebar — berkendaralah pada hari pendek maupun panjang. Sekitar dua galat baku di kiri-kanan memberi rentang 95%. Buku: Subbab 8.3.2.

Meramalkan Satu Hari Baru

$$\hat{y}_0 \pm t\, s_e \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \acute{x})^2}{S_{xx}}} \tag{8.7}$$

Angka “1” adalah serakan hari baru itu sendiri; dua suku lainnya ketidakpastian garis, yang membesar menjauhi $\acute{x}$. Untuk 150 km: **3,48 ± 0,21 L**. Rencanakan ujung atasnya, dan motornya tidak akan kehabisan bensin.

Persamaan (8.7) adalah selang prediksi, lebih lebar daripada ketidakpastian garisnya sendiri, karena satu hari berserak. Jauh dari data suku terakhir membesar: ramalan untuk 400 km akan tidak pasti sekaligus di luar rentang yang pernah dilihat model. Buku: Subbab 8.3.3.

Lebih Banyak Prediktor: Matriks Desain

Perjalanan juga memakan bensin: menunggu, menyalakan mesin, memutar. Model $y \approx \beta_0 + \beta_1 x_1 + \beta_2 x_2$, satu baris per hari:

$$y \approx X\,\beta \tag{8.8}$$

$$X^T X\,\beta = X^T y \tag{8.9}$$

*Persamaan (8.8): **matriks desain** X punya kolom berisi angka satu, kolom kilometer, dan kolom perjalanan. Persamaan (8.9) adalah persamaan normal Bab 6; kita menyelesaikannya dengan QR Householder, tanpa pernah membentuk $X^T X$. Buku: Subbab 8.4.*

Menjalankannya: Garis dan Prediktor

```
40 days, mean 111.75 km and 2.6825 L per day
Sxx = 36092.60, Sxy = 748.4575
liters = 0.3652 + 0.020737 * km
R^2 = 0.9757, standard error 0.1009 L
slope 0.020737 +/- 0.001062 L/km (about 95%)
largest residual: day 17, +0.316 L

liters = 0.2654 + 0.018568 * km + 0.02285 * trips
R^2 = 0.9894, standard error 0.0675 L
tomorrow, 120 km and 17 trips: 2.88 L
```

`line_fit.py` dan `LineFit.java`. Hari 17 memakai 0,316 L lebih banyak daripada ramalan, tiga galat baku — layak ditanyakan sebabnya sebelum dihapus. Dengan perjalanan, R^2 naik ke 0,9894 dan meleset khas turun ke 0,0675 L. Koefisien perjalanan, 0,023 L per perjalanan, adalah pengaruhnya pada jarak yang tetap. Perjalanan 10 km berbiaya sekitar 0,2 L; membandingkannya dengan tarif adalah keputusan Mas Dimas. Buku: Subbab 8.8.

Kurva yang Linear dalam Koefisiennya

Polinomial linear dalam **koefisiennya**, meskipun melengkung terhadap x:

$$y \approx \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d \tag{8.10}$$

Efisiensi bensin terhadap kecepatan berbentuk bukit. Dengan kecepatan terskala t = (kecepatan − 45)/25 dan parabola $c_0 + c_1 t + c_2 t^2$, puncaknya di

$$t^* = -\frac{c_1}{2c_2}, \text{speed}^* = 45 + 25\,t^* \tag{8.11}$$

Persamaan (8.10): setiap pangkat adalah satu kolom lagi dari X. Pusatkan dan skalakan lebih dulu: pangkat 70 sampai kedelapan panjangnya 15 digit dan kolom-kolomnya hampir sejajar. Persamaan (8.11) adalah puncak parabola dalam peubah terskala, dikembalikan ke km/jam. Buku: Subbab 8.5.1–8.5.2.

Eksponensial lewat Logaritma

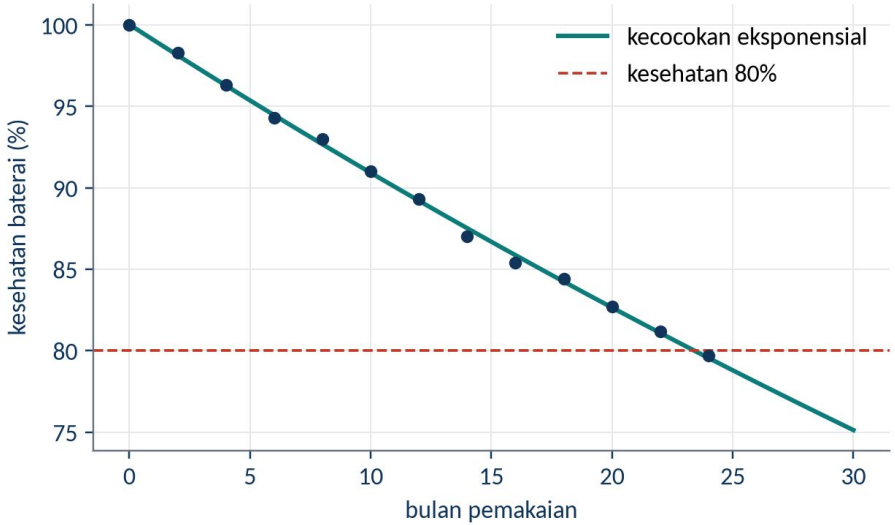
Baterai kehilangan kapasitas kira-kira sebanding dengan sisanya: $H(t) = C\,e^{kt}$, tidak linear dalam C dan k. Logaritmanya linear:

$$\ln H = \ln C + k\,t \tag{8.12}$$

Cocokkan garis lurus pada $(t_i, \ln H_i)$, kembalikan, lalu selesaikan $C\,e^{kt} = 80$ untuk t.

Persamaan (8.12): garis lurus dengan titik potong $\ln C$ dan kemiringan k. Mencocokkan $\ln y$ meminimumkan galat relatif, bukan mutlak; untuk jawaban teliti dalam satuan aslinya, periksa dengan pencocokan tak linear langsung, seperti dalam perbandingan dengan pustaka. Buku: Subbab 8.5.3.

Kurva Baterai



Gambar 8.2 Kesehatan baterai ponsel Mas Dimas setiap dua bulan, pencocokan eksponensial melalui logaritma, dan tingkat 80% yang dicapainya setelah sekitar 23.4 bulan.

```
battery health = 100.03 * exp(-0.00954 * month)
80% is reached after about 23.4 months
```

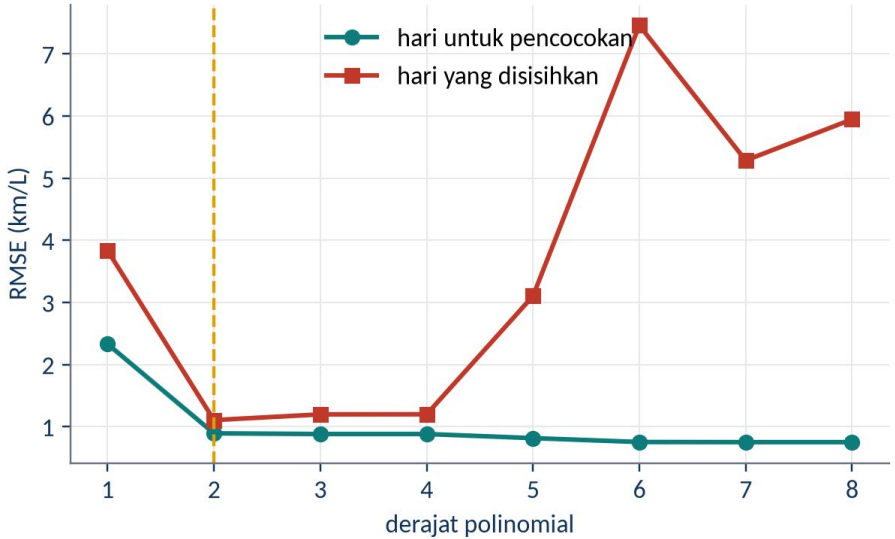
Gambar 8.2: selama dua tahun, eksponensial dengan laju kecil tampak hampir lurus; garis lurus yang dicocokkan langsung akan mengatakan 23,1 bulan. Bedanya kecil di sini, tetapi eksponensial tetap masuk akal di luar data, sedangkan garis lurus lama-lama mencapai nol. Buku: Subbab 8.8.

Overfitting: Derajat Lebih Tinggi, Cocok Lebih Baik, Model Lebih Buruk

Menambah kolom hanya dapat menurunkan SSE, jadi R^2 naik di setiap derajat. Polinomial derajat 35 melalui 36 titik mencapai $R^2 = 1$ dan tidak meramalkan apa-apa.

Cocokkan pada 24 hari, **sisihkan 12**, dan nilai pada hari-hari yang disisihkan:

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{8.13}$$



Gambar 8.3 RMSE pencocokan polinomial derajat 1 sampai 8 pada 24 hari pencocokan dan pada 12 hari yang disisihkan. Galat pencocokan hanya turun; galat hari yang disisihkan paling kecil pada derajat 2 dan tumbuh tajam mulai derajat 5.

Persamaan (8.13) pada data yang belum pernah dilihat model adalah hakim yang jujur. Gambar 8.3: bentuk U klasik galat data yang disisihkan — terlalu sederhana di kiri, terlalu lentur di kanan. Bab 14 bertemu gagasan yang sama sebagai himpunan validasi dalam machine learning. Buku: Subbab 8.6.

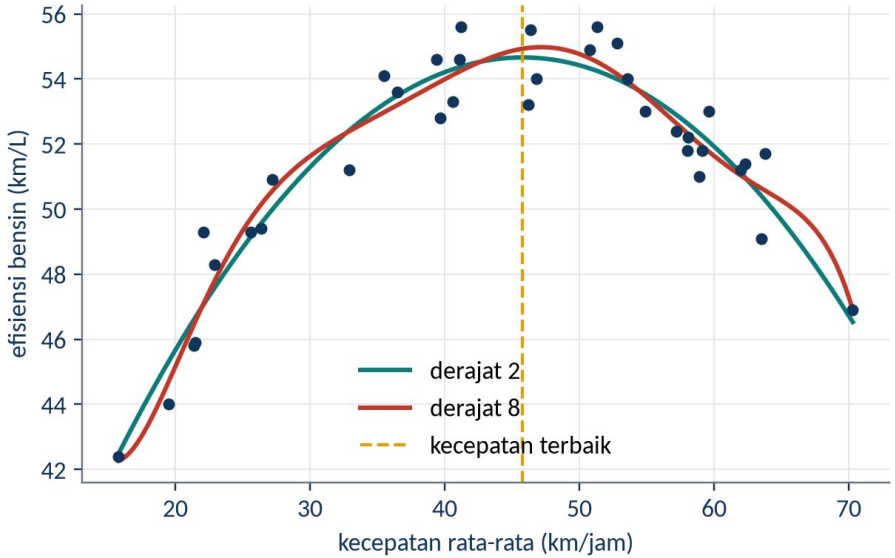
Menjalankannya: Memilih Derajat

```
36 days: 24 to fit, 12 held back
degree  R^2 (fit)  RMSE fit  RMSE held back
     1     0.3998     2.330          3.834
     2     0.9120     0.892          1.106
     3     0.9141     0.881          1.198
     4     0.9141     0.881          1.199
     5     0.9265     0.815          3.103
     6     0.9373     0.753          7.460
     7     0.9377     0.751          5.286
     8     0.9377     0.751          5.948

all days, degree 2: km/L = 54.653 +0.535 x -8.459 x^2
  where x = (speed - 45) / 25
best average speed 45.8 km/h, about 54.7 km/L
```

`poly_fit.py` dan `PolyFit.java`. *Garis lurus cocok buruk pada kedua kelompok karena datanya melengkung. Derajat 2 menurunkan galat data yang disisihkan ke 1,106; derajat 3 dan 4 sedikit lebih cocok pada hari pencocokan tetapi lebih buruk pada hari yang disisihkan; mulai derajat 5 galatnya meledak. Derajat 2 menang, dan puncaknya menaruh kecepatan rata-rata terbaik pada* ***45,8 km/jam****. Buku: Subbab 8.8.*

Kurva Kecepatan



Gambar 8.4 Efisiensi bensin terhadap kecepatan rata-rata pada 36 hari, dengan pencocokan derajat 2 (puncaknya pada 45.8 km/jam ditandai) dan pencocokan derajat 8, yang mengikuti hari-hari tertentu alih-alih kecenderungannya.

Gambar 8.4: kurva derajat 8 mengejar hari-hari tertentu dan bergoyang di tempat yang tidak pernah diukur; parabola menceritakan kisah yang dapat dipakai pengendara. Buku: Subbab 8.8.

Memeriksa dengan Pustaka

```
NumPy 2.5.3, SciPy 1.18.1
linregress: intercept 0.3652, slope 0.020737, R^2 0.9757
  standard error of the slope 0.000531
lstsq with trips: [0.265357 0.018568 0.022854]
Polynomial.fit degree 2: best speed 45.8 km/h, 54.7 km/L
curve_fit: health = 100.04 * exp(-0.00956 * month), 80% after 23.4 months
```

`linregress` melaporkan galat baku kemiringan, 0,000531; "±0,001062" kita adalah dua kalinya, rentang 95%. `lstsq` memakai dekomposisi nilai singular, anggota keluarga QR yang paling tangguh. `curve_fit` mencocokkan eksponensial secara langsung dalam persen dan memberi kurva yang hampir sama serta 23,4 bulan yang sama. Buku: Subbab 8.9.

Metode Numerik dalam Praktik: Profitina

Dasbor analitik historis Profitina menyimpan **skor visibilitas AI mingguan** untuk setiap merek, dengan peringatan dan ekspor CSV.

“Apakah situsnya membaik?” dijawab dengan jujur oleh **kemiringan** garis kuadrat terkecil melalui minggu-minggu itu, beserta galat bakunya Persamaan (8.6).

$\nearrow 0.8 \pm 0.3$ poin per minggu: membaik, dan bukan kebetulan. $\nearrow 0.8 \pm 1.5$: belum dapat dikatakan, betapapun menggembirakannya dua minggu terakhir.

Dua belas minggu memberi kemiringan yang diketahui sampai sekitar ± 0.25 *poin per minggu; dua puluh enam minggu memberi galat baku lebih dari tiga kali lebih kecil. Kesabaran membeli kepastian, dan itu patut dikatakan terus terang kepada pemilik merek yang menginginkan vonis setelah tiga minggu. Angka memberi saran; manusia yang memutuskan. Buku: Subbab 8.12.*

Kesalahan Umum

Menilai model hanya dari R^2. Ia tidak pernah turun bila kolom ditambah; bandingkan pada data yang disisihkan.

Polinomial derajat tinggi dalam satuan mentah. Pusatkan dan skalakan, dan selesaikan dengan QR.

Meminimumkan meleset tegak bila galatnya ada di x. Di sini jaraknya teliti dan bensinnya tidak, jadi meleset tegak sudah benar.

Mengekstrapolasi hasil pencocokan. Model hanya dapat dipercaya dalam rentang datanya.

Membuang "pencilan" tanpa bertanya mengapa. Hari 17 mungkin salah catat, atau satu-satunya hari dengan menunggu lama di bawah hujan.

Rutinitas yang sama mencocokkan penjualan terhadap suhu untuk warung es teh, listrik terhadap jam nyala AC, atau pemakaian air terhadap jumlah tamu di penginapan kecil: taruh kolom-kolomnya di berkas CSV, plot dulu, cocokkan, lihat residunya, dan sisihkan sebagian data. Buku: Subbab 8.11 dan 8.13.

Ringkasan

1. Kuadrat terkecil meminimumkan jumlah kuadrat residu (Persamaan (8.1)); untuk garis, kemiringan S_{xy}/S_{xx} melalui titik rata-rata (Persamaan (8.3)).

2. R^2 (Persamaan (8.4)), galat baku (Persamaan (8.5)), dan galat baku kemiringan (Persamaan (8.6)) mengatakan seberapa jauh ia dapat dipercaya: 0.0207 ± 0.0011 L/km.

3. Hari baru diramalkan dengan selang yang lebih lebar (Persamaan (8.7)): 3.48 ± 0.21 L untuk 150 km.

4. Beberapa prediktor masuk lewat matriks desain (Persamaan (8.8)), diselesaikan dengan QR; polinomial dan eksponensial yang dilogaritmakan juga kuadrat terkecil linear.

5. Hanya galat data yang disisihkan (Persamaan (8.13)) yang menyingkap overfitting; di sini derajat 2 menang, kecepatan terbaik 45,8 km/jam, baterai 80% setelah 23,4 bulan.

Program Java dan Python sama byte demi byte, dan `linregress`, `lstsq`, `Polynomial.fit`, *serta* `curve_fit` *membenarkan setiap hasil. Buku: Subbab 8.14.*

Pertemuan Berikutnya

Bab 9 — Diferensiasi Numerik. Kecepatan dan percepatan dari posisi yang diukur berselang.

Selisih maju, mundur, dan pusat, galatnya, ekstrapolasi Richardson — dan apa yang dilakukan derau terhadap turunan.

Sebelumnya: dengan tabel lima hari, periksa bahwa residunya berjumlah nol dan hitung R^2 lima hari itu.

Kelima nilai $y_i - \acute{y}$ memberi SST; penyelesaiannya ada di buku, bersama kuadrat terkecil berbobot untuk hari-hari yang keandalannya berbeda. Buku: Subbab 8.15–8.16.

Kuliah 9 — Diferensiasi Numerik

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 9

Dosen: Irfan Subakti — yifana@gmail.com

Turunan adalah laju perubahan: kecepatan dari jarak, percepatan dari kecepatan, pertumbuhan dari skor. Bila yang kita punya hanya sampel, garis singgung diganti tali busur. Hari ini kita menurunkan rumus-rumusnya beserta galatnya dari teorema Taylor, bertemu lagi dengan dilema ukuran langkah, mempertajam taksiran dengan ekstrapolasi Richardson, dan mempelajari apa yang dilakukan data berderau terhadap turunan. Buku: Bab 9, Subbab 9.1–9.12.

Cakupan Pertemuan Ini

1. **Masalahnya** — aplikasi berkata “pengereman keras terdeteksi”. Benarkah?
2. **Rumus-rumusnya** — beda maju, mundur, pusat, dan beda kedua, dari teorema Taylor.
3. **Dua jenis galat** — pemotongan lawan pembulatan, dan langkah optimal.
4. **Ekstrapolasi Richardson** — menggabungkan dua langkah untuk menghapus galat terdepan.
5. **Data berderau** — penganalisis perjalanan, alarm palsu, dan menghaluskan sebelum menurunkan.

Penganalisis perjalanan ada dalam Java dan Python dan mencetak tabel yang sama; `np.gradient` dari NumPy mereproduksi kecepatannya dengan tepat. Buku: Bab 9.

Laporan Pengereman Mas Dimas

Mas Dimas mengemudikan ojek daring di **Yogyakarta**, perjalanan pendek di sekitar Jalan Kaliurang.

Suatu malam aplikasinya berkata: ***harsh braking detected***. Beberapa peringatan, dan skor keselamatan pengemudi turun, bersama peluang mendapat order yang lebih baik.

Ia ingat mengerem keras karena seekor kucing pada sekitar 40 km/jam — benar, dan ia akan melakukannya lagi. Tetapi pada perjalanan lain aplikasinya mengeluh padahal ia hanya berhenti pelan di depan pagar pelanggan.

Ponselnya hanya mencatat **jarak, sekali per detik**. Bagaimana ia tahu seberapa keras Mas Dimas mengerem?

Kecepatan adalah turunan pertama jarak, percepatan turunan kedua. Ponsel harus menaksir keduanya dari sampel — dan sampelnya berderau. Buku: Subbab 9.1.

Gagasannya: Tali Busur alih-alih Garis Singgung

Tabel 9.1 Lima bacaan jarak dari perjalanan Mas Dimas, satu per detik, di sekitar pengereman keras.

Waktu t (s)	16	17	18	19	20
jarak s (m)	148,2	159,5	168,7	174,4	176,5

Kecepatan pada 18 s dari tetangga di kedua sisi: $(174.4 - 159.5)/2 = 7.45$ m/s.

Percepatan dari tiga bacaan, **beda kedua**: $(174.4 - 2\times 168.7 + 159.5)/1^2 = -3.5$ m/s².

Tabel 9.1: perlambatan 3,5 m/s² berarti kehilangan kecepatan 12,6 km/jam setiap detik — rem keras sungguhan. Pertanyaannya adalah seberapa teliti bilangan seperti ini, dan itu butuh teorema Taylor. Buku: Subbab 9.2.

Menurunkan Rumus dari Teorema Taylor

$$f(x+h)=f(x)+hf'(x)+\frac{h^2}{2}f''(x)+\frac{h^3}{6}f'''(x)+\cdots+\frac{h^n}{n!}f^{(n)}(x)+R_n \tag{9.1}$$

Berhenti setelah turunan pertama dan selesaikan untuknya — **beda maju**, orde satu:

$$f'(x)\approx\frac{f(x+h)-f(x)}{h},\text{error}=-\frac{h}{2}f''(\xi) \tag{9.2}$$

Persamaan (9.1) adalah alat di balik hampir setiap taksiran galat mata kuliah ini. Persamaan (9.2): bagi dua h dan galatnya terbagi dua. Beda mundur adalah bayangan cerminnya, dengan tanda galat yang berlawanan. Untuk log jarak Mas Dimas, beda maju pada 18 s memakai 18 dan 19 s saja, sehingga ia sebenarnya menaksir kecepatan pada 18,5 s; itulah sumber galat orde satunya. Rumus yang seimbang di kedua sisi akan menghapus galat itu, seperti ditunjukkan langkah berikutnya. Buku: Subbab 9.3.1–9.3.2.

Beda Pusat dan Turunan Kedua

Kurangkan uraian $f(x+h)$ dan $f(x-h)$: suku h^2 saling menghapus.

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \text{error} = -\frac{h^2}{6} f'''(\xi) \tag{9.3}$$

Jumlahkan keduanya: suku-suku ganjil saling menghapus.

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \text{error} = -\frac{h^2}{12} f^{(4)}(\xi) \tag{9.4}$$

Persamaan (9.3) berorde dua: bagi dua h dan galatnya turun empat kali, dengan dua nilai fungsi yang sama. Persamaan (9.4) juga berorde dua, dan eksak untuk kubik. Perhatikan h^2 di penyebutnya: galat baca pada setiap nilai diperbesar $1/h^2$ kali. Buku: Subbab 9.3.3–9.3.4.

Rumus untuk Ujung-ujung

Di bacaan pertama tidak ada tetangga kiri. Pilih koefisien sehingga suku-suku Taylor saling menghapus sampai h^2 — **koefisien tak tentu**:

$$f'(x) \approx \frac{-3f(x)+4f(x+h)-f(x+2h)}{2h}, \text{error} = \frac{h^2}{3} f'''(\xi) \tag{9.5}$$

Tabel 9.2 Rumus beda hingga untuk turunan pertama dan kedua, beserta banyaknya nilai fungsi yang diperlukan dan galat pemotongannya (ξ tidak ditulis).

Turunan	Rumus	Titik	Galat pemotongan
f' maju	$[f(x+h)-f(x)]/h$	2	$-\frac{h}{2}f''$
f' pusat	$[f(x+h)-f(x-h)]/(2h)$	2	$-\frac{h^2}{6}f'''$
f' maju tiga titik	$[-3f(x)+4f(x+h)-f(x+2h)]/(2h)$	3	$+\frac{h^2}{3}f'''$
f' pusat lima titik	$[f(x-2h)-8f(x-h)+8f(x+h)-f(x+2h$	4	$+\frac{h^4}{30}f^{(5)}$
f'' pusat	$[f(x+h)-2f(x)+f(x-h)]/h^2$	3	$-\frac{h^2}{12}f^{(4)}$

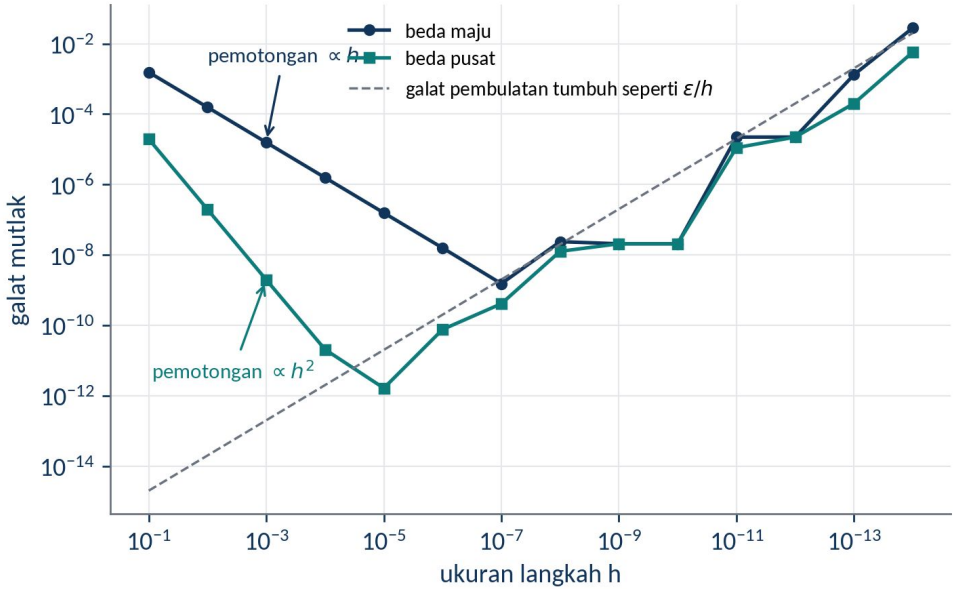
Persamaan (9.5) mempertahankan orde dua di ujung-ujung, sehingga seluruh tabel perjalanan punya ketelitian yang sama. Tabel 9.2 menghimpun rumus-rumus yang umum; buku juga mencantumkan turunan kedua satu sisi empat titik yang dipakai di ujung. Buku: Subbab 9.3.5.

Dua Jenis Galat, Lagi

```
     h     forward        error    central        error
 1e-01  0.2484567313  1.54e-03  0.2500195366  1.95e-05
 1e-02  0.2498439450  1.56e-04  0.2500001953  1.95e-07
 1e-03  0.2499843770  1.56e-05  0.2500000020  1.95e-09
 1e-04  0.2499984375  1.56e-06  0.2500000000  2.05e-11
 1e-05  0.2499998438  1.56e-07  0.2500000000  1.64e-12
 1e-06  0.2499999843  1.57e-08  0.2499999999  7.61e-11
```

`diff_steps.py` dan `DiffSteps.java` menurunkan $\sqrt{x}$ di 4 (nilai sejati 0,25). Galat beda maju turun sepuluh kali per baris, galat beda pusat seratus kali — sampai $h = 10^{-5}$. Di bawahnya beda pusat makin buruk: galat pembulatan, sekitar $2\varepsilon f(x)/h$, mengambil alih. Buku: Subbab 9.4.

Langkah Optimal



Gambar 9.1 Galat beda maju dan beda pusat dari $\sqrt{x}$ di $x=4$ terhadap langkah h (kedua sumbu logaritmik, h mengecil ke kanan). Galat pemotongan turun seperti h atau h^2; galat pembulatan naik seperti ε/h.

$$h^{*}=2\sqrt{\frac{\varepsilon|f|}{M_{2}}}\text{(forward)},\; h^{*}=\left(\frac{3\varepsilon|f|}{M_{3}}\right)^{1/3}\text{(central)} \tag{9.6}$$

Gambar 9.1: dua bentuk V, yang pusat lebih rendah dan lebih ke kiri. Persamaan (9.6): sekitar 10^{-8} untuk beda maju dan 10^{-5} untuk beda pusat dalam presisi ganda. Untuk data hasil ukur, "pembulatan" adalah galat bacaan — bagi ponsel Mas Dimas sekitar 0,1 sampai 0,3 m — sehingga argumen yang sama berkata: jangan percayai langkah yang sangat kecil. Buku: Subbab 9.4.

Ekstrapolasi Richardson

Beda pusat punya galat dalam pangkat genap h: $D(h) = f'(x) + a_1 h^2 + a_2 h^4 + \cdots$. Maka $\dfrac{4\,D(h/2) - D(h)}{3}$ menghapus suku h^2.

Ulangi:

$$D_{i,j} = \frac{4^j D_{i,j-1} - D_{i-1,j-1}}{4^j - 1}, \quad j = 1, 2, \ldots, i \tag{9.7}$$

```
h = 0.800  0.251272405013
h = 0.400  0.250313875299  0.249994365394
h = 0.200  0.250078210575  0.249999655667  0.250000008352
h = 0.100  0.250019536593  0.249999978598  0.250000000127  0.249999999996
```

Persamaan (9.7): setiap kolom menghapus satu suku galat lagi. Dengan langkah tidak lebih kecil dari 0,1 — jauh di atas zona bahaya pembulatan — diagonalnya mencapai sepuluh digit benar: galat 1.27×10^{-3} pada tingkat 0, 5.63×10^{-6} pada tingkat 1, dan seterusnya. Gagasan yang sama menggerakkan integrasi Romberg di Bab 10. Buku: Subbab 9.5.

Penganalisis Perjalanan

Kode 9.1 Dari `speed.py`: percepatan dengan beda kedua pusat, dengan rumus satu sisi di ujung-ujung.

```
def accelerations(s, h):
    """Second derivative: central inside, four-point one-sided at the ends."""
    n = len(s)
    a = [0.0] * n
    a[0] = (2 * s[0] - 5 * s[1] + 4 * s[2] - s[3]) / (h * h)
    for i in range(1, n - 1):
        a[i] = (s[i + 1] - 2 * s[i] + s[i - 1]) / (h * h)
    a[n - 1] = (2 * s[n - 1] - 5 * s[n - 2] + 4 * s[n - 3] - s[n - 4]) / (h * h)
    return a
```

Kode 9.1: kecepatannya memakai Persamaan (9.3) di dalam dan Persamaan (9.5) di ujung; pengereman yang lebih keras daripada -3.0 m/s^2 *dilaporkan. Log* `data/trip.csv` *disiapkan agar sesuai dengan perjalanan persis seperti itu, sehingga kita tahu apa yang sungguh terjadi: sekitar 40 km/jam, kucing, bagian yang lebih lambat, dan berhenti pelan. Buku: Subbab 9.6.*

Menjalankannya: Tiga Alarm

```
  t     s(m)   v(m/s)  v(km/h)  a(m/s^2)
  0      0.0    -0.90     -3.2      3.90
  1      0.6     2.10      7.6      3.00
 ...
 17    159.5    10.25     36.9     -2.10
 18    168.7     7.45     26.8     -3.50  harsh braking
 19    174.4     3.90     14.0     -3.60  harsh braking
 20    176.5     2.15      7.7      0.10
 ...
 35    287.9     6.85     24.7     -3.10  harsh braking
 36    293.2     5.35     19.3      0.10
top speed 41.4 km/h at t = 11 s
```

Baris-baris ini diambil dari keluaran lengkap 41 baris (titik-titik menandai baris yang tidak ditampilkan di sini). ***Kecepatan awal negatif****, kecepatan puncak 41,4 km/jam yang tidak pernah dicapai motornya, percepatan yang berayun antara* -2.3 *dan* $+1.6$ m/s^2 *saat melaju tetap, dan* ***alarm palsu pada 35 s*** *— berhenti pelan tadi. Rumusnya benar; bacaannya berderau. Buku: Subbab 9.6.*

Mengapa Derau Merusak Turunan

WAWASAN KUNCI: Diferensiasi memperbesar derau. Dengan galat bacaan saling bebas bersimpangan baku σ, kecepatan membawa derau $\sqrt{2}\sigma/(2h)$ dan percepatan $\sqrt{6}\sigma/h^2$. Untuk $\sigma = 0.3$ m dan $h = 1$ s itu 0,73 m/s² — cukup untuk mendorong berhenti pelan melewati ambang alarm.

h yang lebih kecil bukan obatnya: itu membuat $1/h^2$ lebih besar. Obatnya adalah memakai **lebih banyak bacaan** per taksiran.

Variansi $s_{i+1} - 2s_i + s_{i-1}$ adalah $(1+4+1)\sigma^2$, dari situlah $\sqrt{6}$. Integrasi, bab berikutnya, adalah kebalikannya: ia merata-ratakan derau sampai hilang. Buku: Subbab 9.6.

Haluskan Sebelum Menurunkan

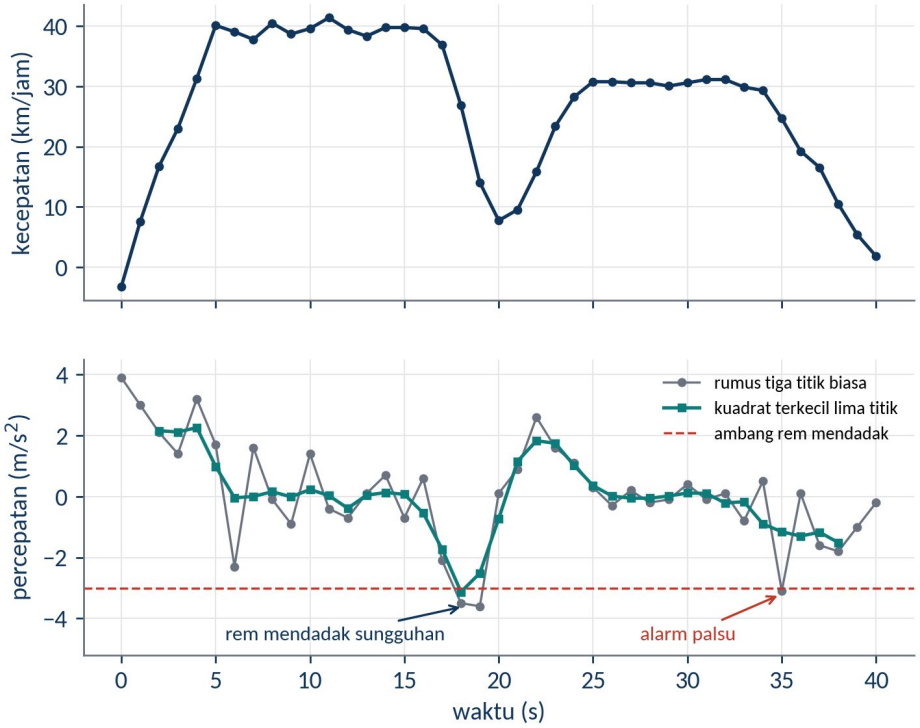
Cocokkan parabola kuadrat terkecil (Bab 8) melalui **lima** bacaan lalu turunkan parabolanya:

$$v \approx \frac{-2s_{-2} - s_{-1} + s_1 + 2s_2}{10h}, a \approx \frac{2s_{-2} - s_{-1} - 2s_0 - s_1 + 2s_2}{7h^2} \tag{9.8}$$

```
 17    10.25      9.43     -2.10     -1.73
 18     7.45      7.15     -3.50     -3.13
 19     3.90      4.62     -3.60     -2.51
 20     2.15      3.05      0.10     -0.73
 ...
 35     6.85      6.77     -3.10     -1.16
harsh braking, plain formulas:  t = 18, 19, 35
harsh braking, smoothed:        t = 18
```

Persamaan (9.8) adalah rumus jenis Savitzky–Golay. Kolom: waktu, kecepatan biasa dan terhaluskan, percepatan biasa dan terhaluskan. Rem keras tetap ditemukan pada 18 s dengan -3.13 *m/s², sedikit lebih lunak daripada* -3.5 *sejati karena pencocokan menyebarkannya; alarm palsu pada 35 s menjadi* -1.16*, dekat dengan* -1.4 *sejati. Saat melaju tetap percepatan terhaluskan tetap dalam* ± 0.4*. Buku: Subbab 9.7.*

Perjalanan dalam Satu Plot



Gambar 9.2 Kecepatan perjalanan Mas Dimas (atas) dan percepatannya (bawah) dari rumus tiga titik biasa dan dari rumus kuadrat terkecil lima titik; hanya kurva yang dihaluskan yang memisahkan rem mendadak sungguhan dari alarm palsu.

> **PERINGATAN:** Rumus penghalus memakai bacaan dari masa depan: pada $t = 18$ ia butuh bacaan pada 19 dan 20. Aplikasi yang harus memperingatkan *selagi* pengemudi mengerem hanya dapat memakai rumus satu sisi, yang lebih berderau, atau menerima sedikit tunda. Setiap turunan waktu-nyata menghadapi pilihan antara derau dan tunda ini.

Gambar 9.2: percepatan biasa berkelana melintasi ambang merah; yang terhaluskan hanya melintasinya di tempat rem sungguhan. Mas Dimas benar tentang kedua perjalanan itu. Buku: Subbab 9.7.

Memeriksa dengan Pustaka

```
NumPy 2.4.4, SciPy 1.17.1
largest speed difference, ours vs np.gradient: 0.0e+00 m/s
a at t = 19 s: ours -3.60, gradient twice -2.65
scipy derivative of sqrt at 4: 0.2500000000000000 (estimated error 8.9e-14, 11 evaluations)
```

`np.gradient` *dengan* `edge_order=2` *memakai rumus pusat dan satu sisi yang sama dan mereproduksi kecepatan kita dengan tepat. Memakainya dua kali untuk percepatan bukan beda kedua — rentangnya empat detik, bukan dua, dan memberi* -2.65 *alih-alih* -3.60 *pada 19 s.* `scipy.differentiate.derivative` *menurunkan fungsi secara adaptif, lengkap dengan taksiran galat. Uji* `test_diff.py` *dan* `DiffTest.java` *memeriksa fakta yang harus berlaku apa pun datanya: beda pusat eksak untuk parabola, galat beda maju terbagi dua bila* h *dibagi dua, dan untuk* $s = t^2$ *penganalisis memberi tepat* $v = 2t$ *dan* $a = 2$*. Buku: Subbab 9.8.*

Metode Numerik dalam Praktik: Profitina

Dasbor analitik historis Profitina memindai ulang situs setiap merek menurut jadwal, menyimpan skor per minggu, dan membunyikan **peringatan**.

Apakah skor saya membaik? Seberapa cepat? Apakah baru saja turun? adalah pertanyaan turunan tentang deret mingguan.

Satu pemindaian berderau (halaman yang kebetulan lambat merespons), sehingga peringatan dari dua minggu tunggal akan memberi alarm palsu persis seperti yang mengganggu Mas Dimas.

Jenis penyelesaiannya adalah Subbab 9.7: taksir kecenderungan dari pencocokan kuadrat terkecil atas beberapa minggu terakhir, terima bahwa perubahan nyata dikenali sedikit lebih lambat, dan jelaskan dengan kata-kata sederhana. Cara Profitina menghitung peringatannya adalah urusannya sendiri; bentuk masalahnya adalah bab ini. Angka memberi saran; pengguna yang memutuskan. Buku: Subbab 9.9.

Kesalahan Umum

Membuat h sekecil mungkin. Di bawah optimum pembulatan mengambil alih; pakai 10^{-8} untuk beda maju, 10^{-5} untuk pusat, atau Richardson.

Menurunkan data ukur mentah. Kenali deraunya dulu; haluskan dengan lebih banyak titik.

Memakai rutin turunan pertama dua kali untuk turunan kedua. Itu melebarkan stensil dan mengubah jawabannya.

Melupakan ujung-ujung. Rumus pusat butuh kedua tetangga; pakai rumus satu sisi dari Tabel 9.2.

Sebelum menurunkan log apa pun, ulangi pengukuran yang seharusnya tidak berubah — motor yang diparkir, tangki yang penuh — dan lihat seberapa goyah bacaannya. Buku: Subbab 9.10–9.11.

Ringkasan

1. Dari Taylor (Persamaan (9.1)): beda maju berorde satu (Persamaan (9.2)), beda pusat dan beda kedua berorde dua (Persamaan (9.3), Persamaan (9.4)).

2. Koefisien tak tentu memberi rumus untuk ujung-ujung (Persamaan (9.5)).

3. Galat pemotongan turun bersama h, galat pembulatan tumbuh seperti $1/h$; optimumnya (Persamaan (9.6)) sekitar 10^{-8} untuk maju, 10^{-5} untuk pusat.

4. Richardson (Persamaan (9.7)) mencapai ketelitian tinggi dengan langkah besar yang aman.

5. Derau ukur diperbesar $1/h$ dan $1/h^2$ kali; rumus kuadrat terkecil lima titik (Persamaan (9.8)) mempertahankan rem sungguhan dan menghapus alarm palsu.

`np.gradient` dengan `edge_order=2` mereproduksi kecepatan kita dengan tepat; `scipy.differentiate.derivative` menurunkan fungsi secara adaptif. Buku: Subbab 9.12.

Pertemuan Berikutnya

Bab 10 — Integrasi Numerik: Trapesium, Simpson, Romberg, dan Gauss. Dari laju kembali ke total.

Luas di bawah kurva dari sampel, orde galat, Richardson lagi sebagai Romberg, dan pilihan titik yang cerdik dari Gauss.

Sebelumnya: dari bacaan pada 34, 35, dan 36 s, hitung dengan tangan kecepatan dan percepatan biasa pada 35 s, dan jelaskan alarm palsunya dengan galat bacaan.

Soal itu juga memberi jarak sejatinya, sehingga Anda dapat menghitung galat setiap bacaan dan melihat dengan tepat bagaimana berhenti pelan menjadi -3.1 m/s^2*. Bacaan yang jaraknya tidak sama ditangani dalam penyelesaiannya. Buku: Subbab 9.13–9.14.*

Kuliah 10 — Integrasi Numerik: Trapesium, Simpson, Romberg, dan Gauss

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 10

Dosen: Irfan Subakti — yifana@gmail.com

Menjumlahkan banyak potongan kecil adalah salah satu hal tertua yang dilakukan manusia dengan bilangan: luas ladang dari pias-pias, hujan selama badai dari bacaan setiap sepuluh menit, energi yang dipakai mesin. Masing-masing adalah integral. Hari ini kita membangun aturan-aturan klasik, membuktikan orde galatnya, menaksir galat tanpa mengetahui kebenarannya, dan bertemu dua gagasan yang lebih cerdik: ekstrapolasi Romberg dan pilihan titik Gauss. Buku: Bab 10, Subbab 10.1–10.14.

Cakupan Pertemuan Ini

1. Masalahnya — luas kebun di antara jalan yang lurus dan sungai yang berkelok.

2. Trapesium dan Simpson — penurunan, orde galat, dan taksiran galat dari pembagian dua.

3. Romberg — ekstrapolasi Richardson terhadap aturan trapesium.

4. Kuadratur Gauss — memilih tempat mengukur, dan Simpson adaptif.

5. Memeriksa — orde yang diukur, dan alat integrasi SciPy terkini.

Diferensiasi memperbesar derau; integrasi merata-ratakannya. Itulah sebabnya integrasi lebih ramah daripada diferensiasi, dan aturan-aturannya dapat didorong sampai ketelitian yang sangat tinggi. Buku: Bab 10.

Seberapa Luas Kebunnya?

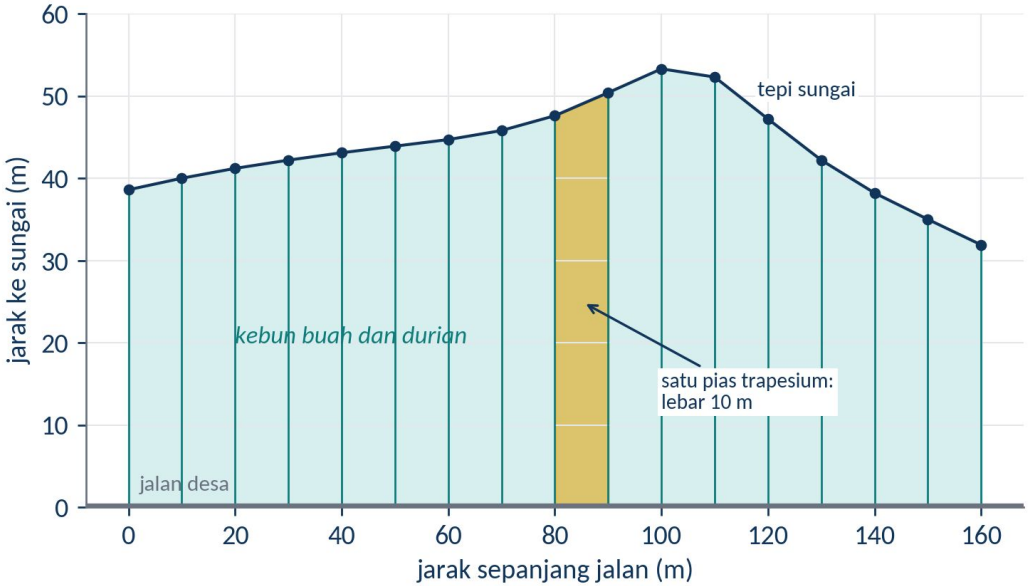
Keluarga Pak Harjono memiliki **kebun buah dan durian**: sebidang lahan memanjang di antara jalan desa yang lurus dan sungai kecil, **160 m** di sepanjang jalan.

Anto, anak yang meneruskan kebun, butuh luasnya: pohon durian berjarak sekitar 10 m, jadi berapa bibit, dan berapa kompos? Surat tanah lama juga mencantumkan luas yang tidak begitu dipercaya siapa pun di keluarga.

Dengan meteran 50 m ia mengukur, setiap 10 m, jarak tegak lurus dari jalan ke tepi sungai — **jarak tegak** (*offset*): 38,6 m di awal, 31,9 m di pojok terjauh, 17 jarak tegak seluruhnya.

Sebelum membayar juru ukur, Anto ingin tahu kira-kira berapa seharusnya jawabannya. Pengukuran resmi adalah wewenang juru ukur; hitungan ini untuk perencanaan. Buku: Subbab 10.1.

Gagasannya: Pias-pias



Gambar 10.1 Kebun di antara jalan desa yang lurus dan tepi sungai, dengan tujuh belas jarak tegak yang diukur setiap 10 m; pias yang disorot adalah salah satu dari enam belas trapesium dalam aturan trapesium.

Anggap setiap potongan tepi sungai lurus: setiap pias adalah trapesium. Yang pertama: $10 \times (38.6 + 40.0)/2 = 393.0\ \text{m}^2$.

Atau cocokkan parabola melalui tiga jarak tegak: dua pias pertama memberi $\frac{10}{3}(38.6+4\times40.0+41.2)=799.3\ \text{m}^2$.

Gambar 10.1: enam belas pias, masing-masing selebar 10 m. Menjumlahkan semua trapesium memberi bobot penuh pada setiap jarak tegak di dalam dan bobot setengah pada kedua ujung. Versi parabolanya adalah aturan Simpson. Kedua cara hanya memakai angka-angka yang sudah ada di buku catatan Anto; bedanya hanya pada bentuk potongan tepi sungai yang kita anggap di antara dua jarak tegak — lurus, atau melengkung mengikuti tiga titik. Buku: Subbab 10.2.

Aturan Trapesium dan Galatnya

Satu pias, dari Taylor di sekitar titik tengahnya:

$$\int_a^b f(x)\,dx - \frac{h}{2}(f(a)+f(b)) = \left(\frac{1}{24}-\frac{1}{8}\right)h^3 f''(m) + \cdots = -\frac{h^3}{12}f''(\xi) \tag{10.1}$$

Semua n pias, $n=(b-a)/h$:

$$T(h) = h\left(\frac{f_0}{2}+f_1+f_2+\cdots+f_{n-1}+\frac{f_n}{2}\right), I - T(h) = -\frac{(b-a)h^2}{12}f''(\xi) \tag{10.2}$$

Persamaan (10.1): setiap pias meleset sebesar h^3; ada $n \propto 1/h$ pias, jadi Persamaan (10.2) berorde dua. Tandanya mengikuti kelengkungan: di tempat tepi sungai menggembung keluar, puncak-puncak lurus memotong luasan dan trapesiumnya kekurangan. Buku: Subbab 10.3.1.

Rumus Euler–Maclaurin

$$T(h) - I = \frac{h^2}{12}\left(f'(b) - f'(a)\right) - \frac{h^4}{720}\left(f'''(b) - f'''(a)\right) + \cdots \tag{10.3}$$

Galatnya adalah deret dalam **pangkat genap** h, dengan koefisien yang tidak bergantung pada h.

Itu tepat keadaan ekstrapolasi Richardson di Bab 9.

Persamaan (10.3) punya akibat yang mengejutkan: untuk fungsi periodik yang mulus yang diintegralkan atas satu periode penuh, semua selisih turunan lenyap dan aturan trapesium konvergen lebih cepat daripada pangkat h mana pun. Rumus ini juga mendasari Simpson dan Romberg, yang menyusul. Buku: Subbab 10.3.2.

Aturan-aturan Simpson

Bobot untuk tiga titik yang mengintegralkan 1, x, dan x^2 secara eksak adalah $\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$ — dan, karena simetri, x^3 juga:

$$S(h)=\frac{h}{3}(f_0+4f_1+2f_2+4f_3+\cdots+2f_{n-2}+4f_{n-1}+f_n),\, I-S(h)=-\frac{(b-a)h^4}{180}f^{(4)}(\xi) \tag{10.4}$$

$$S(h)=\frac{4T(h)-T(2h)}{3} \tag{10.5}$$

Persamaan (10.4) butuh banyak pias genap dan berorde empat. Persamaan (10.5): aturan Simpson adalah satu langkah Richardson terhadap aturan trapesium. Untuk banyak pias ganjil, pakai aturan 3/8 pada tiga pias terakhir. Buku: Subbab 10.4.1–10.4.2.

Aturan 3/8 Simpson, dan Taksiran Galat Cuma-cuma

$$\int_{x_0}^{x_3} f(x)\,dx \approx \frac{3h}{8}(f_0+3f_1+3f_2+f_3),\text{error} = -\frac{3h^5}{80}f^{(4)}(\xi) \tag{10.6}$$

Bandingkan jawaban dengan semua titik dan jawaban dengan setiap titik kedua:

$$I - T(h) \approx \frac{T(h)-T(2h)}{3}, I - S(h) \approx \frac{S(h)-S(2h)}{15} \tag{10.7}$$

Persamaan (10.6) menangani banyak pias ganjil dengan orde yang sama. Persamaan (10.7) mengikuti orde galatnya: membagi dua h membagi galat trapesium dengan 4 dan galat Simpson dengan 16. Ia menaksir galat tanpa mengetahui kebenaran. Buku: Subbab 10.4.3 dan 10.5.

Kalkulator Kebun

Kode 10.1 Dari `orchard.py`: aturan trapesium dan Simpson gabungan untuk data berjarak sama.

```
def trapezoid(y, h):
    """Composite trapezoidal rule for equally spaced values y."""
    return h * (y[0] / 2 + sum(y[1:-1]) + y[-1] / 2)


def simpson(y, h):
    """Composite Simpson 1/3 rule; needs an even number of intervals."""
    n = len(y) - 1
    if n % 2:
        raise ValueError("Simpson's 1/3 rule needs an even number of intervals")
    odd = sum(y[1:n:2])                          # y1, y3, ..., y(n-1)
    even = sum(y[2:n:2])                         # y2, y4, ..., y(n-2)
    return h / 3 * (y[0] + 4 * odd + 2 * even + y[n])
```

Kode 10.1: irisan `y[1:n:2]` dan `y[2:n:2]` mengambil nilai dalam yang ganjil dan genap. Simpson menolak banyak pias ganjil alih-alih diam-diam mengembalikan luas yang salah. Berkas datanya disiapkan dari kurva sungai yang diketahui, sehingga luas sejatinya diketahui: 7026,8 m². Buku: Subbab 10.5.

Menjalankannya: Kebun

```
17 offsets, every 10 m along 160 m of road
rule        every 10 m  every 20 m  error estimate
trapezoid       7023.5      7011.0             4.2
Simpson         7027.7      7039.3            -0.8
area 7027 m2 = 0.70 ha, room for about 70 durian trees at 10 m x 10 m
```

Trapesium kurang 3,3 m^2 dan taksirannya berkata "tambahkan sekitar 4,2" — tandanya benar, besarnya benar. Simpson lebih 0,9 m^2 dan berkata "kurangi sekitar 0,8". Jawaban yang dikoreksi, 7026,9 m^2, berjarak kurang dari sepersepuluh meter persegi dari kebenaran — lebih baik daripada yang pantas didapat meteran itu. Setiap jarak tegak dibulatkan ke 0,1 m, yang sendirian sudah menggeser luas sekitar satu meter persegi. Buku: Subbab 10.5.

Dari Mana Galatnya Datang

Di sekitar kelokan, dari 100 sampai 120 m, jarak tegaknya 53,3, 52,3, dan 47,2 m.

Trapesium: 1025,5 m^2. Simpson: 1032,3 m^2. Sejati: 1031,3 m^2.

Tepi sungai menggembung keluar; puncak-puncak lurus memotong hampir enam meter persegi, parabola mengikuti gembungan itu sampai kurang dari satu.

> **WAWASAN KUNCI:** Bagi Anto jawabannya 0,70 hektare, lebih kurang beberapa meter persegi. Jika surat tanahnya menyebut 0,6 ha, mintalah juru ukur memeriksa dengan cermat; jika menyebut 0,7, keluarga boleh tenang.

Galat aturan dan galat pengukuran kini sama besarnya. Mendorong aturannya lebih jauh tidak ada gunanya; meteran yang lebih baik justru berguna. Buku: Subbab 10.5.

Memeriksa Ordenya

```
   n   trapezoid error  ratio    Simpson error  ratio
   2        1.519e-02              1.297e-03
   4        3.877e-03     3.9      1.068e-04    12.1
   8        9.747e-04     4.0      7.350e-06    14.5
  16        2.440e-04     4.0      4.723e-07    15.6
  32        6.103e-05     4.0      2.973e-08    15.9
  64        1.526e-05     4.0      1.862e-09    16.0
```

`quadrature.py` dan `Quadrature.java` mengintegralkan $1/(1+x)$ dari 0 sampai 1, yang nilainya, $\ln 2$, kita ketahui. Menggandakan banyak pias membagi galat trapesium dengan 4 dan galat Simpson dengan 16: orde 2 dan 4, terukur. Rasio Simpson mula-mula 12,1 lalu naik mendekati 16, karena orde empat baru tampak jelas setelah langkahnya cukup kecil; rasio trapesium sudah 4,0 sejak awal. Menguji orde pada integral yang jawabannya diketahui adalah cara termurah untuk memastikan program tidak diam-diam salah. Buku: Subbab 10.6.

Integrasi Romberg

Dengan $R_{i,0}$ aturan trapesium pada 2^i pias, tabel Richardson berlaku tanpa perubahan:

$$R_{i,j} = \frac{4^j R_{i,j-1} - R_{i-1,j-1}}{4^j - 1}, \; j = 1, \ldots, i \tag{10.8}$$

```
n =  1  0.750000000000
n =  2  0.708333333333  0.694444444444
n =  4  0.697023809524  0.693253968254  0.693174603175
n =  8  0.694121850372  0.693154530655  0.693147901481  0.693147477645
n = 16  0.693391202208  0.693147652819  0.693147194297  0.693147183072  0.693147181917
R[4][4] = 0.693147181916745, error 1.4e-09, 17 function values
```

Persamaan (10.8): kolom 1 adalah Simpson; kolom 2 aturan Boole; kolom berikutnya terus menghapus galat. Setiap baris baru memakai ulang semua nilai fungsi lama dan hanya menambah titik-titik tengah yang baru. Tujuh belas nilai memberi galat 1.4×10^{-9}*, sedangkan aturan trapesium sendirian akan butuh ribuan. Buku: Subbab 10.7.*

Kuadratur Gauss: Memilih Tempat Mengukur

Bila kita boleh memilih titik sekaligus bobotnya, dua titik mengintegralkan kubik secara eksak:

$$\int_{-1}^{1} f(x)\,dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \tag{10.9}$$

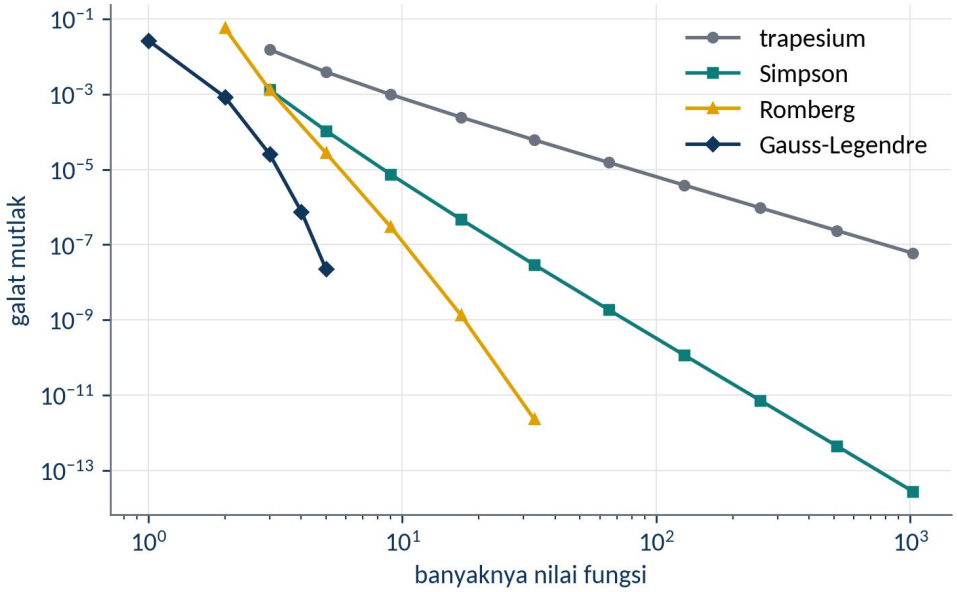
n titik di akar-akar polinomial Legendre mengintegralkan derajat $2n-1$ secara eksak. Selang mana pun dipetakan ke $[-1,1]$:

$$\int_{a}^{b} f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right)dt \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{a+b}{2} + \frac{b-a}{2}t_i\right) \tag{10.10}$$

Persamaan (10.9): derajat dua kali aturan trapesium dari banyak titik yang sama. Persamaan (10.10) menskalakan ulang simpul dan bobot ke $[a,b]$. Syaratnya: Gauss butuh f di titik-titik miliknya, jadi ia bekerja untuk rumus, bukan untuk jarak tegak Anto. Buku: Subbab 10.8.

Gauss Melawan yang Lain

points	Gauss-Legendre	error	trapezoid error	Simpson error
1	0.666666666667	2.6e-02		
2	0.692307692308	8.4e-04	5.7e-02	
3	0.693121693122	2.5e-05	1.5e-02	1.3e-03
4	0.693146417445	7.6e-07	6.9e-03	
5	0.693147157853	2.3e-08	3.9e-03	1.1e-04



Gambar 10.2 Galat terhadap banyaknya nilai fungsi untuk $\int_0^1 dx/(1+x) = \ln 2$: trapesium (orde 2), Simpson (orde 4), Romberg, dan Gauss–Legendre.

Dengan lima nilai Gauss–Legendre mencapai 2.3×10^{-8}; trapesium di 3.9×10^{-3} dan Simpson di 1.1×10^{-4}. Gambar 10.2: per nilai fungsi, Gauss yang terbaik, Romberg dekat di belakangnya, dan keduanya jauh meninggalkan aturan berorde tetap. Buku: Subbab 10.8.

Membuatnya Lebih Baik: Simpson Adaptif

Langkah seragam memboroskan tenaga. Hitung Simpson pada satu selang dan pada kedua separuhnya; jika sepakat dalam toleransi, terima; jika tidak, belah dan ulangi secara rekursif.

```
tolerance   adaptive result  error    function values
    1e-04   0.666663972068   2.7e-06        37
    1e-06   0.666666647130   2.0e-08       105
    1e-08   0.666666666641   2.6e-11       317
uniform Simpson needs, for an error below 1e-8:
  n = 65536 intervals, error 4.8e-09, 65537 function values
```

Integrannya $\sqrt{x}$ pada $[0,1]$, yang kemiringannya tak hingga di 0 merusak orde galat aturan seragam. Aturan adaptif menghabiskan langkah kecilnya dekat 0 dan langkah besar di tempat lain: 317 nilai fungsi, sedangkan Simpson seragam butuh 65.537. Buku: Subbab 10.9.

Memeriksa dengan Pustaka

```
NumPy 2.4.4, SciPy 1.17.1
orchard, trapezoid: 7023.5 m2
orchard, simpson:   7027.7 m2
orchard, romb:      7026.6 m2
quad 1/(1+x):   0.693147180559945 (estimated error 7.7e-15)
fixed_quad n=5: 0.693147157853
quad sqrt(x):   0.666666666666667 (231 function values)
```

Untuk data, `trapezoid`*,* `simpson`*, dan* `romb` *menerima larik;* `romb` *butuh* 2^k+1 *nilai berjarak sama, dan 17 jarak tegak Anto kebetulan memenuhinya; hasilnya 7026,6 m², tetapi kolom ekstrapolasi yang tinggi kurang dapat dipercaya pada data hasil ukur. Untuk fungsi,* `quad` *adalah Gauss–Kronrod adaptif dari QUADPACK dengan taksiran galat, dan* `fixed_quad` *adalah Gauss lima titik kita.* `np.trapz` *dan* `integrate.romberg` *sudah tidak ada dalam rilis terkini. Buku: Subbab 10.10.*

Metode Numerik dalam Praktik: Profitina

“Berapa **skor rata-rata** saya kuartal ini?” Rata-rata terhadap waktu adalah integral dibagi panjang periodenya.

Dengan pemindaian mingguan berjarak sama, aturan trapesium memberi bobot setengah pada minggu pertama dan terakhir; dengan pemindaian tambahan menjelang kampanye, trapesium pada tanggal sebenarnya mencegah minggu-minggu sibuk terlalu berbobot.

Emisi PROFIT yang meluruh: total yang dikeluarkan selama suatu periode adalah integral laju emisi — trapesium untuk tabel, Gauss atau aturan adaptif untuk rumus yang mulus.

“Total perbaikan selama kampanye” adalah luas antara kurva skor dan tingkat awalnya. Parameter emisi dan bagian dalam oracle adalah milik Profitina sendiri; matematikanya adalah bab ini. Buku: Subbab 10.11.

Kesalahan Umum

Simpson dengan banyak pias ganjil. Pakai aturan 3/8 untuk sisa yang ganjil.

Lupa separuh bobot di ujung-ujung. Jumlah biasa dikali h adalah aturan persegi panjang, hanya berorde satu.

Jarak tidak sama dengan rumus berjarak sama. Integralkan setiap pias dengan lebarnya sendiri.

Ekstrapolasi orde tinggi pada data berderau. Pada nilai hasil ukur, hentikan Romberg setelah satu atau dua kolom.

Mengharapkan Gauss memakai data Anda. Ia butuh nilai di titik-titiknya sendiri.

Di lapangan: ukur jarak tegak dengan siku, berjarak sama, dengan banyak pias genap; maka Simpson langsung berlaku dan taksiran galatnya cuma-cuma. Buku: Subbab 10.12–10.13.

Ringkasan

1. Trapesium: garis, galat $O(h^2)$ (Persamaan (10.2)). Simpson: parabola, eksak untuk kubik, $O(h^4)$ (Persamaan (10.4)); aturan 3/8 untuk pias ganjil.

2. Euler–Maclaurin (Persamaan (10.3)): galat trapesium adalah deret dalam h^2; Richardson memberi Simpson (Persamaan (10.5)) dan Romberg (Persamaan (10.8)).

3. Pembagian dua menaksir galat tanpa kebenaran (Persamaan (10.7)): 7026,9 m^2 dibanding 7026,8 m^2 sejati.

4. Gauss–Legendre memilih titik dan bobot, eksak sampai derajat $2n-1$ (Persamaan (10.9), Persamaan (10.10)); ia butuh fungsi, bukan data.

5. Simpson adaptif menaruh langkah kecil hanya di tempat yang perlu; SciPy menyediakan semuanya.

Integrasi merata-ratakan galat; itulah sebabnya kebun yang diukur dengan tangan keluar dengan meleset kurang dari sepersepuluh meter persegi. Buku: Subbab 10.14.

Pertemuan Berikutnya

Bab 11 — Persamaan Diferensial Biasa: dari Euler ke Runge-Kutta. Bila lajunya diketahui dan keadaannya harus dicari, langkah demi langkah.

Metode Euler, galatnya, Heun dan Runge–Kutta orde empat klasik, sistem persamaan, dan kestabilan.

Sebelumnya: 40 m pertama kebun dengan tangan, trapesium dan Simpson, dengan taksiran galat dari selisihnya.

Jarak tegaknya 38,6, 40,0, 41,2, 42,2, dan 43,1 m; empat pias, jadi Simpson berlaku. Penyelesaiannya ada di buku. Buku: Subbab 10.15–10.16.

Kuliah 11 — Persamaan Diferensial Biasa: dari Euler ke Runge-Kutta

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 11

Dosen: Irfan Subakti — yifana@gmail.com

Bila laju perubahan suatu besaran bergantung pada besaran itu sendiri, kita punya persamaan diferensial. Kebanyakan tidak dapat diselesaikan dengan rumus, tetapi semuanya dapat dilangkahkan maju dalam waktu: dari keadaan sekarang, keadaan sesaat kemudian, dan seterusnya. Hari ini: metode Euler, keluarga Runge–Kutta, satu pagi irigasi sungguhan dengan sakelar dan luapan, sebuah sistem yang memperlihatkan wujud ketidakstabilan, dan pemecah adaptif SciPy. Buku: Bab 11, Subbab 11.1–11.13.

Cakupan Pertemuan Ini

1. **Masalahnya** — tandon irigasi tetes: kapan pompa harus dimatikan?
2. **Modelnya** — neraca massa, masalah nilai awal, dan satu solusi eksak untuk menguji.
3. **Euler dan Runge–Kutta** — satu garis singgung per langkah, atau beberapa kemiringan per langkah; orde 1, 2, dan 4, terukur.
4. **Simulator pagi** — sakelar, luapan, tandon kering, dan biseksi yang membungkus sebuah simulasi.
5. **Sistem dan kestabilan** — jok sepeda motor setelah polisi tidur; dan `solve_ivp`.

Setiap program dalam Java dan Python mencetak angka yang sama; fungsi-fungsi langkahnya dapat dipertukarkan, sehingga satu penggerak membandingkan semua metode. Buku: Bab 11.

Cukupkah Tandonnya sampai Pukul Sepuluh?

Musim kemarau di kebun durian Bab 10. Anto menanam **tujuh puluh pohon durian muda** dan membangun irigasi tetes gravitasi dari **tandon 1.000 liter** di puncak lereng.

Setiap pagi: pompa menyala pukul 06.00, katup tetes dibuka pukul 06.30, pompa dimatikan “kalau ingat”.

Pada beberapa pagi tandon **meluap** — air sumur yang bersih dan listrik yang memompanya terbuang. Pada pagi lain tandon **kering** sebelum pukul sepuluh, dan pohon-pohon di ujung mendapat terlalu sedikit.

Kapan ia harus mematikan pompa?

Ia dapat mencari tahu dengan mencoba, satu pagi demi satu pagi. Tetapi aliran keluar lewat penetes bergantung pada tinggi air: laju perubahan ketinggian bergantung pada ketinggian itu sendiri. Itulah persamaan diferensial. Buku: Subbab 11.1.

Gagasannya dalam Kata-kata Sederhana

Luas dasar tandon satu meter persegi. Saat penuh, ia meneteskan sekitar **10 liter per menit**, yang menurunkan ketinggian satu sentimeter per menit.

Anggap alirannya tetap 10 L/menit selama sepuluh menit: ketinggian turun 10 cm, menjadi 90 cm. Tetapi alirannya melambat ketika ketinggian turun, jadi penurunan sebenarnya sedikit lebih kecil.

Hitung ulang lajunya pada ketinggian baru, ambil langkah berikutnya, dan ulangi. Itulah **metode Euler**.

Setiap metode hari ini adalah jawaban yang lebih baik atas pertanyaan yang sama: kemiringan mana yang harus dipakai satu langkah? Buku: Subbab 11.2.

Model dan Solusi Eksaknya

Neraca massa, dengan luas dasar A, ketinggian h, aliran masuk dari pompa dan aliran keluar Torricelli lewat penetes:

$$A\frac{dh}{dt} = Q_{\text{in}}(t) - k\sqrt{h} \tag{11.1}$$

Masalah nilai awal yang umum:

$$y'(t) = f(t, y(t)), y(t_0) = y_0 \tag{11.2}$$

Dengan pompa mati, memisahkan peubah memberi

$$h(t) = \left(\sqrt{h_0} - \frac{kt}{2A}\right)^2, 0 \le t \le \frac{2A\sqrt{h_0}}{k} \tag{11.3}$$

Persamaan (11.1) berkata: laju perubahan persediaan sama dengan yang masuk dikurangi yang keluar. Akar kuadratnya berasal dari hukum Torricelli: air yang keluar dari lubang di bawah tinggi h melaju sekitar $\sqrt{2gh}$. Persamaan (11.2) adalah bentuk yang diterima setiap pemecah. Persamaan (11.3) adalah tongkat ukur kita: untuk tandon Anto ia berbunyi $h = (1 - 0.005t)^2$ dengan t dalam menit. Buku: Subbab 11.3.

Metode Euler

Teorema Taylor dengan turunan diganti f, dan sisanya dibuang:

$$y_{n+1} = y_n + h f(t_n, y_n) \tag{11.4}$$

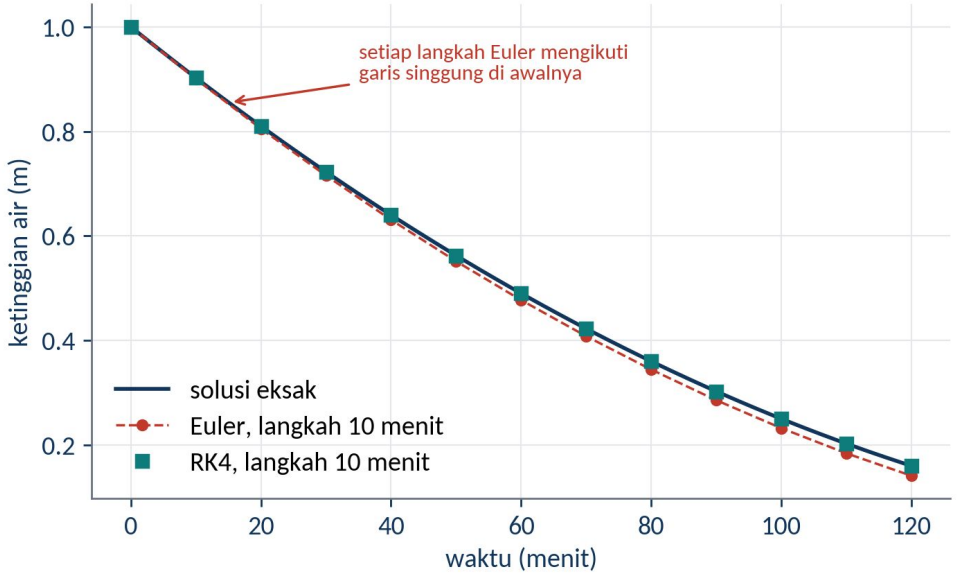
Setiap langkah meleset $O(h^2)$; ada $O(1/h)$ langkah, dan galatnya dapat tumbuh di sepanjang jalan:

$$\left| y(t_n) - y_n \right| \le \frac{hM}{2L}\left(e^{L(t_n - t_0)} - 1\right) \tag{11.5}$$

Persamaan (11.4): ikuti garis singgung untuk satu langkah. Persamaan (11.5): galat globalnya berorde satu dalam h; M membatasi y'' dan L adalah konstanta Lipschitz f, yang menyatakan seberapa cepat solusi-solusi yang berdekatan saling menjauh. Untuk tandon yang mengosong justru sebaliknya, sehingga batas ini pesimistis. Buku: Subbab 11.4.

Menjalankannya: Euler pada Tandon yang Mengosong

```
  t(min)   Euler h(m)   exact h(m)     error
       0     1.000000     1.000000  0.000000
      10     0.900000     0.902500 -0.002500
      20     0.805132     0.810000 -0.004868
      30     0.715403     0.722500 -0.007097
      40     0.630821     0.640000 -0.009179
      50     0.551397     0.562500 -0.011103
```



Gambar 11.1 Tandon yang mengosong: solusi eksak, metode Euler dengan langkah 10 menit (setiap langkah mengikuti garis singgung di awalnya dan mengeluarkan air sedikit terlalu banyak), dan metode Runge–Kutta klasik dengan langkah yang sama.

`tank_euler.py` dan `TankEuler.java`. *Langkah pertama adalah hitungan tangan kita: 0,90 dibanding 0,9025 yang eksak. Gambar 11.1: Euler selalu memakai kemiringan di awal langkah, tempat aliran paling besar, sehingga setiap kali ia mengeluarkan air sedikit terlalu banyak dan galatnya menumpuk. Buku: Subbab 11.4.*

Runge–Kutta: Ambil Kemiringan Beberapa Kali

Dua tahap, dengan parameter bebas yang dipilih dengan mencocokkan deret Taylor sampai orde dua:

$$k_1 = f(t_n, y_n),\; k_2 = f(t_n + c\,h,\; y_n + a\,h\,k_1),\; y_{n+1} = y_n + h(b_1 k_1 + b_2 k_2) \tag{11.6}$$

Syaratnya $b_1 + b_2 = 1$ dan $b_2 c = b_2 a = \frac{1}{2}$. **Heun**: $b_1 = b_2 = \frac{1}{2}$, $c = 1$. **Titik tengah**: $b_2 = 1$, $c = \frac{1}{2}$.

Persamaan (11.6): Heun merata-ratakan kemiringan di awal dan di ramalan Euler; titik tengah memakai kemiringan di tengah jalan. Keduanya berorde dua: satu evaluasi kemiringan tambahan per langkah, dan galatnya turun bersama h^2. Buku: Subbab 11.5.1.

Runge–Kutta Orde Empat Klasik

$$
\begin{aligned}
k_1 &= f(t_n, y_n), & k_2 &= f\left(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_2\right), & k_4 &= f(t_n + h,\ y_n + hk_3), \\
y_{n+1} &= y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)
\end{aligned}
\tag{11.7}
$$

Kode 11.1 Dari `ode_orders.py`: satu langkah RK4, dapat dipertukarkan dengan langkah Euler, Heun, dan titik tengah.

```
def rk4_step(f, t, y, h):
    k1 = f(t, y)
    k2 = f(t + h / 2, y + h / 2 * k1)
    k3 = f(t + h / 2, y + h / 2 * k2)
    k4 = f(t + h, y + h * k3)
    return y + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

Persamaan (11.7): empat kemiringan — awal, dua di tengah, akhir — berbobot 1, 2, 2, 1. Bila f hanya bergantung pada t, ini tepat aturan Simpson dari Bab 10. Kode 11.1: sebuah langkah adalah fungsi, sehingga satu penggerak `solve` menjalankan setiap metode. Buku: Subbab 11.5.2.

Orde-ordenya, Terukur

```
Euler:
  step 10.00   error 1.286e-02
  step  5.00   error 6.335e-03  ratio   2.03
  step  2.50   error 3.144e-03  ratio   2.01
Heun:
  step 10.00   error 3.988e-04
  step  5.00   error 9.666e-05  ratio   4.13
  step  2.50   error 2.380e-05  ratio   4.06
RK4:
  step 10.00   error 1.535e-07
  step  5.00   error 9.345e-09  ratio  16.42
  step  2.50   error 5.762e-10  ratio  16.22
```

Galat pada menit ke-60 untuk langkah yang dibagi dua (program juga mencetak titik tengah dan ukuran langkah keempat). Rasio 2, 4, dan 16: orde 1, 2, dan 4. Dengan enam langkah 10 menit, galat RK4 seperenam mikrometer untuk 24 evaluasi kemiringan; Euler dengan 48 evaluasi masih meleset 1,6 mm — sepuluh ribu kali lebih buruk untuk dua kali pekerjaan. Buku: Subbab 11.5.3.

Simulator Pagi

Pukul 06.00 tandon berisi 25 cm. Pompa menambah 20 L/menit sampai dimatikan; katup tetes dibuka pukul 06.30.

Di atas 1 m, air keluar lewat pipa luapan; di bawah 1 cm penetes berhenti.

RK4 dengan langkah satu menit, dan setiap langkah **dipotong tepat di sebuah sakelar**, sehingga tidak ada langkah yang mengangkangi perubahan mendadak.

> **PERINGATAN:** Ketelitian Runge–Kutta mengandaikan ruas kanan yang mulus di dalam setiap langkah. Langkah yang mengangkangi sakelar pompa mencampur dua persamaan yang berbeda dan kehilangan ordenya. Potong langkahnya di setiap peristiwa.

Simulatornya adalah Persamaan (11.1) dengan suku pompa yang mati, suku tetes yang hidup, dan dua batas. Buku: Subbab 11.6.

Menjalankannya: Dua Pagi

```
pump off at 07:30, drip valve open from 06:30
 time   level(cm)  liters
06:00       25.0     250
06:30       85.0     850
07:00      100.0    1000
07:30      100.0    1000
08:00       72.3     723
...
highest level 100.0 cm, spilled 454 liters
drippers stop at 10:31
check with half the step: highest 100.0 cm, spilled 455 liters, dry at 10:30
```

Baris-baris antara 08.00 dan ringkasan tidak ditampilkan di sini. Dengan pompa menyala sampai 07.30, tandon penuh sekitar pukul 06.44 dan ***454 liter*** *meluap. Solusi eksak memeriksa waktu keringnya: dari penuh pukul 07.30,* $1 - 0.005\,s = 0.1$ *memberi 180 menit, pukul 10.30. Dengan pompa dimatikan pukul 06.45, hanya 5 liter yang meluap, tetapi penetes berhenti pukul 09.46. Buku: Subbab 11.6.*

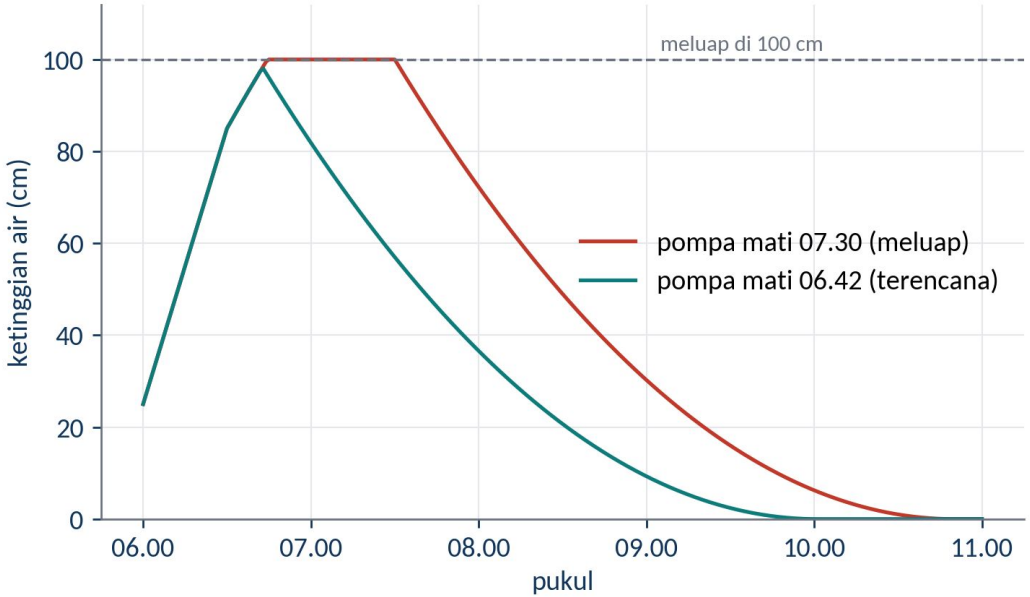
Merencanakan Pompa: Biseksi Membungkus Simulasi

Pertanyaan Anto sebenarnya terbalik: *waktu mati pompa T mana yang mengisi tandon sampai 98 cm dan tidak lebih?* Cari akar $g(T) = \text{top}(T) - 0.98$ — biseksi Bab 2, dengan seluruh simulasi sebagai fungsinya.

```
 step   pump off   highest level(cm)
    1     60.000        100.00
    2     45.000        100.00
    3     37.500         92.44
    4     41.250         96.52
    5     43.125         98.54
...
   12     42.729         98.00
switch the pump off after 42.7 min, at about 06:42: top 98.0 cm, spilled 0 liters, drippers stop at
09:41
```

`pump_plan.py` dan `PumpPlan.java`. *Dua belas pembagian dua membawa selangnya ke 0,015 menit, jauh lebih halus daripada ketelitian siapa pun mematikan pompa. Memotong langkah di sakelar penting di sini: tanpanya g akan bergerak meloncat-loncat dan biseksi tidak dapat menemukan akarnya. Buku: Subbab 11.7.*

Pagi yang Direncanakan



Gambar 11.2 Ketinggian air pada dua pagi: dengan pompa dibiarkan menyala sampai 07.30 tandon meluap selama tiga perempat jam; dengan pompa dimatikan pukul 06.42, seperti direncanakan dengan biseksi, ketinggiannya memuncak di 98 cm dan tidak pernah meluap.

Gambar 11.2: rencana itu menghindari pemborosan tetapi mengakhiri penyiraman pukul 09.41. Jalan keluarnya adalah pompa yang mengalirkan lebih sedikit untuk lebih lama, misalnya dengan menutup sebagian katup di keluarannya; satu soal latihan mencari aliran itu dengan perencana yang sama. Buku: Subbab 11.7.

Sistem: Jok Sepeda Motor

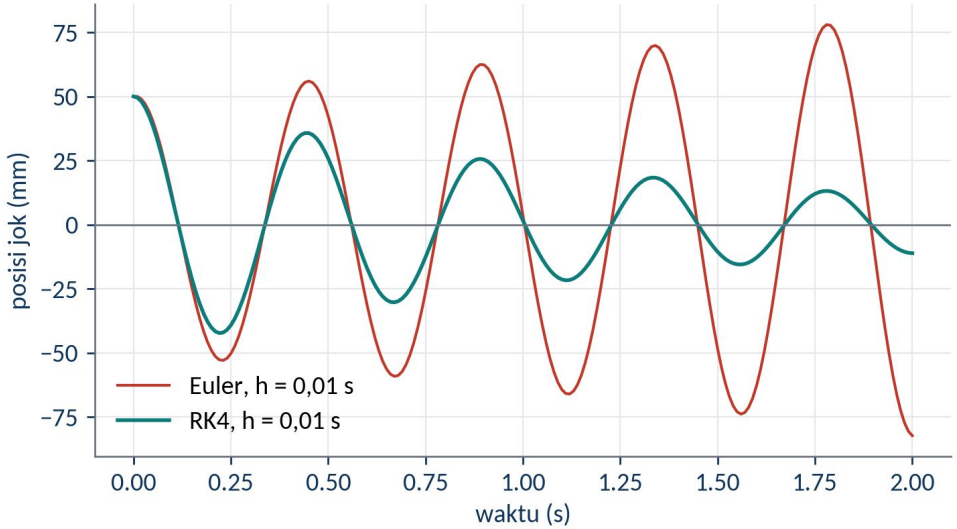
Jok pada pegas dan peredam kejut setelah polisi tidur: $m\,x'' + c\,x' + k\,x = 0$. Perkenalkan kecepatan v:

$$x' = v\,,\; v' = -\frac{c\,v + k\,x}{m} \tag{11.8}$$

seat position (mm) at t =	0.5 s	1.0 s	1.5 s	2.0 s
good shock, Euler h=0.01	6.401	0.509	0.003	-0.007
good shock, RK4 h=0.01	4.386	0.344	0.024	0.001
worn shock, Euler h=0.01	42.236	3.606	-48.413	-82.305
worn shock, RK4 h=0.01	25.762	1.608	-10.594	-11.134
worn shock, RK4 h=0.001	25.761	1.607	-10.595	-11.134

Persamaan (11.8): persamaan orde berapa pun menjadi sistem orde satu; Euler dan RK4 berlaku tanpa perubahan, dengan vektor. Dengan peredam aus, ayunan Euler membesar sampai 82 mm, sedangkan RK4, yang dibenarkan langkah sepuluh kali lebih kecil, memperlihatkannya meredam. Buku: Subbab 11.8.1.

Mengapa Euler Meledak: Kestabilan



Gambar 11.3 Jok dengan peredam kejut yang aus setelah polisi tidur: RK4 memperlihatkan ayunan yang meredam, sedangkan metode Euler dengan langkah yang sama membuatnya membesar, sebuah ketidakstabilan metode, bukan ketidakstabilan sepeda motornya.

Pada persamaan uji $y' = \lambda y$, Euler mengalikan dengan $1 + h\lambda$ setiap langkah. Ia meluruh hanya jika

$$|1 + h\lambda| \le 1 \tag{11.9}$$

*Gambar 11.3: langkah yang sama, dua metode, perilaku yang berlawanan. Persamaan (11.9) adalah cakram berjari-jari 1 yang berpusat di -1. Untuk jok ia mensyaratkan $h \le 2\zeta/\omega$: 0,05 s untuk peredam yang baik, **0,0075 s** untuk yang aus, sehingga $h = 0.01$ s berada di luar, dan setiap langkah mengalikan ayunan sekitar 1,0025 kali. Osilasi yang sedikit teredam adalah kasus terburuk bagi Euler. Buku: Subbab 11.8.2.*

Masalah Kaku, dan Menaksir Galat

> **WAWASAN KUNCI:** Metode yang tidak stabil menghasilkan angka yang mulus dan tampak masuk akal tetapi salah. Bila simulasi melakukan hal yang mustahil secara fisika — suspensi yang makin memantul, ketinggian yang menjadi negatif — curigai langkahnya sebelum fisikanya, dan bagi dua langkahnya.

Membagi dua langkah menaksir galat metode berorde p:

$$y(t) - y_{h/2} \approx \frac{y_{h/2} - y_h}{2^p - 1} \tag{11.10}$$

*Masalah dengan proses yang sangat cepat dan sangat lambat sekaligus disebut **kaku** (stiff): kestabilan memaksa metode eksplisit memakai langkah kecil lama setelah proses cepatnya lenyap. Metode implisit seperti Euler mundur menyelesaikan sebuah persamaan di setiap langkah dan tetap stabil. Persamaan (11.10) adalah taksiran galat Bab 10 untuk PDB. Buku: Subbab 11.8.2 dan 11.9.*

Memeriksa dengan Pustaka

```
NumPy 2.4.4, SciPy 1.17.1
tank, RK45 rtol=1e-03: h(60) = 0.490000035, error 3.5e-08, 26 slope evaluations
tank, RK45 rtol=1e-06: h(60) = 0.490000020, error 2.0e-08, 38 slope evaluations
tank, RK45 rtol=1e-09: h(60) = 0.490000000, error 5.7e-11, 68 slope evaluations
worn shock, DOP853, seat position (mm): 25.761  1.607  -10.595  -11.134
```

Metode bawaan `solve_ivp`, `"RK45"`, adalah pasangan Dormand–Prince: rumus orde lima dan orde empat yang berbagi kemiringan, yang selisihnya menaksir galat dan menentukan langkah berikutnya secara otomatis. `DOP853` mereproduksi jok RK4 kita sampai digit terakhir yang dicetak; `"Radau"` dan `"BDF"` adalah metode implisit untuk masalah kaku. Buku: Subbab 11.9.

Metode Numerik dalam Praktik: Profitina

Emisi PROFIT yang meluruh: lajunya menuruti $E' = -\lambda E$, dan pasokannya menuruti $S' = E(t) - B(t)$, emisi dikurangi pembakaran. Dengan pembakaran nyata yang tidak teratur, ia dilangkahkan maju seperti tandon — peristiwa tidak dikangkangi, proyeksi diperiksa dengan membagi dua langkah.

Rata-rata bergerak eksponensial yang akrab, $y_{n+1} = y_n + \alpha(x_n - y_n)$, adalah metode Euler pada tunda orde satu dengan $\alpha = h/\tau$.

Dilihat begitu, ia stabil dan mulus untuk α antara 0 dan 1, dan akan berosilasi lalu meledak di atas 2 — Persamaan (11.9) dengan $\lambda = -1/\tau$.

Tampilan terhaluskan seperti riwayat skor mingguan atau kurs PROFIT↔IDR mengikuti nilai mentah dengan tunda semacam ini. Cara Profitina menghitung angkanya sendiri adalah urusannya; bentuk matematikanya adalah bab ini. Buku: Subbab 11.10.

Kesalahan Umum

Memercayai Euler karena langkahnya kecil. Ketelitian mahal pada orde satu, dan osilasi dapat tidak stabil. Pakai RK4 atau pemecah adaptif.

Mengangkangi peristiwa. Potong langkah di setiap sakelar.

Tidak pernah memeriksa. Bandingkan dengan kasus eksak, atau bagi dua langkahnya.

Satuan yang salah. Tulis neracanya dengan kata-kata, lalu dengan simbol, dan periksa satuan setiap suku.

Kebanyakan PDB praktis adalah neraca — air di tandon, uang di rekening, stok di gudang, muatan di baterai, orang dalam antrean. Buku: Subbab 11.11–11.12.

Ringkasan

1. Masalah nilai awal (Persamaan (11.2)) memberi laju dari waktu dan keadaan; tandon (Persamaan (11.1)) punya solusi eksak (Persamaan (11.3)) untuk menguji.

2. Euler (Persamaan (11.4)) berorde satu (Persamaan (11.5)); metode dua tahap (Persamaan (11.6)) berorde dua; RK4 (Persamaan (11.7)) berorde empat.

3. Simulator memotong langkah di sakelar; biseksi yang membungkusnya menemukan waktu mati pompa, 06.42.

4. Persamaan orde dua menjadi sistem (Persamaan (11.8)); Euler tidak stabil pada peredam aus (Persamaan (11.9)), RK4 teliti.

5. Membagi dua langkah menaksir galat (Persamaan (11.10)); `solve_ivp` melakukannya secara otomatis.

Melangkah maju, memeriksa dengan membagi dua, menghormati peristiwa, dan mewaspadai ketidakstabilan: begitulah setiap simulasi di rekayasa dan sains dijaga tetap jujur. Buku: Subbab 11.13.

Pertemuan Berikutnya

Bab 12 — Nilai Eigen: Metode Pangkat dan Kegunaannya. Halaman mana yang paling penting? Ke keadaan mana sebuah sistem mengendap?

Metode pangkat, iterasi invers dan bergeser, dan PageRank.

Sebelumnya: dua langkah Heun 10 menit dengan tangan untuk tandon yang mengosong, dibandingkan dengan ketinggian eksak dan dengan Euler.

Penyelesaiannya juga menurunkan metode Euler semi-implisit, yang tetap stabil pada peredam aus dengan langkah yang sama. Buku: Subbab 11.14–11.15.

Kuliah 12 — Nilai Eigen: Metode Pangkat dan Kegunaannya

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 12

Dosen: Irfan Subakti — yifana@gmail.com

Ada pertanyaan yang bukan tentang saat tertentu, melainkan tentang apa yang dilakukan sebuah sistem dalam jangka panjang atau menurut sifatnya: pangsa pelanggan yang akhirnya dimiliki setiap toko, frekuensi yang disukai sebuah bangunan untuk berayun, halaman yang paling sering disinggahi pembaca acak. Setiap jawabannya adalah nilai eigen atau vektor eigen. Hari ini kita mencarinya dengan gagasan paling sederhana — kalikan, lagi dan lagi — beserta varian-variannya yang lebih tajam. Buku: Bab 12, Subbab 12.1–12.11.

Cakupan Pertemuan Ini

1. **Masalahnya** — tiga binatu di satu kampung dan pelanggan yang berpindah.
2. **Vektor eigen dengan kata-kata sederhana** — arah yang hanya diregangkan oleh sebuah matriks.
3. **Metode pangkat** — laju konvergensi, hasil bagi Rayleigh, iterasi invers dan bergeser.
4. **Ruko yang berayun** — frekuensi alami dan resonansi.
5. **PageRank** — vektor eigen web; dan `numpy.linalg`.

Tiga program dalam Java dan Python — `market`, `power`, `pagerank` — semuanya dibangun di atas satu perulangan: kalikan dengan matriks, normalkan, ulangi. Buku: Bab 12.

Tiga Binatu di Satu Kampung

Di sebuah kampung di **Malang**, Laundry Bersih dekat masjid, Laundry Kilat di jalan utama, dan Laundry Wangi, yang terbaru, dekat sekolah, berbagi beberapa ratus rumah tangga.

Pelanggan tidak setia selamanya. Bu Endang, pemilik Bersih, bertanya-tanya: apakah Bersih akan mempertahankan pangsanya, tumbuh, atau pelan-pelan kehilangan pelanggan ke pendatang baru?

Survei 200 rumah tangga: dari pelanggan Bersih, **80% tetap**, 10% pindah ke Kilat, 10% ke Wangi. Kilat mempertahankan 75%; Wangi hanya 60%.

Survei itu menanyakan binatu yang dipakai setiap rumah tangga bulan lalu dan bulan ini. Membagi hitungannya dengan total setiap binatu memberi bagian yang berpindah. Buku: Subbab 12.1.

Vektor Eigen dengan Kata-kata Sederhana

DEFINISI. Vektor eigen v dari matriks persegi A adalah vektor tak nol yang hanya diregangkan oleh A; faktor regangannya λ adalah **nilai eigen**-nya.

$$A v = \lambda v \tag{12.1}$$

Contoh: A dengan baris $(2,1)$ dan $(1,2)$; $(1,1)$ diregangkan 3 kali, $(1,-1)$ satu kali. Setiap vektor lain berbelok.

Persamaan (12.1): nilai eigen menggambarkan apa yang dilakukan sistem bila dibiarkan sendiri — pola mana yang tumbuh, mana yang padam, dan seberapa cepat. Kalikan vektor sembarang dengan A berulang kali: komponen sepanjang $(1,1)$ tumbuh tiga kali setiap langkah, komponen sepanjang $(1,-1)$ tetap, sehingga vektornya makin lama makin mengarah ke $(1,1)$. Itulah seluruh gagasan metode pangkat. Buku: Subbab 12.2.

Pasar Binatu sebagai Rantai Markov

$$P=\begin{pmatrix}0.80 & 0.15 & 0.20\\ 0.10 & 0.75 & 0.20\\ 0.10 & 0.10 & 0.60\end{pmatrix},\quad x_{k+1}=P\,x_k \tag{12.2}$$

Kolom j menyatakan ke mana pelanggan binatu j pergi; setiap kolom berjumlah 1. Pangsa bulan depan adalah P kali pangsa sekarang.

Tulis keadaan awal sebagai kombinasi vektor eigen, dengan pangsa stasioner s untuk $\lambda_1=1$:

$$x_k=P^k x_0=s+c_2\lambda_2^k v_2+c_3\lambda_3^k v_3\to s \tag{12.3}$$

*Persamaan (12.2): sebuah **rantai Markov**. Persamaan (12.3): nilai eigen lainnya lebih kecil daripada 1, jadi suku-sukunya padam dan setiap campuran awal berakhir di s. Jarak ke s menyusut seperti besarnya λ_2 pangkat k. Buku: Subbab 12.3.*

Pasar dalam Kode

Kode 12.1 Dari `market.py`: pangsa bulan demi bulan — metode pangkat pada matriks perpindahan.

```
def main():
    x = [0.40, 0.35, 0.25]                  # shares this month
    print("month   " + "   ".join(f"{n:>7}" for n in NAMES) + "     change")
    print(f"{0:5d}   " + "   ".join(f"{v:7.4f}" for v in x))
    changes = []
    for month in range(1, 16):
        new = mat_vec(P, x)
        change = sum(abs(new[i] - x[i]) for i in range(3))
        changes.append(change)
        x = new
```

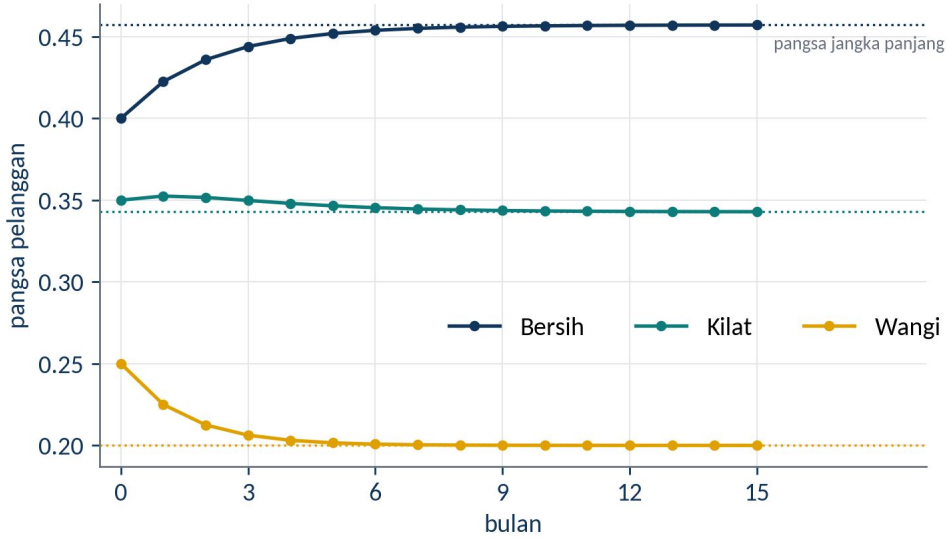
Kode 12.1: tidak ada teori nilai eigen yang terlihat, hanya "bulan depan adalah P kali bulan ini". Baris pencetakan dan rasio terakhir tidak ditampilkan di sini. Matriks stokastik mempertahankan jumlah pangsa, jadi perulangan ini tidak perlu normalisasi. Buku: Subbab 12.3.

Menjalankannya: Ke Mana Pelanggan Berakhir

```
month   Bersih    Kilat    Wangi     change
    0   0.4000   0.3500   0.2500
    1   0.4225   0.3525   0.2250   5.00e-02
    2   0.4359   0.3516   0.2125   2.68e-02
    3   0.4439   0.3498   0.2063   1.61e-02
    4   0.4489   0.3480   0.2031   9.86e-03
    5   0.4519   0.3465   0.2016   6.10e-03
   ...
   15   0.4571   0.3429   0.2000   6.91e-05
change shrinks by a factor 0.6471 per month
long-run shares: Bersih 45.7%, Kilat 34.3%, Wangi 20.0%
```

*Baris 6 sampai 14 tidak ditampilkan di sini. Bersih tumbuh dari 40% ke **45,7%**, Wangi mengendap di 20%. Pergeserannya menyusut sekitar 0,65 per bulan — nilai eigen kedua, 0,65, terukur tanpa menghitungnya. Pangsa stasioner eksaknya* $(16/35, 12/35, 7/35)$*. Buku: Subbab 12.3.*

Pangsa dalam Satu Plot



Gambar 12.1 Pangsa ketiga binatu bulan demi bulan dari 40%, 35%, dan 25%; garis titik-titik adalah pangsa stasioner, vektor eigen matriks perpindahan untuk nilai eigen 1.

Gambar 12.1: dalam setahun pangsanya sudah berjarak beberapa persepuluh persen dari nilai jangka panjangnya, apa pun campuran awalnya. Bagi Bu Endang pesannya menggembirakan tetapi tidak dramatis: jika pelanggan terus berpindah seperti bulan ini, Bersih tumbuh dari 40% ke sekitar 46%, terutama dengan mengambil dari Wangi. Pangsa jangka panjang hanya bergantung pada bagian yang berpindah, bukan pada

campuran awal; menjalankan ulang dengan satu entri diubah, misalnya Wangi mempertahankan 75% alih-alih 60%, memperlihatkan pasar setelah perubahan itu. Buku: Subbab 12.3.

Metode Pangkat secara Umum

Untuk matriks apa pun dengan satu nilai eigen dominan: ulangi $x \leftarrow Ax/\|Ax\|$. Vektornya berbelok ke v_1 dengan laju sebesar λ_2/λ_1.

Untuk A simetris, **hasil bagi Rayleigh** menaksir nilai eigen dengan laju *dua kali* lipat:

$$r(x) = \frac{x^T A x}{x^T x} \tag{12.4}$$

Untuk mencari nilai eigen yang terdekat ke **geseran** σ, terapkan metode pangkat pada inversnya:

$$(A - \sigma I)^{-1} v = \frac{1}{\lambda - \sigma} v \tag{12.5}$$

Persamaan (12.4): galat sebesar ϵ pada vektor memberi galat sebesar ϵ^2 pada $r(x)$. Persamaan (12.5): nilai eigen yang paling dekat ke σ menjadi yang dominan bagi inversnya. Faktorkan $A - \sigma I$ sekali dengan LU (Bab 6), lalu setiap langkah hanya dua penyelesaian segitiga yang murah. Buku: Subbab 12.4.

Bagaimana Ruko Berayun

Indonesia terletak di pertemuan lempeng tektonik. **Ruko** tiga lantai — toko di bawah, keluarga di atas — dimodelkan sebagai tiga massa lantai di atas kolom yang lentur.

Matriks kekakuan K dengan baris $(2,-1,0)$, $(-1,2,-1)$, $(0,-1,1)$ dikali k; setiap massa lantai m. Getaran bebas memberi

$$\frac{1}{m}K\,v=\omega^{2}v,\quad f=\frac{\omega}{2\pi}=\frac{\sqrt{\lambda}}{2\pi} \tag{12.6}$$

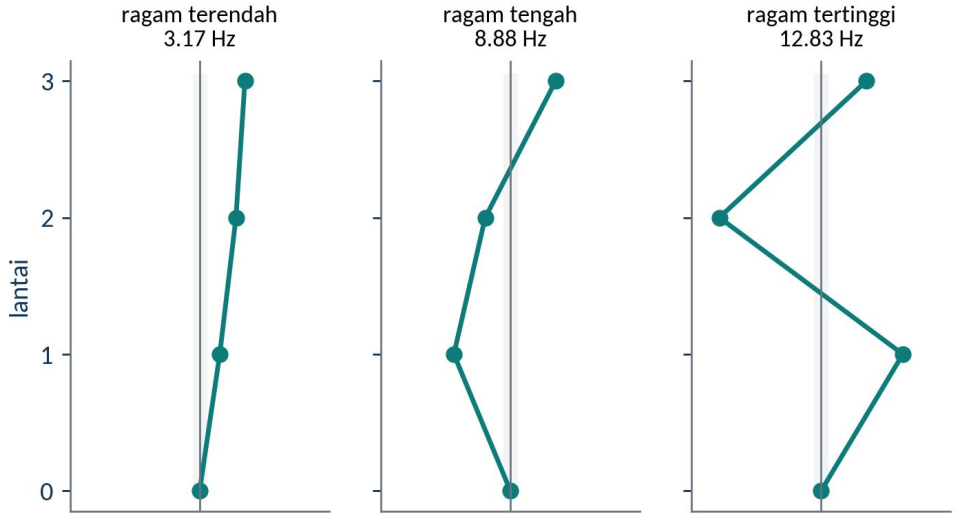
Persamaan (12.6): setiap nilai eigen adalah kuadrat frekuensi sudut alami, setiap vektor eigen sebuah ***bentuk ragam*** *— pola ketika semua lantai bergerak serempak. Model ini hanya memperlihatkan gagasannya; setiap pertanyaan tentang keamanan bangunan sungguhan adalah wewenang insinyur struktur yang berizin. Buku: Subbab 12.5.1–12.5.2.*

Menjalankannya: Tiga Ragam

```
highest lambda   6493.959  f 12.83 Hz  iterations  18  shape  1.802  -2.247   1.000
lowest  lambda    396.125  f  3.17 Hz  iterations   7  shape  0.445   0.802   1.000
middle  lambda   3109.916  f  8.88 Hz  iterations   6  shape -1.247  -0.555   1.000
```

`power.py` dan `Power.java`. *Metode pangkat butuh 18 langkah: hasil bagi Rayleigh membaik* $(3110/6494)^2 = 0.23$ *kali per langkah. Iterasi invers butuh 7: rasionya* $(396/3110)^2 = 0.016$. *Iterasi invers bergeser dengan* $\sigma = 3000$ *butuh 6: rasionya* $(110/2604)^2 = 0.0018$. *Geseran yang baik mengubah metode yang lambat menjadi cepat. Bentuk ragam terendah* $(0.445, 0.802, 1)$ *berarti ketiga lantai bergoyang searah, makin tinggi makin jauh; ragam tertinggi punya tanda yang berganti-ganti dari lantai ke lantai. Buku: Subbab 12.5.2.*

Bentuk Ragam, dan Mengapa Penting



Gambar 12.2 Ketiga bentuk ragam model ruko: simpangan ke samping setiap lantai (tanah di 0), diskalakan agar atap bergerak satu satuan, beserta frekuensi alami setiap ragam.

WAWASAN KUNCI: Diguncang di dekat frekuensi alaminya, bangunan berayun jauh lebih besar daripada tanahnya. Dengan redaman 5%, guncangan 1 Hz diperbesar sekitar 1,1 kali; pada 3 Hz sekitar 7 kali; pada 3,17 Hz, 10 kali. Gempa yang sama dapat

> menjadi goyangan lembut bagi satu bangunan dan ayunan berbahaya bagi bangunan lain, hanya bergantung pada sebuah nilai eigen.

Gambar 12.2: pada ragam terendah semua lantai bergerak ke arah yang sama; pada ragam yang lebih tinggi mereka saling berlawanan. Atap yang lebih berat — tandon air, tempat jemuran — mengubah masalahnya menjadi masalah umum $K\,v = \omega^2 M\,v$*, dan menurunkan frekuensi dasar dari 3,17 ke 2,80 Hz. Buku: Subbab 12.5.3–12.5.4.*

PageRank: Vektor Eigen Web

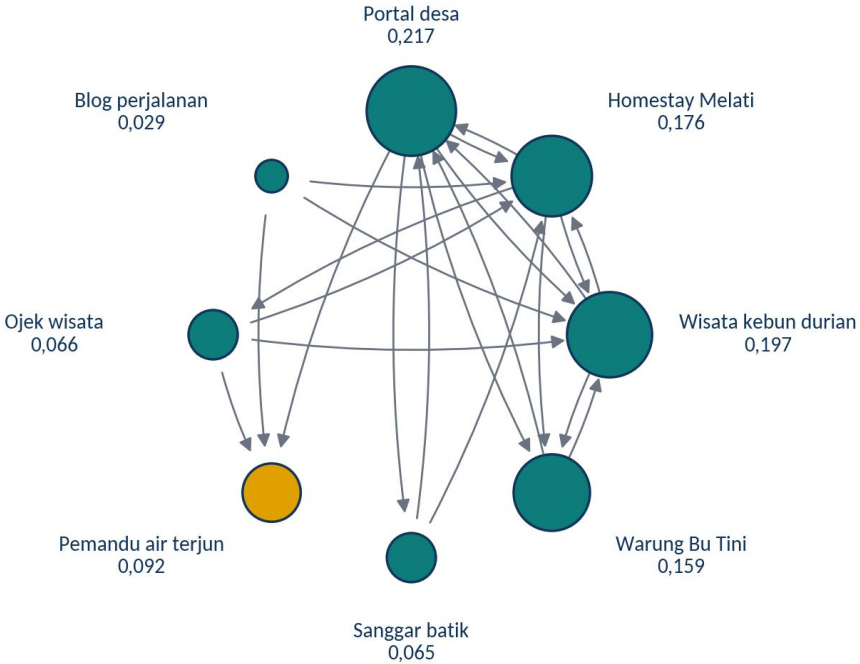
Seorang pembaca menjelajah secara acak: dengan peluang d ia mengikuti tautan acak, jika tidak ia melompat ke halaman mana saja. Halaman tanpa tautan keluar (**menggantung**) mengirimnya ke mana saja.

$$r_i = \frac{1-d}{n} + d\sum_{j \to i} \frac{r_j}{L_j} + d\sum_{j\,\text{dangling}} \frac{r_j}{n} \tag{12.7}$$

Persamaan (12.7): L_j adalah banyaknya tautan keluar dari halaman j. Peringkatnya adalah distribusi stasioner pembaca acak ini — vektor eigen untuk nilai eigen 1 dari matriks Google. Dengan $d = 0.85$, seperti di makalah aslinya, metode pangkat konvergen paling lambat dengan laju d. Buku: Subbab 12.6.

Menjalankannya: Web Desa Wisata

```
converged after 25 iterations, ranks sum to 1.000000000000
1. Village portal       0.2168   (linked from 4)
2. Durian orchard tour  0.1969   (linked from 5)
3. Homestay Melati      0.1757   (linked from 5)
4. Warung Bu Tini       0.1585   (linked from 3)
5. Waterfall guide      0.0922   (linked from 3)
6. Ojek wisata          0.0659   (linked from 1)
```



Gambar 12.3 Jaringan desa wisata: panah adalah tautan, luas lingkaran sebanding dengan PageRank, dan pemandu air terjun (kuning tua) adalah halaman menggantung tanpa tautan keluar.

Enam dari delapan halaman. Portal desa di peringkat pertama walaupun dua halaman punya lebih banyak tautan masuk: ia ditautkan dari halaman-halaman penting, yang meneruskan sebagian besar peringkatnya. Gambar 12.3: pemandu air terjun yang menggantung tetap punya peringkat karena aturan halaman menggantung menyebarkannya ke semua halaman. Buku: Subbab 12.6.

Mengapa 0,85?

d yang kecil memberi lompatan acak bobot besar: dengan $d = 0.5$ kedelapan peringkat berada antara 0,07 dan 0,17 dan hampir tidak berkata apa-apa tentang tautannya.

d yang mendekati 1 membiarkan tautan berbicara hampir sendirian, tetapi nilai eigen kedua matriks Google mendekati 1: metode pangkat melambat, dan peringkatnya peka terhadap perubahan kecil pada tautan.

PageRank **terpersonalisasi** selalu melompat ke satu halaman pilihan: berangkat dari tur kebun durian, halaman mana yang akhirnya dibaca pengunjung?

Mesin pencari modern dan mesin jawab AI menggabungkan banyak sinyal, dan struktur tautan hanyalah salah satunya, tetapi gagasan bahwa kepentingan diwariskan lewat tautan dan dihitung sebagai vektor eigen tetap menjadi salah satu gagasan pokok web. Buku: Subbab 12.6.

Memeriksa dengan Pustaka

```
NumPy 2.4.4, SciPy 1.17.1
market eigenvalues: [1.   0.65 0.5 ]
stationary shares:  [0.457143 0.342857 0.2     ]
building eigenvalues: [ 396.125 3109.916 6493.959]
frequencies (Hz):     [ 3.17  8.88 12.83]
PageRank: [0.2168 0.1757 0.1969 0.1585 0.0654 0.0922 0.0659 0.0285]
second eigenvalue of G: 0.3847
```

`eig` dan `eigvalsh` menghitung semua nilai eigen matriks padat kecil sekaligus dengan algoritma QR dari LAPACK. Nilai eigen kedua pasar adalah 0,65, tepat faktor penyusutan yang kita ukur; bangunan dan PageRank sepakat pada setiap digit yang dicetak. Nilai eigen kedua matriks Google ini, 0,3847, jauh di bawah d, jadi 25 iterasi lebih dari cukup. Untuk matriks jarang yang sangat besar, `scipy.sparse.linalg.eigs` mencari beberapa nilai eigen saja. Buku: Subbab 12.7.

Metode Numerik dalam Praktik: Profitina

Profitina mengukur seberapa siap situs sebuah merek untuk dikutip mesin jawab AI; satu keluarga sinyalnya menyangkut **kedudukan teknis** — bagaimana sebuah situs dibangun dan terhubung.

Ketika pemindainya merayapi sebuah situs, ia membangun struktur tautan halaman-halamannya. “Apakah halaman produk yang paling dipedulikan pemilik terkubur tiga klik dalamnya dengan sedikit tautan menuju ke sana?” adalah pertanyaan tentang vektor eigen dominan graf itu.

Di komunitas waktu-nyata, pembaca berpindah antarkanal seperti pelanggan berpindah antarbinatu; perhatian jangka panjang per kanal adalah distribusi stasioner.

Sinyal dan bobot persisnya adalah milik Profitina; gagasan vektor eigennya adalah bab ini, begitu pula wawasan bahwa nilai eigen kedua menentukan seberapa cepat kanal baru mencapai pangsa jangka panjangnya. Angka memberi saran; orang-orang yang menjalankan merek yang memutuskan. Buku: Subbab 12.8.

Kesalahan Umum

Mencampur konvensi baris dan kolom. Milik kita kolomnya berjumlah 1; periksa jumlahnya.

Lupa menormalkan. $A^k x$ meluap atau menyusut ke nol.

Berhenti karena perubahan kecil padahal nilai eigennya berdekatan. Dengan rasio 0,999 vektornya hampir tidak bergerak jauh sebelum teliti.

Mengharapkan metode pangkat menemukan nilai eigen kembar atau kompleks. Pakai geseran, atau algoritma QR pustaka.

Menginvers matriks dalam iterasi invers. Faktorkan $A - \sigma I$ sekali lalu selesaikan.

Untuk toko Anda sendiri, susun tabel perpindahan dari hitungan nyata dan katakan seberapa kasar bagian-bagiannya bila hanya ada beberapa lusin pelanggan per toko. Buku: Subbab 12.9–12.10.

Ringkasan

1. Vektor eigen adalah arah yang hanya diregangkan matriks (Persamaan (12.1)); nilai eigen menggambarkan apa yang dilakukan sistem bila dibiarkan sendiri.

2. Pelanggan yang berpindah membentuk rantai Markov (Persamaan (12.2)); pangsa stasioner adalah vektor eigen untuk 1, didekati seperti λ_2^k (Persamaan (12.3)): 45,7%, 34,3%, 20,0%.

3. Metode pangkat konvergen dengan laju λ_2/λ_1; hasil bagi Rayleigh (Persamaan (12.4)) menggandakannya untuk matriks simetris.

4. Iterasi invers dan bergeser (Persamaan (12.5)) memberi frekuensi ruko (Persamaan (12.6)): 3,17, 8,88, dan 12,83 Hz.

5. PageRank (Persamaan (12.7)) adalah distribusi stasioner pembaca acak yang teredam; NumPy sepakat di mana-mana.

Satu perulangan — kalikan, normalkan, ulangi — menjawab tiga pertanyaan yang sangat berbeda. Buku: Subbab 12.11.

Pertemuan Berikutnya

Bab 13 — Optimasi: Golden Section, Newton, dan Gradient Descent. Harga terbaik, rute termurah, galat terkecil.

Pencarian satu dimensi tanpa turunan, Newton untuk minimum, dan gradient descent di banyak dimensi — mesin penggerak machine learning.

Sebelumnya: pangsa binatu setelah satu dan dua bulan dengan tangan, dan pemeriksaan bahwa $(16/35, 12/35, 7/35)$ memenuhi $P\,s = s$.

Penyelesaiannya juga mencari ragam ruko tertinggi kedua dengan deflasi Hotelling. Buku: Subbab 12.12–12.13.

Kuliah 13 — Optimasi: Golden Section, Newton, dan Gradient Descent

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 13

Dosen: Irfan Subakti — yifana@gmail.com

Setiap usaha kecil bertanya setelan tombol mana yang memberi hasil terbaik. Bab 2 dan 3 mencari tempat sebuah fungsi bernilai nol; hari ini kita mencari tempat ia terbesar atau terkecil. Kita mulai dengan pencarian yang hanya butuh nilai, beralih ke metode Newton, dan berakhir dengan gradient descent di banyak peubah — metode yang melatih setiap model di bab berikutnya. Buku: Bab 13, Subbab 13.1–13.11.

Cakupan Pertemuan Ini

1. **Masalahnya** — harga terbaik secangkir es teh.
2. **Pencarian golden section** — kurung yang menyusut 0,618 kali per evaluasi.
3. **Metode Newton untuk optimum** — konvergensi kuadratik, dan pelarian di luar titik belok.
4. **Banyak peubah** — gradien, Hessian, gradient descent, batas kestabilan dan lajunya.
5. **Menjalankan dan memeriksa** — dua harga sekaligus, dan pengoptimal SciPy.

Memaksimumkan f sama dengan meminimumkan $-f$; kita berpindah bebas di antara keduanya. Semua program mencetak teks identik dalam Java dan Python. Buku: Bab 13.

Harga Mbak Rina

Mbak Rina berjualan ***es teh jumbo*** di depan sebuah SMK di **Sidoarjo**. Setiap cangkir memakan biaya sekitar Rp1.500.

Selama dua tahun ia memasang harga Rp3.000. Tetangganya menaikkan harga ke Rp5.000 dan tetap laris. Haruskah ia ikut?

Terlalu murah, setiap cangkir hampir tidak menghasilkan apa-apa; terlalu mahal, sedikit siswa yang membeli. Di antaranya ada harga dengan **laba harian** terbesar.

Empat minggu percobaan harga, dicocokkan seperti di Bab 8, memberi kurva permintaan eksponensial.

Bentuk yang sama — lebih banyak lebih baik sampai titik tertentu, lalu lebih buruk — muncul pada harga per kilogram binatu, jam kerja pengemudi ojek, dan pupuk seorang petani. Buku: Subbab 13.1.

Modelnya

Cangkir terjual per hari pada harga p (ribu rupiah):

$$D(p) = A\,e^{-p/s},\quad A = 600,\ s = 2.5 \tag{13.1}$$

Laba harian dengan biaya $c = 1.5$ per cangkir:

$$\pi(p) = (p - c)\,D(p) = (p - c)\,A\,e^{-p/s} \tag{13.2}$$

Persamaan (13.1): setiap tambahan Rp2.500 pada harga membagi penjualan dengan e. Persamaan (13.2): margin kali cangkir. Pada Rp3.000 ia mendapat $1.5 \times 600\,e^{-1.2} = 271$ ribu rupiah sehari. Kalkulus memberi jawabannya, $p^{*} = c + s = 4$, sehingga kita dapat memeriksa setiap metode. Buku: Subbab 13.1.

Gagasannya: Jepit Puncaknya

Fungsi dengan satu puncak pada $[a, b]$ disebut **unimodal**. Bandingkan dua titik uji $x_1 < x_2$: jika laba di x_1 lebih tinggi, puncaknya tidak di luar x_2; jika tidak, puncaknya tidak sebelum x_1.

Letakkan titik uji secara simetris:

$$x_1 = b - r\,L,\; x_2 = a + r\,L,\; \frac{1}{2} < r < 1 \tag{13.3}$$

Pilih r sehingga satu titik uji lama dipakai ulang sebagai yang baru:

$$r^2 = 1 - r \;\Leftrightarrow\; r^2 + r - 1 = 0 \;\Leftrightarrow\; r = \frac{\sqrt{5}-1}{2} \approx 0.6180340 \tag{13.4}$$

Persamaan (13.3): $L = b - a$. *Persamaan (13.4): rasio emas. Setiap langkah berbiaya **satu** evaluasi baru dan menyimpan 61,8% selang — saudara biseksi di dunia optimasi. Buku: Subbab 13.2 dan 13.3.1.*

Pencarian Golden Section

Pseudokode 13.1 *Golden-section search* untuk maksimum fungsi unimodal pada $[a, b]$.

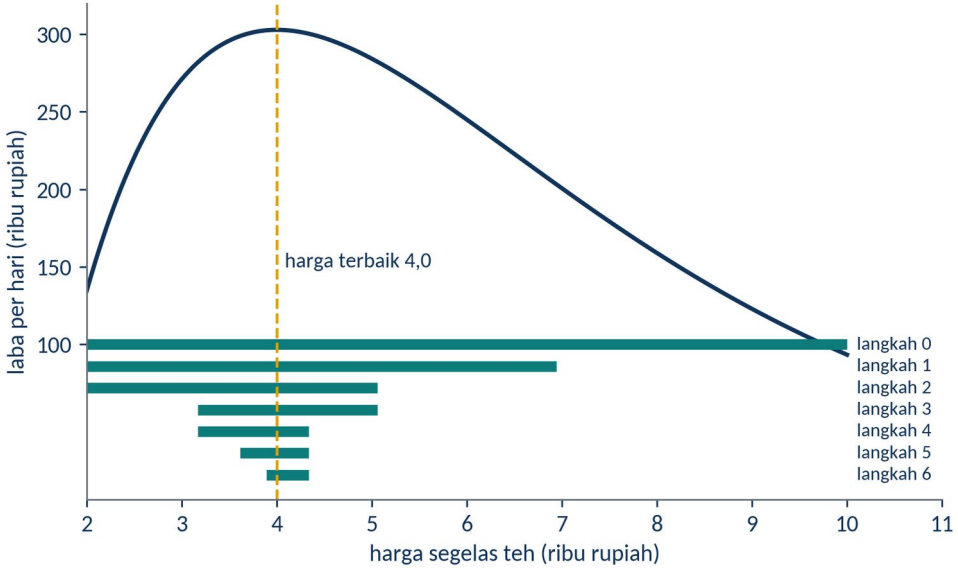
```
GOLDEN-MAX(f, a, b, tol)
 1  r = (sqrt(5) - 1) / 2
 2  x1 = b - r(b - a);  x2 = a + r(b - a)
 3  f1 = f(x1);  f2 = f(x2)
 4  while b - a > tol
 5      if f1 > f2                    // peak is not in (x2, b]
 6          b = x2;  x2 = x1;  f2 = f1
 7          x1 = b - r(b - a);  f1 = f(x1)
 8      else                          // peak is not in [a, x1)
 9          a = x1;  x1 = x2;  f1 = f2
10          x2 = a + r(b - a);  f2 = f(x2)
11  return (a + b) / 2
```

$$L_n = r^n L_0, n \geq \frac{\ln(L_0/\varepsilon)}{\ln(1/r)} \approx 2.078 \ln \frac{L_0}{\varepsilon} \tag{13.5}$$

Pseudokode 13.1: baris 6 dan 9 memakai ulang satu titik uji, sehingga setiap langkah mengevaluasi f sekali. Persamaan (13.5): seperti biseksi, banyaknya langkah diketahui sebelumnya. Buku: Subbab 13.3.1.

Menjalankannya: Harga Teh

k	a	b	b - a	x1	x2	pi(x1)	pi(x2)
1	2.00000	6.94427	4.94427	3.88854	5.05573	302.5347	282.3644
2	2.00000	5.05573	3.05573	3.16718	3.88854	281.7984	302.5347
3	3.16718	5.05573	1.88854	3.88854	4.33437	302.5347	300.3659
4	3.16718	4.33437	1.16718	3.61301	3.88854	298.8193	302.5347
5	3.61301	4.33437	0.72136	3.88854	4.05883	302.5347	302.7622
6	3.88854	4.33437	0.44582	4.05883	4.16408	302.7622	302.2204



Gambar 13.1 Laba harian Mbak Rina dan selang *golden section* pada langkah 0 sampai 6: setiap batang panjangnya 0,618 kali batang di atasnya, dan semuanya memuat harga terbaik.

Baris-baris pertama `golden.py` dan `Golden.java`, mulai dari $[2, 9]$. Perhatikan titik uji 3,88854 muncul lagi baris demi baris: itulah pemakaian ulang. Gambar 13.1: kurungnya selalu memuat Rp4.000. Buku: Subbab 13.5.

Seberapa Teliti Puncak Dapat Ditemukan?

$$\delta_{\min} \approx \sqrt{\epsilon_{\text{mach}}} \sqrt{\frac{2\left|f\left(x^{*}\right)\right|}{\left|f''\left(x^{*}\right)\right|}} \tag{13.6}$$

WAWASAN KUNCI: Di dekat puncak fungsinya datar: bergeser sejauh δ hanya mengubahnya sekitar $\frac{1}{2} f'' \delta^{2}$. Begitu perubahan itu di bawah pembulatan, perbandingannya tidak berarti lagi. Pencarian berdasarkan nilai hanya dapat menemukan optimum sampai sekitar akar kuadrat presisi mesin — sekitar delapan digit.

Persamaan (13.6): kedataran yang sama justru kabar baik dalam praktik — harga bulat di dekat optimum hampir tidak merugikan apa-apa. Bagi Mbak Rina, Rp4.000 optimal sekaligus mudah untuk memberi kembalian. Buku: Subbab 13.3.1.

Metode Newton untuk Optimum

Di optimum $f'=0$: terapkan Newton dari Bab 3 pada f'. Setara dengan itu, cocokkan parabola dari uraian Taylor orde dua lalu lompat ke puncaknya:

$$x_{k+1}=x_k-\frac{f'(x_k)}{f''(x_k)} \tag{13.7}$$

Untuk laba teh, dengan margin $m=p-c$:

$$\pi'(p)=A\,e^{-p/s}\left(1-\frac{m}{s}\right),\pi''(p)=A\,e^{-p/s}\left(\frac{m}{s^2}-\frac{2}{s}\right) \tag{13.8}$$

$$e_{k+1}=-\frac{e_k^2}{s-e_k} \tag{13.9}$$

Persamaan (13.7) memakai kelengkungan untuk menentukan besar langkah. Persamaan (13.8) dan Persamaan (13.9): untuk laba ini galatnya menuruti rekursi yang eksak — kuadratik di dekat optimum, tetapi untuk galat di atas s tandanya berbalik dan iterasinya lari. Titik beloknya di $m=2s$, harga 6,5. Buku: Subbab 13.3.2.

Menjalankannya: Newton Biasa dan Berpengaman

```
plain Newton from p0 = 3.00
   1  p =   3.71428571  error = 2.857e-01
   2  p =   3.97069597  error = 2.930e-02
   3  p =   3.99966049  error = 3.395e-04
   4  p =   3.99999995  error = 4.610e-08
   5  p =   4.00000000  error = 8.882e-16
plain Newton from p0 = 8.00
   1  p =  14.66666667  error = 1.067e+01
   2  p =  17.93197279  error = 1.393e+01
   3  p =  20.97868508  error = 1.698e+01
safeguarded Newton on [2, 9]
   1  bisect  p =   3.75000000  bracket [2.00000, 5.50000]
   2  newton  p =   3.97727273  bracket [3.75000, 5.50000]
...
p = 4.0000000000 after 6 steps
```

Dari 3,00 setiap galat kira-kira menguadratkan galat sebelumnya. Dari 8,00, di luar titik belok, Newton berjalan menjauh selamanya, menambah sekitar tiga ribu rupiah per langkah pada harga yang tidak lagi membuat siapa pun membeli teh. Versi berpengaman mempertahankan kurung, membagi dua sekali, lalu menerima setiap langkah Newton: enam langkah. Baris-baris dihilangkan di tempat yang ditandai. Buku: Subbab 13.5.

Banyak Peubah: Gradien dan Hessian

$$\nabla f(x) = \begin{pmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_n \end{pmatrix}, H(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{i,j=1}^{n} \tag{13.10}$$

$$f(x+h) \approx f(x) + \nabla f(x)^T h + \frac{1}{2} h^T H(x) h \tag{13.11}$$

Mbak Rina menambah **gorengan**; siswa yang membeli teh sering juga membeli camilan, sehingga setiap harga sedikit menurunkan penjualan produk lainnya:

$$T = 600\, e^{-p_1/2.5 - 0.10\, p_2}, G = 900\, e^{-p_2 - 0.08\, p_1} \tag{13.12}$$

Persamaan (13.10): gradien menunjuk ke arah menanjak; Hessian menggambarkan kelengkungan ke segala arah. Persamaan (13.11): di sekitar sebuah titik setiap fungsi mulus tampak seperti mangkuk kuadratik. Persamaan (13.12): teh T dan camilan G, dengan biaya 1,5 dan 0,6 ribu rupiah. Buku: Subbab 13.3.3.

Gradient Descent dan Batas Kestabilannya

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{13.13}$$

Pada mangkuk kuadratik, galat di setiap arah eigen H dikalikan $1 - \alpha \lambda_i$ setiap langkah:

$$\left|1 - \alpha \lambda_i\right| < 1 \text{ for all } i \Leftrightarrow 0 < \alpha < \frac{2}{\lambda_{\max}} \tag{13.14}$$

$$\rho = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa - 1}{\kappa + 1}, \kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{13.15}$$

*Persamaan (13.13): berjalan menurun (untuk laba, minimumkan $-\pi$). Persamaan (13.14) adalah daerah kestabilan Bab 11 dalam wujud lain. Persamaan (13.15): dengan langkah tetap terbaik, lajunya ditentukan **bilangan kondisi** Hessian — mangkuk bundar cukup satu langkah, lembah sempit yang panjang merayap. Buku: Subbab 13.3.4.*

Memilih Langkah Otomatis; Newton di n Dimensi

Backtracking: coba satu langkah, bagi dua sampai penurunannya paling sedikit sebagian kecil dari yang dijanjikan gradien (Armijo):

$$f(x_k - \alpha \nabla f_k) \le f(x_k) - c\,\alpha \|\nabla f_k\|^2, c = 10^{-4} \tag{13.16}$$

Newton: selesaikan sistem linear dengan Hessian di setiap langkah:

$$H(x_k) s_k = \nabla f(x_k), x_{k+1} = x_k - s_k \tag{13.17}$$

Persamaan (13.16) tidak pernah mengambil langkah yang tidak stabil, dari titik awal mana pun. Persamaan (13.17): konvergensi kuadratik berapa pun bilangan kondisinya, dengan biaya turunan kedua dan satu penyelesaian ala Bab 4 per langkah, dan hanya di tempat H definit positif. Buku: Subbab 13.3.4–13.3.5.

Metode-metode Berdampingan

Tabel 13.1 Pengoptimal dalam bab ini: apa yang diperlukan, berapa biaya satu langkah, dan seberapa cepat konvergensinya.

Metode	Butuh	Biaya per langkah	Konvergensi	Bahaya utama
Golden section	nilai, kurung	1 evaluasi	linear, laju 0,618	beberapa puncak
Newton (1 peubah)	f', f''	2 turunan	kuadratik dekat x^*	sisi salah titik belok
Gradient descent	∇f	1 gradien	linear, $(\kappa-1)/(\kappa+1)$	langkah di atas $2/\lambda_{\max}$
Newton (n peubah)	∇f, H	gradien, Hessian, penyelesaian	kuadratik dekat x^*	H tidak definit positif

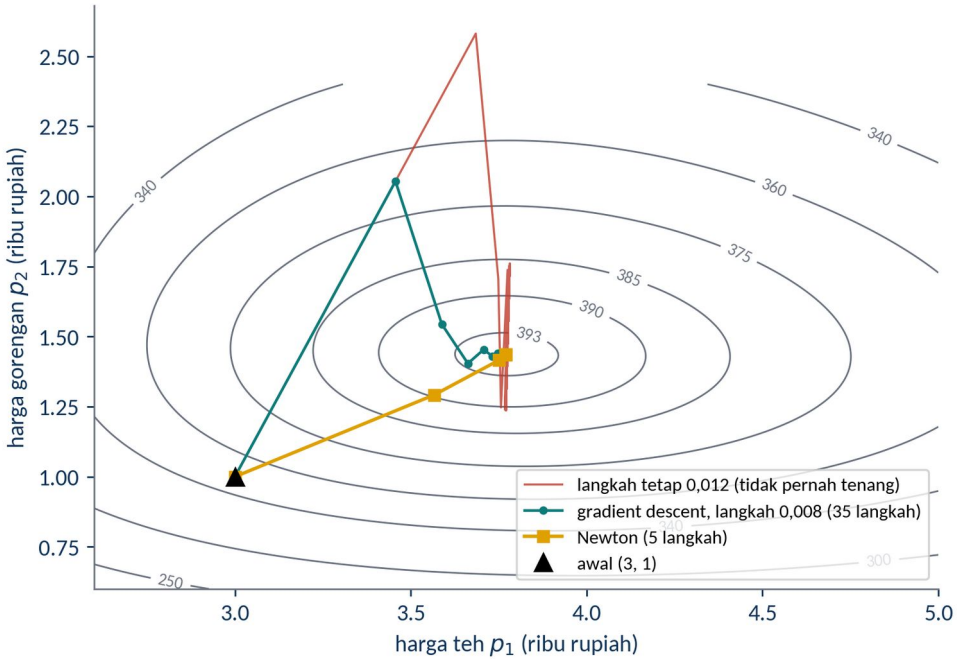
Tabel 13.1 bercermin pada tabel pencari akar Bab 3: aman dan lambat lawan cepat dan rapuh, dengan pengaman untuk menggabungkan keduanya. Buku: Subbab 13.3.

Menjalankannya: Dua Harga

```
gradient at (3, 1), formula           (-57.073365, -131.739616)
gradient at (3, 1), central difference (-57.073365, -131.739616)
fixed step 0.002: p = (3.769928, 1.435197) after 166 steps
fixed step 0.008: p = (3.769928, 1.435197) after 35 steps
fixed step 0.012: no convergence in 2000 steps, |grad| = 43.533
fixed step 0.020: no convergence in 2000 steps, |grad| = 212.849
backtracking  : p = (3.769928, 1.435197) after 35 steps, 294 trial points
...
best prices: tea Rp 3770, snack Rp 1435
sales: 115.1 cups, 158.5 snacks; profit 393.533 thousand rupiah a day
Hessian eigenvalues 49.3370 and 182.0565: condition number 3.690
step limit 2/lambda_max = 0.01099, best fixed step 2/(lo + hi) = 0.00864
```

Rumus gradien diperiksa dulu terhadap beda pusat (Bab 9). Langkah 0,002 dan 0,008 konvergen; 0,012 dan 0,020 berada di atas batas 0,01099 dan tidak pernah konvergen — tepat Persamaan (13.14). Newton, yang tidak ditampilkan di sini, mencapai gradien 3×10^{-12} *dalam lima langkah. Dengan* $\kappa = 3.69$*, Persamaan (13.15) meramalkan* $\rho = 0.574$*. Buku: Subbab 13.5.*

Lintasan pada Bentang Laba



Gambar 13.2 Kontur laba harian (ribu rupiah) terhadap kedua harga, beserta lintasan *gradient descent* (langkah 0,008), langkah tetap di atas batas kestabilan (0,012), dan metode Newton, semuanya mulai dari (3, 1).

Gambar 13.2: gradient descent berzig-zag melintasi lembah; langkah yang tidak stabil memantul dari sisi ke sisi dengan ayunan yang membesar; Newton memotong lurus ke puncak. Bagi Mbak Rina: teh sekitar Rp3.800 dan gorengan Rp1.500 — pembulatan merugikan jauh di bawah seribu rupiah sehari karena puncaknya datar. Buku: Subbab 13.5.

Membandingkan dengan SciPy

```
SciPy golden : p = 3.99999998, 45 evaluations
SciPy Brent  : p = 4.00000000, 11 evaluations
SciPy BFGS       : p = (3.769928, 1.435197), 9 iterations, 11 evaluations
SciPy Newton-CG  : p = (3.769928, 1.435197), 5 iterations, 6 evaluations
SciPy Nelder-Mead: p = (3.769919, 1.435177), 35 iterations, 67 evaluations
```

`check_scipy.py` *mengimpor fungsi tujuan kita sendiri, sehingga hanya pengoptimalnya yang berbeda. Metode Brent menggabungkan golden section dengan langkah parabola; BFGS membangun Hessian hampiran dari gradien; Newton-CG adalah Newton dengan penyelesaian iteratif; Nelder–Mead sama sekali tidak butuh turunan dan paling lambat serta paling kurang teliti. Semua pustaka meminimumkan: kirim* $-\pi$ *. Buku: Subbab 13.6.*

Metode Numerik dalam Praktik: Profitina

Profitina memindai ulang situs setiap merek dengan irama sesuai tingkat langganannya dan menyimpan tren skor mingguan. Memindai lebih sering berarti skor lebih segar dan peringatan lebih cepat, tetapi memakan waktu server dan membebani situs pelanggan.

Sebagai fungsi dari selang pemindaian, biaya total (kebasian ditambah beban) adalah laba Mbak Rina yang dibalik: minimum satu peubah, dan golden section adalah alat yang wajar.

Menyetel parameter kurva emisi PROFIT yang meluruh agar penawaran dan permintaan tetap dalam rentang yang wajar, pada akhirnya, adalah meminimumkan sebuah fungsi rugi atas beberapa parameter kontinu.

Kedataran minimum adalah kabar baik: selang bulat di dekatnya — harian, mingguan — hampir tidak merugikan apa-apa. Memilih di antara sekumpulan kecil kandidat adalah optimasi atas himpunan diskret: evaluasi dan bandingkan. Buku: Subbab 13.9.

Kesalahan Umum

Memaksimumkan padahal kodenya meminimumkan. Kirim $-\pi$ dan balik lagi tanda nilai yang dilaporkan.

Golden section pada fungsi berpuncak banyak. Ia menemukan *sebuah* puncak; plot dulu.

Newton biasa dari titik awal yang buruk. Di luar titik belok ia lari; pertahankan kurung.

Langkah tetap di atas $2/\lambda_{\max}$. Pakai backtracking.

Gradien yang belum diuji. Periksa terhadap beda pusat sebelum mengoptimasi.

Untuk warung Anda sendiri: ubah harga selama beberapa hari setiap kali, hitung yang terjual, cocokkan kurva permintaannya, lalu bergeraklah ke optimum dengan langkah kecil — model memberi saran, penjual yang memutuskan. Buku: Subbab 13.8 dan 13.10.

Ringkasan

1. Golden section menyusutkan kurung dengan $r = 0.618$ per evaluasi (Persamaan (13.4)); banyak langkah diketahui sebelumnya (Persamaan (13.5)); ketelitian dibatasi kedataran (Persamaan (13.6)).

2. Newton untuk optimum (Persamaan (13.7)) konvergen kuadratik (Persamaan (13.9)); kurung membuatnya aman.

3. Gradient descent (Persamaan (13.13)) butuh $\alpha < 2/\lambda_{\max}$ (Persamaan (13.14)); lajunya ditentukan κ (Persamaan (13.15)); backtracking Armijo memilih langkah yang aman (Persamaan (13.16)).

4. Newton di n peubah (Persamaan (13.17)) konvergen kuadratik berapa pun κ.

5. Harga teh terbaik Rp4.000 (302,8 ribu sehari); dengan gorengan Rp3.770 dan Rp1.435 (393,5 ribu); SciPy sepakat.

Gradient descent dengan langkah yang wajar adalah mesin penggerak bab berikutnya. Buku: Subbab 13.11.

Pertemuan Berikutnya

Bab 14 — Metode Numerik untuk Machine Learning. Melatih model adalah kuadrat terkecil, gradien, dan optimasi dalam skala besar.

Regresi linear dan logistik yang dilatih dengan gradient descent, penskalaan fitur, dan evaluasi yang jujur.

Sebelumnya: dua langkah golden section lagi dengan tangan, diperiksa terhadap baris 2 dan 3 program.

Penyelesaiannya ada di buku dalam kedua bahasa. Buku: Subbab 13.12–13.13.

Kuliah 14 — Metode Numerik untuk Machine Learning

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 14

Dosen: Irfan Subakti — yifana@gmail.com

Machine learning terdengar seperti pokok bahasan baru. Di baliknya, sebuah model adalah rumus berbobot, dan belajar berarti memilih bobot yang membuat sebuah fungsi rugi kecil: kuadrat terkecil dari Bab 6 dan 8, gradien dan Newton dari Bab 13, metode pangkat dari Bab 12, dan pembulatan dari Bab 1. Hari ini kita melatih dua model pada catatan seorang pengemudi ojek dan menyaksikan setiap persoalan numerik yang pernah kita temui muncul kembali. Buku: Bab 14, Subbab 14.1–14.11.

Cakupan Pertemuan Ini

1. Masalahnya — berapa lama sebuah perjalanan, dan apakah pelanggan akan membatalkan?

2. Regresi linear — fungsi rugi, gradiennya, persamaan normal lawan QR.

3. Pelatihan dengan gradient descent — laju belajar dari metode pangkat, standardisasi, SGD, momentum, dan Adam.

4. Regresi logistik — peluang, log-loss, metode Newton (IRLS), dan rumus yang stabil.

5. Memeriksa — NumPy dan SciPy, float32 lawan float64, evaluasi yang jujur.

Satu modul per bahasa menampung balok-balok penyusunnya; tiga program memakainya; setiap program Java mencetak persis yang dicetak kembarannya dalam Python. Buku: Bab 14.

Dua Pertanyaan Mas Dimas

Mas Dimas mengemudikan ojek di **Yogyakarta**. Dua bulan perjalanan, **400 baris**: panjang perjalanan, jam sibuk, hujan, jarak jemput, lama perjalanan dalam menit, dan apakah pelanggan membatalkan.

Berapa lama perjalanan ini? Taksiran aplikasi sering meleset saat hujan dan jam sibuk.

Akankah pelanggan ini membatalkan? Pembatalan setelah tiga kilometer menjemput memakan bensin dan seperempat jam dengan sia-sia.

Kedua pertanyaan menyangkut uang, dan keduanya dijawab dengan mencocokkan bobot pada data. Yang pertama berupa bilangan (regresi), yang kedua berupa peluang (klasifikasi). Tidak ada yang ajaib di sini: model yang baik hanya merangkum dengan jujur apa yang sudah tercatat, dan hanya berlaku selama kebiasaan kota, cuaca, dan pelanggannya tidak berubah jauh. Buku: Subbab 14.1.

Sebuah Model dan Sebuah Fungsi Rugi

Model mengubah **fitur-fitur** sebuah perjalanan menjadi ramalan — di sini jumlah berbobot:

$$\hat{y} = w_0 + w_1\,\mathrm{km} + w_2\,\mathrm{rush} + w_3\,\mathrm{rain} = x^T w \tag{14.1}$$

Belajar adalah memilih w yang membuat rata-rata kuadrat galat kecil:

$$L(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(x_i^T w - y_i\right)^2 = \frac{1}{2N}\,\|\,X w - y\,\|^2 \tag{14.2}$$

Persamaan (14.1): jam sibuk dan hujan bernilai 0 atau 1. Persamaan (14.2) adalah kuadrat terkecil Bab 8 dengan faktor $\frac{1}{2N}$ *yang merapikan gradien dan membuatnya tidak bergantung pada ukuran data. Buku: Subbab 14.2.*

Gradien, Hessian, dan Persamaan Normal

$$\nabla L(w) = \frac{1}{N} X^T (X w - y), \nabla^2 L = \frac{1}{N} X^T X \tag{14.3}$$

$$X^T X w = X^T y \tag{14.4}$$

$$\kappa(X^T X) = \kappa(X)^2 \tag{14.5}$$

$$\frac{\| \hat{w} - w \|}{\| w \|} \lesssim \kappa(X)^2 \epsilon_{\text{mach}} \text{(normal equations)}, \lesssim \kappa(X) \epsilon_{\text{mach}} \text{(QR)} \tag{14.6}$$

Persamaan (14.3): Hessiannya tetap, jadi fungsi ruginya mangkuk kuadratik. Persamaan (14.4): gradien nol. Persamaan (14.5) dan Persamaan (14.6) adalah peringatan Bab 6: persamaan normal kehilangan dua kali lebih banyak digit daripada QR. Buku: Subbab 14.3.1–14.3.2.

Menjalankannya: Model Lama Perjalanan

```
400 trips; model minutes = w0 + w1 km + w2 rush + w3 rain
normal equations:   4.00351   2.59547   5.68935   5.52595
Householder QR  :   4.00351   2.59547   5.68935   5.52595
RMSE 2.5693 minutes, R^2 0.9486

polynomials in distance: RMSE (minutes) by method
degree  normal equations   Householder QR
5 km           4.524659         4.524659
9 km           4.503983         4.503983
11 km          4.502751         4.502704
12 km     Cholesky fails         4.494450
12 std         4.494450         4.494450
```

*4,0 menit waktu tetap, **2,6 per kilometer**, tambahan 5,7 pada jam sibuk, 5,5 saat hujan; ramalan yang khas meleset dua setengah menit, kira-kira serakan yang wajar akibat lampu lalu lintas. Pada derajat 11 persamaan normal memberi RMSE yang lebih besar daripada minimum sejatinya — ia sama sekali belum menemukan penyelesaian kuadrat terkecil; pada derajat 12 Cholesky gagal. Dengan jarak yang distandardisasi, kedua metode berhasil. Buku: Subbab 14.5.*

Standardisasi Fitur

$$u_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{14.7}$$

Setiap fitur mendapat rata-rata 0 dan sebaran 1. Pangkat-pangkat bilangan antara sekitar -1.6 dan 2.1 jauh lebih tidak mirip daripada pangkat-pangkat bilangan antara 0,9 dan 15.

> **PERINGATAN:** Terapkan rata-rata dan sebaran *data latih* pada data baru. Menghitungnya ulang pada data baru diam-diam mengubah modelnya.

Persamaan (14.7): untuk data ini, $\kappa(X^T X)$ turun dari 546 ke 1,21. Akibatnya pada pelatihan sangat besar: laju belajar yang aman naik dari sekitar 0,026 ke sekitar 1,8, dan mangkuknya menjadi hampir bundar, sehingga setiap arah konvergen dengan kecepatan yang hampir sama. Buku: Subbab 14.3.2.

Pelatihan dengan Gradient Descent

Gradient descent pada Persamaan (14.2) adalah metode Bab 13 pada mangkuk kuadratik: stabil hanya untuk laju belajar di bawah $2/\lambda_{\max}$ dari $X^T X/N$.

$\lambda_{\max}$ didapat dari beberapa langkah **metode pangkat** (Bab 12) — tanpa menebak.

SGD mini-batch menaksir gradien dari B baris acak:

$$w \leftarrow w - \alpha \frac{1}{B} \sum_{i \in B} \left(x_i^T w - y_i \right) x_i \tag{14.8}$$

Persamaan (14.8): setiap langkah murah dan berderau; dengan jutaan baris, langkah murah yang menang. Dengan laju belajar tetap deraunya tidak pernah padam: SGD berakhir dalam awan di sekitar minimum, sebuah lantai derau. Buku: Subbab 14.3.3.

Momentum dan Adam

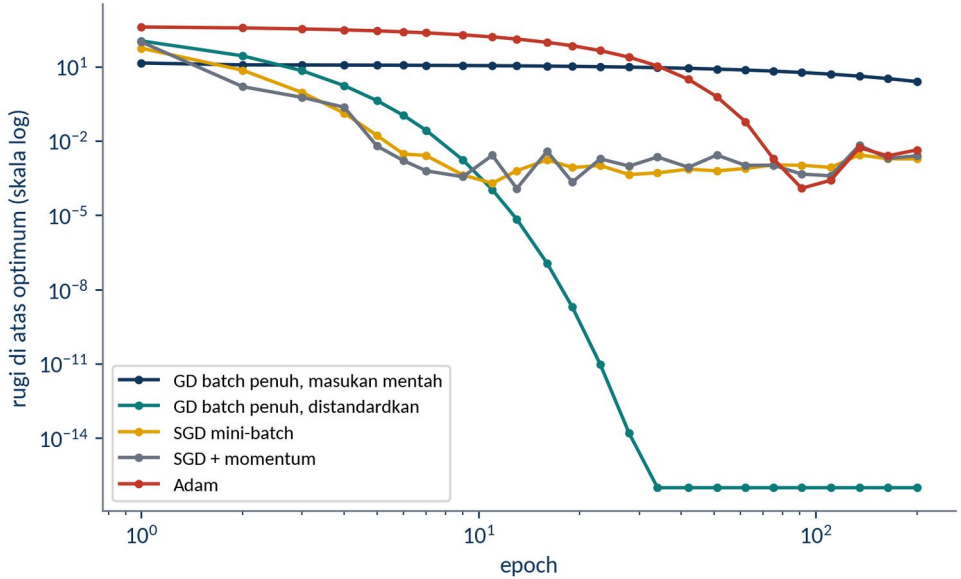
$$m \leftarrow \beta m + g, w \leftarrow w - \alpha m, \beta = 0.9 \tag{14.9}$$

$$m \leftarrow \beta_1 m + (1-\beta_1) g, v \leftarrow \beta_2 v + (1-\beta_2) g^2, w \leftarrow w - \alpha \frac{m/(1-\beta_1^t)}{\sqrt{v/(1-\beta_2^t)} + 10^{-8}} \tag{14.10}$$

Persamaan (14.9): langkahnya mengingat arahnya, yang meredam zig-zag melintasi lembah sempit. Persamaan (14.10): Adam menskalakan langkah setiap bobot dengan taksiran berjalan besarnya gradien bobot itu — obat per koordinat untuk penskalaan yang buruk. Keduanya alat bawaan deep learning. Pada masalah kita yang kecil dan sudah distandardisasi, keunggulannya tidak terlihat; pada jaringan dengan jutaan bobot yang skalanya berbeda-beda, merekalah yang membuat pelatihan mungkin. Buku: Subbab 14.3.3.

Menjalankannya: Pelatihan

```
lambda_max of X^T X / N: raw 77.409, standardized 1.092
method              inputs  lr      epochs  excess loss  reached
full-batch GD       raw     0.012      200     2.51e+00        -
full-batch GD       std     0.5        200     0.00e+00       12
mini-batch SGD      std     0.05       200     1.88e-03       92
SGD + momentum      std     0.005      200     2.51e-03       24
Adam                std     0.05       200     4.35e-03       88
```



Gambar 14.1 Rugi di atas optimum kuadrat terkecil selama pelatihan: *gradient descent* batch penuh pada masukan mentah dan terstandardisasi, serta SGD *mini-batch*, *momentum*, dan Adam pada masukan terstandardisasi (kedua sumbu logaritmik).

Pada masukan mentah laju yang aman di bawah $2/77.4$*, dan 200 epoch menyisakan rugi 2,51 di atas optimum. Setelah distandardisasi, lajunya boleh 0,5 dan ruginya mencapai optimum dalam 12 epoch. Gambar 14.1: metode stokastik turun cepat lalu mendatar di lantai deraunya masing-masing. Buku: Subbab 14.5.*

Regresi Logistik: dari Jumlah ke Peluang

$$p=\sigma(z)=\frac{1}{1+e^{-z}},\sigma'(z)=\sigma(z)(1-\sigma(z)) \tag{14.11}$$

$$L(w)=-\frac{1}{N}\sum_{i=1}^{N}[t_i\ln p_i+(1-t_i)\ln(1-p_i)],p_i=\sigma(x_i^T w) \tag{14.12}$$

$$L(w)=\frac{1}{N}\sum_{i=1}^{N}[\mathrm{softplus}(z_i)-t_i z_i],z_i=x_i^T w \tag{14.13}$$

Persamaan (14.11): sigmoid memeras jumlah berbobot apa pun ke dalam $(0,1)$. *Persamaan (14.12): log-loss menghukum keras jawaban salah yang terlalu yakin;* t_i *bernilai 1 untuk pembatalan. Persamaan (14.13) adalah rugi yang sama yang ditulis ulang sehingga tidak ada yang meluap. Buku: Subbab 14.3.4.*

Metode Newton: IRLS

$$\nabla L = \frac{1}{N} X^T (p - t), H = \frac{1}{N} X^T W X, W = \text{diag}(p_i(1 - p_i)) \tag{14.14}$$

$$\frac{1}{N} X^T W X s = \frac{1}{N} X^T (p - t), w \leftarrow w - s \tag{14.15}$$

Persamaan (14.14): gradiennya berbentuk sama seperti pada kuadrat terkecil; Hessiannya semidefinit positif, sehingga ruginya konveks dan tidak punya minimum palsu. Persamaan (14.15) adalah langkah Newton Bab 13 — masalah kuadrat terkecil berbobot di setiap iterasi, karena itu disebut iteratively reweighted least squares. Buku: Subbab 14.3.4.

Rumus yang Stabil

$$\text{softplus}(z) = \max(z, 0) + \log 1p\left(e^{-|z|}\right) \tag{14.16}$$

Kode 14.1 Dari `mlcore.py`: softplus yang stabil, yang menjaga log-loss tetap berhingga.

```
def softplus(z):
    """log(1 + e^z) without overflow: max(z, 0) + log1p(e^-|z|)."""
    return max(z, 0.0) + math.log1p(math.exp(-abs(z)))
```

$$\epsilon_{32} = 2^{-24} \approx 6.0 \times 10^{-8}, \epsilon_{64} = 2^{-53} \approx 1.1 \times 10^{-16} \tag{14.17}$$

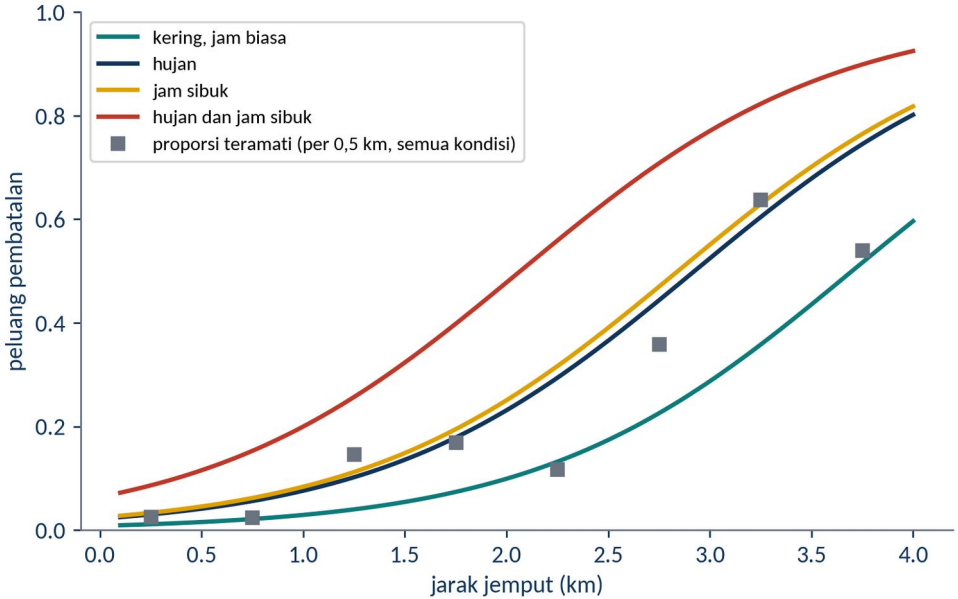
Persamaan (14.16): eksponensial hanya pernah diambil dari bilangan yang tidak positif. Kode 14.1: tiga baris saja. Persamaan (14.17): machine learning sering menghitung dalam float32 demi kecepatan — sekitar tujuh digit, sehingga jumlah panjang dan pengurangan butuh kehati-hatian Bab 1. Buku: Subbab 14.3.5–14.3.6.

Menjalankannya: Akankah Pelanggan Membatalkan?

```
one trip, stable versus naive log-loss
  z =    -5.0, cancelled = 1: naive   5.006715, stable 5.006715
  z =   -40.0, cancelled = 1: naive  40.000000, stable 40.000000
  z =  -800.0, cancelled = 1: naive        inf, stable 800.000000
  z =    40.0, cancelled = 0: naive        inf, stable 40.000000
gradient check, max difference: 2.98e-10
Newton / IRLS from w = 0
  1  loss 0.4596129941  |grad| 5.70e-02
  ...
  6  loss 0.4317353354  |grad| 4.68e-16
weights: -4.80740 1.29936 1.00720 1.11312
odds of cancelling x3.667 per km of pickup, x2.738 in rain, x3.044 in rush hour
```

Rumus naif memberi tak hingga padahal rugi sejatinya 800 atau 40. Pemeriksaan gradien sepakat sampai 3×10^{-10}*. Newton butuh* ***enam langkah****; gradient descent pada jarak jemput yang distandardisasi butuh 128 epoch untuk mendekat sampai* 10^{-6}*. Setiap kilometer jarak jemput mengalikan peluang relatif (odds) pembatalan dengan 3,7. Buku: Subbab 14.5.*

Peluang yang Dipelajari



Gambar 14.2 Peluang sebuah pesanan dibatalkan, sebagaimana dipelajari regresi logistik, terhadap jarak jemput untuk empat kombinasi hujan dan jam sibuk; kotak menunjukkan proporsi teramati per setengah kilometer atas semua kondisi.

> **WAWASAN KUNCI:** Peluang belum merupakan keputusan. Dengan ambang 0,5 model benar 79% dari waktu, menangkap 57 dari 110 pembatalan, dan membunyikan 31 alarm palsu. Letak ambangnya bergantung pada berapa kerugian Mas Dimas akibat pembatalan yang terlewat dan akibat pesanan yang ditolak.

Gambar 14.2: kurva-kurvanya naik tajam bersama jarak jemput dan lebih tinggi saat hujan dan jam sibuk. Bila hanya 27,5% pesanan yang dibatalkan, "tidak ada yang membatalkan" sudah 72,5% tepat — laporkan yang tertangkap dan yang ditandai keliru, bukan hanya ketepatan. Buku: Subbab 14.5.

Membandingkan dengan NumPy dan SciPy

```
lstsq weights: 4.00351 2.59547 5.68935 5.52595
cond(X^T X): raw 546.1, standardized 1.209
degree  5: cond(X) 5.63e+06, cond(X^T X) 3.17e+13
degree  9: cond(X) 3.15e+12, cond(X^T X) 1.62e+23
degree 11: cond(X) 2.55e+15, cond(X^T X) 9.39e+26
degree 12: cond(X) 7.62e+16, cond(X^T X) 5.73e+28
BFGS logistic weights: -4.80740 1.29936 1.00720 1.11312 loss 0.4317353354
```

`lstsq` *memakai dekomposisi nilai singular dan sepakat. Bilangan kondisinya membenarkan Persamaan (14.5): pada derajat 11* $\kappa(X^T X)$ *sekitar* 10^{27}*, jauh di atas* $1/\epsilon_{64}$*, sehingga persamaan normal tidak punya digit tersisa; QR, dengan* $\kappa(X)$ *sekitar* 10^{15}*, masih bertahan dengan beberapa digit. BFGS dari SciPy dengan* `log_expit` *pustaka yang stabil menemukan bobot logistik yang sama. Buku: Subbab 14.6.*

Membuatnya Lebih Baik

Regularisasi. Menambah $\frac{\lambda}{2}\|w\|^2$ (ridge) menggeser setiap nilai eigen $X^T X$ naik sebesar $N\lambda$: κ mengecil, Cholesky tidak mungkin gagal, bobot berhenti mengejar derau.

Sisihkan data. Cocokkan pada sebagian catatan, ukur pada sisanya — pelajaran Bab 8, kini disebut himpunan validasi.

Akumulasikan dengan hati-hati. Jumlah panjang dari rugi dalam float32 kehilangan digit; jumlahkan dalam float64 atau berpasangan.

Langkah-langkah yang sama berlaku untuk usaha kecil mana pun yang menyimpan catatan: pesanan binatu, waktu tunggu di antrean puskesmas, hasil panen setiap baris kebun. Model memberi saran; orangnya yang memutuskan. Buku: Subbab 14.7–14.8.

Metode Numerik dalam Praktik: Profitina

Fitur AI Profitina hanya berjalan pada **model lokal**: Qwen2.5 lewat Ollama, model 7B untuk pekerjaan bermutu dan model kecil opsional (qwen3:1.7b) sebagai jalur cepat untuk klasifikasi singkat.

Model kecil yang menjawab “ya atau tidak” mengerjakan, dalam skala besar, apa yang dikerjakan model pembatalan: jumlah berbobot yang diubah menjadi peluang oleh sigmoid, atau sepupunya untuk banyak kelas, softmax — dihitung secara stabil dengan bentuk log-sum-exp.

Alur umpan balik: umpan balik pengguna atas jawaban AI dikurasi menjadi dataset, modelnya dilatih dengannya lalu **diukur sebelum dirilis**.

Fitur Link Safety memperlihatkan bahwa ambang sebuah pengklasifikasi adalah keputusan: lapisan aturannya menangkap 27 dari 28 teks penipuan yang realistis tanpa positif palsu pada 30 teks normal dalam tolok ukur internal, dan menggeser ambang akan menukar penipuan yang terlewat dengan alarm palsu, persis seperti ambang Mas Dimas. Buku: Subbab 14.9.

Kesalahan Umum

Menyelesaikan kuadrat terkecil dengan menginvers $X^T X$. Pakai QR atau rutin pustaka.

Fitur yang tidak diskalakan. Standardisasi dengan Persamaan (14.7), memakai statistik data latih.

Laju belajar yang disalin dari tutorial. Taksir $\lambda_{\max}$ dengan metode pangkat.

Log-loss seperti tertulis di buku teks. Pakai bentuk softplus.

Mengira lantai derau SGD sebagai konvergensi; memercayai ketepatan saja.

Setiap kesalahan ini adalah kesalahan metode numerik dari bab sebelumnya yang berganti pakaian. Buku: Subbab 14.10.

Ringkasan

1. Belajar adalah meminimumkan fungsi rugi: kuadrat terkecil (Persamaan (14.2)) untuk bilangan, log-loss (Persamaan (14.12)) untuk peluang.

2. Persamaan normal (Persamaan (14.4)) mengkuadratkan κ (Persamaan (14.5)); QR tidak (Persamaan (14.6)).

3. Laju gradient descent ditentukan κ; metode pangkat memberi laju belajar yang aman; standardisasi memangkas κ dari 546 ke 1,21 dan epoch-nya menjadi 12.

4. SGD (Persamaan (14.8)), momentum, dan Adam menukar langkah eksak dengan langkah murah; IRLS (Persamaan (14.15)) konvergen dalam enam langkah.

5. Rumus stabil (Persamaan (14.16)) dan pemeriksaan gradien menjaga pelatihan tetap jujur. Lama perjalanan: sekitar 4,0 + 2,6 per km, +5,7 saat jam sibuk, +5,5 saat hujan.

Metode numeriknya adalah milik Bab 1, 6, 8, 12, dan 13; machine learning adalah tempat semuanya bertemu. Buku: Subbab 14.11.

Pertemuan Berikutnya

Bab 15 — Komputasi Numerik Cepat dan Paralel. Jawaban yang sama, lebih cepat.

Ke mana waktu habis, memori dan cache, vektorisasi, dan banyak inti yang mengerjakan satu masalah.

Sebelumnya: cocokkan garis tiga perjalanan dengan tangan memakai jarak yang distandardisasi, lalu kembalikan ke satuan semula.

Garis yang dikembalikan harus $4.333 + 2.667\ \text{km}$*, sama seperti tanpa standardisasi; penyelesaiannya ada di buku. Buku: Subbab 14.12–14.13.*

Kuliah 15 — Komputasi Numerik Cepat dan Paralel

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 15

Dosen: Irfan Subakti — yifana@gmail.com

Metode numerik yang benar tetapi memakan waktu sejam, bagi kebanyakan orang, adalah metode yang tidak akan mereka pakai. Hari ini kita mengambil satu komputasi dari Bab 5, suhu wajan martabak, memperhalusnya sampai setiap milimeter, lalu membuatnya lebih dari sepuluh ribu kali lebih cepat: dengan memahami ke mana waktu habis, dengan vektorisasi, dengan memakai beberapa inti, dan terutama dengan memilih algoritma yang lebih baik. Di sepanjang jalan, jumlah paralel ternyata mengubah jawabannya. Buku: Bab 15, Subbab 15.1–15.11.

Cakupan Pertemuan Ini

1. Masalahnya — sejam untuk satu gambar wajan.

2. Ke mana waktu habis — operasi lawan lalu lintas memori: roofline.

3. Kode yang lebih cepat — vektorisasi, perulangan terkompilasi, utas (*thread*), dan hukum Amdahl.

4. Algoritma yang lebih cepat — banyak sapuan Jacobi lawan SOR, dan urutan merah-hitam untuk SOR paralel.

5. Jumlah yang dapat diulang — penjumlahan naif, per potongan, berpasangan, dan Kahan.

Setiap program mencetak angka yang sama dalam Java dan Python, pada satu utas atau beberapa; hanya baris waktu opsional yang berbeda dari mesin ke mesin. Buku: Bab 15.

Sejam untuk Satu Gambar

Wajan Bab 5: persegi 40 cm, tepi-tepinya pada **200, 60, 80, dan 150 °C**.

Seorang pandai besi yang membuat wajan untuk selusin gerobak ingin suhunya **setiap milimeter** — bercak pucat satu sentimeter merusak sebuah martabak. Itu berarti $400 \times 400 = 160.000$ peubah.

Perulangan Python biasa: sekitar 15 juta pembaruan titik per detik. Jacobi butuh sekitar **340.000 sapuan** pada resolusi ini: 5.4×10^{10} pembaruan — **sejam** per gambar.

Ia ingin dua puluh gambar sebelum makan siang.

Tiga jalan menuju kecepatan: buat setiap pembaruan lebih murah, kerjakan beberapa pembaruan sekaligus, atau butuhkan lebih sedikit pembaruan. Bab ini mengukur ketiganya pada satu mesin. Hasil akhirnya bukan hanya lebih cepat, tetapi juga tetap sama sampai digit terakhir, dalam kedua bahasa dan dengan berapa pun utas: kecepatan yang mengubah jawaban bukanlah kemajuan. Buku: Subbab 15.1.

Gagasannya dalam Kata-kata Sederhana

Komputer menghabiskan waktu untuk **aritmetika**, untuk **memindahkan data** dari memori, dan untuk **pembukuan** (interpreter yang memutuskan arti setiap baris).

Hapus pembukuannya: operasi seluruh larik, atau bahasa terkompilasi.

Bagi pekerjaannya: beberapa inti mengerjakan jalur grid yang terpisah — jika metodenya mengizinkan.

Kurangi pekerjaannya: metode yang konvergen dalam lebih sedikit sapuan.

Jalan terakhir biasanya yang terpanjang. Bahasa yang lebih cepat mengalikan kecepatan dengan sebuah konstanta; algoritma yang lebih baik dapat mengubah cara pekerjaan tumbuh bersama ukuran masalah. Buku: Subbab 15.2.

Model Biaya: Roofline

W operasi dengan laju F, Q byte dipindahkan dengan lebar pita B:

$$T \geq \max\left(\frac{W}{F}, \frac{Q}{B}\right) \tag{15.1}$$

Satu sapuan Jacobi mengerjakan $4n^2$ operasi dan mengalirkan sekitar $16n^2$ byte: **0,25 operasi per byte**.

Sebuah inti dapat menghitung jauh lebih cepat daripada yang dapat dikirim memori pada rasio itu: Jacobi **terikat memori**.

Persamaan (15.1): rasio W/Q adalah intensitas aritmetika. Di bawah F/B, aritmetika yang lebih cepat atau inti yang lebih banyak tidak menolong begitu saluran memori penuh. Digambar pada grafik log-log, batas ini adalah "roofline" yang dipakai para insinyur kinerja untuk memutuskan apa yang perlu dioptimalkan. Buku: Subbab 15.3.1.

Vektorisasi

Kode 15.1 Dari `griddle_numpy.py`: satu pernyataan NumPy memperbarui satu blok baris utuh.

```
def sweep_rows(u, v, lo, hi):
    """Jacobi update of inner rows lo..hi-1: one line does a whole block."""
    v[lo:hi, 1:-1] = 0.25 * (u[lo - 1:hi - 1, 1:-1] + u[lo + 1:hi + 1, 1:-1]
                             + u[lo:hi, :-2] + u[lo:hi, 2:])
```

Kode 15.1: irisan-irisan yang digeser adalah keempat tetangga setiap titik sekaligus. Perulangan atas titik-titik kini berjalan di dalam kode terkompilasi NumPy, bukan di interpreter. Dengan `lo = 1` dan `hi = n + 1` ia adalah sapuan penuh; dengan batas lain ia adalah jalur milik satu utas. Buku: Subbab 15.3.2.

Paralelisme dan Hukum Amdahl

Jika bagian f dari pekerjaan berjalan paralel sempurna pada p inti dan sisanya tetap serial:

$$S(p) = \frac{T_1}{T_p} = \frac{1}{(1-f) + f/p} \leq \frac{1}{1-f} \tag{15.2}$$

Dengan $f = 0.95$, delapan inti memberi 5,93, dan berapa pun intinya tidak pernah lebih dari 20.

> **PERINGATAN:** Jacobi terbagi dengan aman menjadi jalur-jalur: setiap utas membaca grid lama dan menulis barisnya sendiri di grid baru. Gauss–Seidel dan SOR memperbarui *di tempat*; bila dibagi begitu saja ke beberapa utas, keduanya mengalami **kondisi balapan** (*race condition*) dan memberi hasil yang bergantung pada waktu.

Persamaan (15.2): bagian serial, sekecil apa pun, pada akhirnya mendominasi; sinkronisasi dan lebar pita memori bersama mengambil bagian lagi. Utas Python mengerjakan aritmetika Python murni secara bergiliran karena kunci interpreter global; NumPy melepas kunci itu di dalam perulangannya. Buku: Subbab 15.3.3.

Algoritma yang Lebih Baik

Dari teori Bab 5, banyak sapuan untuk menyusutkan galat sebesar ε pada grid $n \times n$:

$$k_J \approx \frac{\ln(1/\varepsilon)}{1-\rho_J} \approx \frac{2(n+1)^2}{\pi^2} \ln\frac{1}{\varepsilon} \tag{15.3}$$

$$k_{\text{SOR}} \approx \frac{n+1}{2\pi} \ln\frac{1}{\varepsilon} \tag{15.4}$$

Untuk $n = 400$: sekitar **340.000** sapuan Jacobi lawan sekitar **1.200** untuk SOR optimal.

*Persamaan (15.3) tumbuh seperti n^2, Persamaan (15.4) seperti n. Faktor beberapa ratus — lebih besar daripada yang dapat diberikan bahasa atau banyaknya inti. **Urutan merah-hitam** mewarnai grid seperti papan catur: titik merah hanya bergantung pada titik hitam dan sebaliknya, sehingga setiap setengah sapuan SOR adalah satu langkah tervektorisasi dan paralel. Buku: Subbab 15.3.4.*

Mengapa Jumlah Paralel Mengubah Jawabannya

Penjumlahan floating point tidak asosiatif. Menjumlahkan n bilangan dari kiri ke kanan dapat meleset sebesar

$$|\hat{s} - s| \leq (n-1)u \sum_{i=1}^{n} |x_i| + O(u^2) \tag{15.5}$$

Menjumlahkan berpasangan, lalu pasangan dari pasangan, hanya sebesar

$$|\hat{s} - s| \leq \lceil \log_2 n \rceil u \sum_{i=1}^{n} |x_i| + O(u^2) \tag{15.6}$$

Persamaan (15.5): batasnya tumbuh bersama n. Persamaan (15.6): ia hanya tumbuh bersama $\log_2 n$ — untuk sepuluh juta suku, 24 alih-alih sepuluh juta. Membagi sebuah jumlah ke beberapa utas mengubah urutannya, jadi mengubah pembulatannya, jadi mengubah jawabannya. Buku: Subbab 15.3.5.

Penjumlahan Terkompensasi Kahan

Pseudokode 15.1 Penjumlahan terkompensasi Kahan: c memulihkan galat pembulatan setiap penjumlahan dan menyuapkannya ke penjumlahan berikutnya.

```
KAHAN-SUM(x[1..n])
 1  s = 0;  c = 0
 2  for i = 1 to n
 3      y = x[i] - c          // the next term, corrected by what was lost
 4      t = s + y             // big + small: low bits of y are lost here
 5      c = (t - s) - y       // (t - s) is what was really added; minus y
 6      s = t                 //   leaves exactly the part that was lost
 7  return s
```

Pseudokode 15.1 adalah trik buku kas laundry Bab 1, kini untuk pekerjaan paralel. Galatnya sekitar $2u$ kali jumlah nilai mutlaknya, tidak bergantung pada n. Biayanya empat operasi per suku alih-alih satu, tetapi untuk perulangan yang terikat memori operasi tambahan itu hampir gratis, karena prosesor toh sedang menunggu data. Buku: Subbab 15.3.5.

Menjalankannya: Sepuluh Juta Kali 0,1

```
sum of 10000000 copies of 0.1; exact value 1000000.0
naive, one thread  999999.9998389754  error -1.610e-04
naive, 2 chunks    999999.9999107814  error -8.922e-05
naive, 4 chunks    1000000.0000402561  error +4.026e-05
naive, 8 chunks    1000000.0000223045  error +2.230e-05
naive, 16 chunks   999999.9999899360  error -1.006e-05
pairwise           1000000.0000000000  error +0.000e+00
Kahan              1000000.0000000000  error +0.000e+00
```

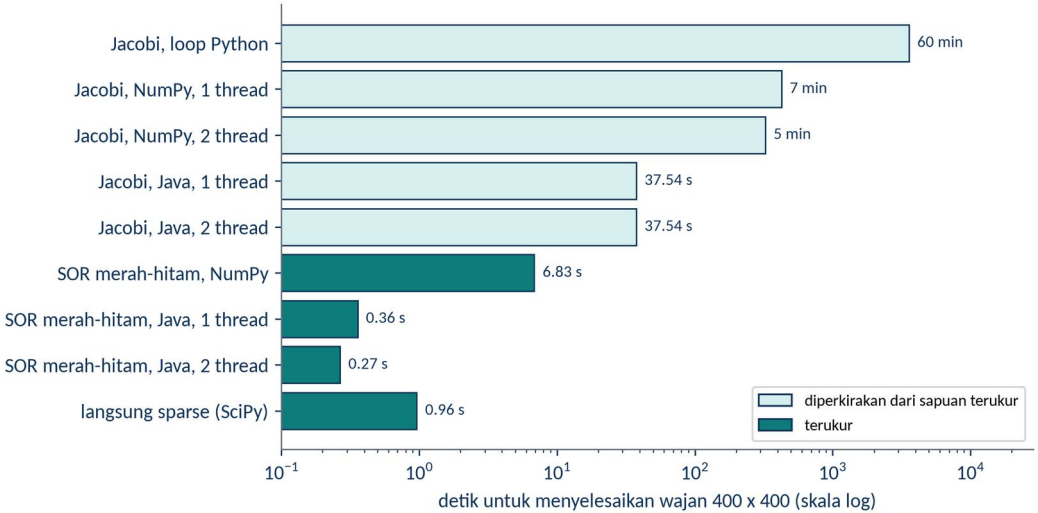
Setiap pemotongan memberi jawaban salah yang berbeda, kadang terlalu kecil, kadang terlalu besar. Program yang menjumlahkan “dengan utas sebanyak inti” akan mencetak satu hasil di laptop 4 inti dan hasil lain di server 16 inti. Berpasangan dan Kahan benar sampai semua digit yang dicetak. Aturannya: tetapkan potongan berdasarkan data, bukan berdasarkan banyaknya inti. Buku: Subbab 15.5.

Menjalankannya: Wajan Halus

```
grid 400 x 400, 2000 Jacobi sweeps, 1 thread(s)
center 0.000000 C, mean 28.428053 C, last change 3.168e-02
time 0.236 s, 1356.4 million updates/s
grid 400 x 400, 2000 Jacobi sweeps, 2 thread(s)
center 0.000000 C, mean 28.428053 C, last change 3.168e-02
time 0.247 s, 1295.8 million updates/s
grid 400 x 400, red-black SOR, omega 1.984453: 1171 sweeps
center 122.572805 C, mean 122.499983 C
time 0.362 s
```

*Jalannya Java pada mesin uji dua inti. Python biasa: 15,6 juta pembaruan per detik; NumPy: 133 juta pada satu utas; Java: **1,36 miliar**, dan tidak lebih pada dua utas — pada laju itu perulangannya dibatasi memori (Persamaan (15.1)). Jacobi setelah 2000 sapuan bahkan belum menghangatkan bagian tengah. SOR merah-hitam menyelesaikan seluruh pelat dalam 1171 sapuan dan sekitar sepertiga detik. Buku: Subbab 15.5.*

Dari Mana Kecepatannya Datang

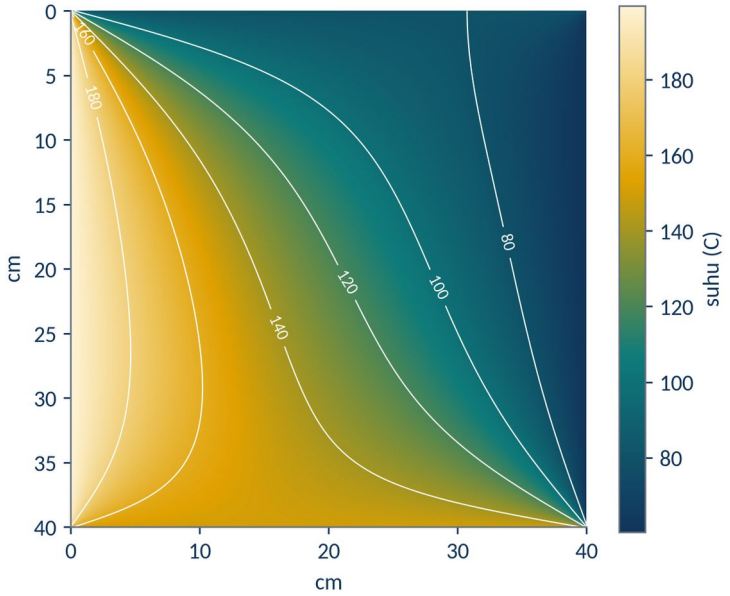


Gambar 15.1 Waktu untuk menyelesaikan wajan 400×400 pada mesin uji dua inti: Jacobi (diperkirakan dari waktu terukur per sapuan dan banyaknya sapuan yang diperlukan), SOR merah-hitam, dan pemecah langsung *sparse* dari SciPy (terukur). Skalanya logaritmik.

WAWASAN KUNCI: Dari yang paling lambat ke yang paling cepat ada faktor lebih dari sepuluh ribu, dan faktor tunggal terbesar, beberapa ratus, datang dari **algoritmanya**, bukan dari bahasa atau inti. Hitung sapuan sebelum menghitung nanodetik.

Gambar 15.1: batang Jacobi mengalikan waktu terukur per sapuan dengan 340.000 (sejam Python biasa tidak membuktikan hal baru); SOR dan pemecah langsung adalah jalannya yang terukur. Buku: Subbab 15.5.

Gambar untuk Pandai Besi



Gambar 15.2 Suhu tunak wajan 40 cm pada grid 1 mm, dengan tepi kiri 200, kanan 60, atas 80, dan bawah 150 derajat Celsius; garis putih adalah isoterm.

Gambar 15.2: 160.000 suhu dalam sepertiga detik, sehingga dua puluh tata letak kompor hanya butuh beberapa detik, bukan sehari. Rata-ratanya, 122,5 °C, tepat rata-rata keempat suhu tepi, sebagaimana dituntut simetri — pemeriksaan cuma-cuma bagi seluruh komputasi. Buku: Subbab 15.5.

Membandingkan dengan NumPy dan SciPy

```
math.fsum (correctly rounded): 1000000.0000000000
numpy.sum (pairwise)         : 1000000.0000000000
sparse direct solve, 160000 unknowns: center 122.572847 C, mean 122.500000 C, 0.87 s
```

`math.fsum` mengembalikan jumlah nilai eksak yang dibulatkan dengan benar; `sum` dari NumPy berpasangan dan sepakat. Pemecah langsung jarang SciPy memperlakukan wajan sebagai sistem linear Bab 5, dibangun dengan `kronsum`, dan butuh 0,87 s; SOR merah-hitam sepakat dengannya di bagian tengah sampai sekitar 5×10^{-5} *°C. Buku: Subbab 15.6.*

Membuatnya Lebih Baik

Ukur sebelum mengoptimalkan. `cProfile` di Python, Java Flight Recorder di JDK. Jalankan Java pertama kali sudah termasuk kompilasi; ukur beberapa kali.

Proses atau Python free-threaded untuk kode Python yang terikat CPU. `ProcessPoolExecutor` menjalankan proses terpisah; build *free-threaded* Python 3.14 yang didukung resmi menghapus kunci interpreter, dengan sedikit mengorbankan kecepatan satu utas.

Algoritma yang lebih baik sebelum mesin yang lebih besar. SOR, gradien konjugat berprakondisi, multigrid, atau pemecah langsung jarang masing-masing mengalahkan Jacobi berlipat-lipat orde. Paralelkan algoritma yang baik, bukan yang buruk.

Jaga hasil tetap dapat diulang. Tetapkan pemotongan berdasarkan data; beri benih setiap aliran bilangan acak dari tugasnya, bukan dari utasnya; simpan satu jalankan acuan berutas tunggal.

Prosesor grafis punya ribuan inti sederhana dan lebar pita memori yang sangat tinggi; sapuan stensil seperti Jacobi berjalan di sana hampir tanpa perubahan.

Roofline, hukum Amdahl, dan reduksi yang dapat diulang berlaku di GPU tanpa perubahan; hanya bilangan F dan B yang lebih besar. Buku: Subbab 15.7.

Metode Numerik dalam Praktik: Profitina

Memindai sebuah situs sebagian besar adalah **menunggu** halaman tiba, sehingga banyak pemindaian dapat berlangsung bersamaan bahkan dengan sedikit inti: konkurensi yang dibatasi masukan dan keluaran, dengan pertanyaan berapa banyak permintaan yang pantas dikirim ke sebuah situs sekaligus.

Memberi skor halaman dan menjalankan model Qwen2.5 lokal adalah **komputasi**, tempat Amdahl dan roofline berlaku: kecepatan model sebagian besar ditentukan oleh seberapa cepat bobotnya dapat dibaca dari memori — itulah sebabnya ada model sangat kecil sebagai jalur cepat.

Gerbang rilis — lebih dari seribu uji otomatis — sangat mudah diparalelkan, asalkan tidak ada dua uji yang berbagi keadaan.

Tren skor mingguan adalah jumlah atas tabel yang terus bertambah dan harus keluar sama apa pun urutan barisnya: penjumlahan yang dapat diulang dari bab ini. Buku: Subbab 15.9.

Kesalahan Umum

Mengoptimalkan hal yang salah. Sapuan Jacobi yang sepuluh kali lebih cepat tetap ratusan kali lebih lambat daripada SOR.

Mengukur waktu sekali saja. Ulangi, laporkan yang terbaik atau median, dan sebutkan mesinnya.

Mengharapkan p inti memberi p kali kecepatan. Amdahl, sinkronisasi, dan lebar pita mengambil bagiannya.

Memparalelkan pembaruan di tempat; membiarkan banyaknya utas menentukan sebuah jumlah.

Utas Python untuk aritmetika Python murni. Pakai NumPy, proses, atau build free-threaded.

Ukur sebelum mengoptimalkan: profiler memperlihatkan ke mana waktu sebenarnya habis, dan jarang di tempat yang kita duga. Daftar periksa yang sama melayani makro spreadsheet, simulasi kebun, atau pelatihan sebuah model. Buku: Subbab 15.8 dan 15.10.

Ringkasan

1. Waktu habis untuk aritmetika, perpindahan data, dan pembukuan; roofline (Persamaan (15.1)) memperlihatkan Jacobi terikat memori.
2. Vektorisasi menyingkirkan interpreter (15,6 menjadi 133 juta pembaruan per detik); Java terkompilasi mencapai 1,36 miliar, batas memorinya.
3. Hukum Amdahl (Persamaan (15.2)) membatasi percepatan paralel; Jacobi terbagi dengan aman, SOR butuh urutan merah-hitam.
4. Jacobi butuh sekitar 340.000 sapuan, SOR 1171 (Persamaan (15.3), Persamaan (15.4)): algoritma yang lebih baik mengalahkan bahasa yang lebih cepat dan inti tambahan.
5. Jumlah bergantung pada urutan (Persamaan (15.5)); berpasangan (Persamaan (15.6)) dan Kahan (Pseudokode 15.1) membuatnya teliti dan dapat diulang.

Digit yang identik di perulangan Python, NumPy, dan Java datang dari mengerjakan operasi yang sama dengan urutan yang sama. Buku: Subbab 15.11.

Pertemuan Berikutnya

Bab 16 — Proyek Akhir: Menyimulasikan Sistem Nyata dari Awal sampai Akhir. Satu masalah nyata, semua alat mata kuliah ini.

Pemodelan, persamaan diferensial, pencocokan, pencarian akar, optimasi, dan Monte Carlo, diperiksa di setiap langkah.

Sebelumnya: hukum Amdahl untuk perulangan yang memakan 90% waktu, pada 4, 16, dan tak hingga banyaknya inti.

Soal itu juga menanyakan bagian mana yang harus paralel agar 16 inti memberi percepatan 12; penyelesaiannya ada di buku. Buku: Subbab 15.12–15.13.

Kuliah 16 — Proyek Akhir: Menyimulasikan Sistem Nyata dari Awal sampai Akhir

Mata kuliah: Metode Numerik

Buku: Irfan Subakti, *Metode Numerik: Dari Analisis Galat hingga Machine Learning, dengan Java dan Python*, Bab 16

Dosen: Irfan Subakti — yifana@gmail.com

Setiap bab sejauh ini menyelesaikan satu jenis masalah. Pertanyaan nyata tidak datang tersusun per bab. Hari ini satu pertanyaan — seberapa besar tandon air yang dibutuhkan kebun durian selama kemarau? — dijawab oleh simulasi yang dirakit dari metode-metode seluruh mata kuliah, setiap bagian diperiksa dengan metode kedua sebelum dipercaya, dan keseluruhannya diulang untuk ratusan musim yang mungkin terjadi. Buku: Bab 16, Subbab 16.1–16.11.

Cakupan Pertemuan Ini

1. Masalahnya — tandon yang kering pada bulan September.

2. Rencananya — simulasi harian yang dibangun dari bagian-bagian yang sudah diperiksa.

3. Bagian-bagiannya — tandon, pompa, cuaca, badai, saluran tetes, dan sensor, masing-masing dengan dua metode.

4. Banyak musim — Monte Carlo secara paralel, dapat diulang.

5. Keputusannya — pencarian golden section pada biaya tahunan, dan aturan newsvendor yang membenarkannya.

Program-programnya ada dalam Java dan Python dan mencetak hasil yang identik; SciPy menghitung ulang setiap bagian. Buku: Bab 16.

Kemarau Pak Hadi

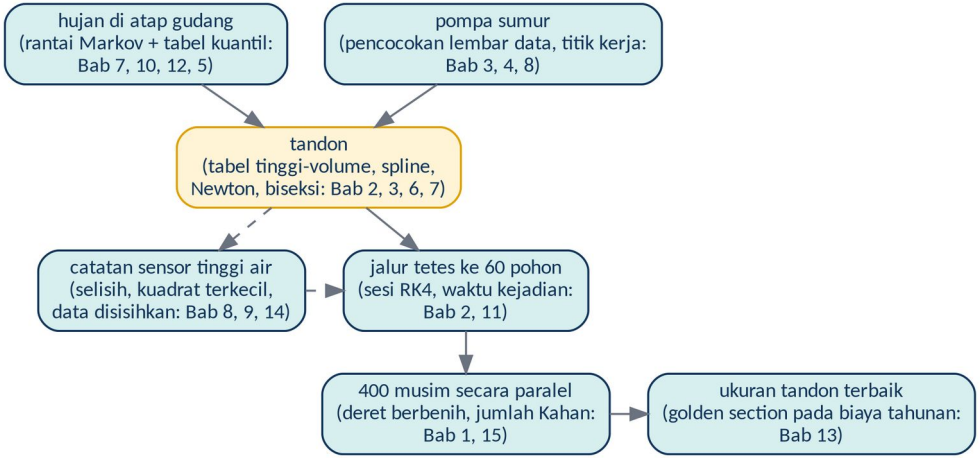
Kebun buah dan durian di timur **Jember**: enam puluh pohon durian yang sedang berbuah lebat. Buahnya bergantung pada lolosnya pohon dari **Juni sampai Oktober** tanpa cekaman kekeringan.

Air: sumur dengan pompa listrik yang mengangkat air ke **tandon 13.000 liter**, atap gudang 120 m^2 yang mengalir ke tandon itu, dan saluran tetes gravitasi ke setiap pohon setiap pagi, paling lambat sampai pukul 08.30.

September lalu tandonnya hampir kosong setiap pagi selama tiga minggu; panennya lebih kecil. Seorang penjual menawarkan tandon sekitar **Rp1.500 per liter**. Berapa liter?

Tidak ada rumus yang menjawabnya: hujan itu acak, sumur melemah, kebutuhan pohon bertambah, bentuk tandon menentukan tekanan. Masing-masing adalah masalah kecil yang dapat diselesaikan bab sebelumnya. Buku: Subbab 16.1.

Rencananya



Gambar 16.1 Sistem air kebun sebagai simulasi: setiap bagian dihitung dengan metode dari bab-bab sebelumnya, bagian-bagian itu bertemu di tandon, dan banyak musim simulasi menjadi masukan bagi pilihan ukuran tandon.

Satu hari demi satu hari: tentukan cuaca, tambahkan hujan dari atap, jalankan penyiraman pagi, catat apakah pohon mendapat jatahnya, pompa di sore hari.

Satu musim adalah satu masa depan yang mungkin. Ulangi untuk ratusan musim; pilih tandon yang baik di semuanya.

*Gambar 16.1: bagian-bagiannya bertemu di tandon. Disiplin yang membuat simulasi dapat dipercaya adalah membangunnya dari **bagian-bagian yang sudah diperiksa**: setiap bagian dihitung dengan dua cara sebelum satu musim pun dijalankan. Buku: Subbab 16.2.*

Setiap Bab Punya Tugas

Tabel 16.1 Tempat metode setiap bab bekerja dalam simulasi kebun.

Bagian	Pertanyaan	Metode (bab)
Tandon	Volume pada suatu ketinggian, ketinggian untuk suatu volume?	spline, LU tridiagonal, Newton, biseksi (7, 6, 3, 2)
Pompa	Berapa alirannya saat mengangkat ke tandon?	kuadrat terkecil, eliminasi, Newton (8, 4, 3)
Cuaca	Seberapa sering dan seberapa banyak hujan?	rantai Markov, metode pangkat, Gauss–Seidel (12, 5)
Catatan badai	Berapa hujan dari satu badai?	trapesium dan Simpson (10)
Saluran tetes	Berapa lama jatah itu keluar?	Runge–Kutta, regula falsi (11, 2)
Sensor ketinggian	Apakah model tetes cocok?	beda hingga, kuadrat terkecil, data yang disisihkan (9, 8, 14)
Musim	Bagaimana dengan banyak musim kemarau?	aliran acak berbenih, pekerja paralel, jumlah terkompensasi (1, 15)
Keputusan	Tandon mana yang termurah per tahun?	golden section (13)

Tabel 16.1 adalah mata kuliah ini dalam satu halaman. Berkas bersama `toolbox.py` / `Toolbox.java` menghimpun metode Bab 2 sampai 15 yang dipakai ulang oleh simulasi. Buku: Subbab 16.2.

Tandon dan Pompa

Tabel ember dari pembuat tandon memberi volume terhadap ketinggian untuk 13.000 L; ukuran lain menskalakannya:

$$h_C(V) = h\left(\frac{V}{C/13000}\right) \tag{16.1}$$

Lembar data pompa, dicocokkan dengan kuadrat terkecil, dan titik kerja tempat tinggi angkat pompa bertemu tinggi angkat sistem:

$$H_p(Q) = a + bQ + cQ^2 \tag{16.2}$$

$$g(Q) = H_p(Q) - H_s(Q) = 0 \tag{16.3}$$

$$E = \frac{1000}{60\,Q^*}P = \frac{1000}{60 \times 59.752} \times 0.8 = 0.2231\ \text{kWh per 1000 liters} \tag{16.4}$$

Persamaan (16.1): tandon yang lebih tinggi dengan bentuk yang sama. Persamaan (16.2): kuadratik yang dicocokkan pada lembar data. Persamaan (16.3) adalah masalah pencarian akar, diselesaikan dengan Newton. Persamaan (16.4): pada 0,8 kW, seribu liter berbiaya 0,2231 kWh, sekitar Rp322. Buku: Subbab 16.3.1–16.3.2.

Cuaca sebagai Rantai Markov

Tiga jenis hari — kering, hujan ringan, hujan lebat — dengan hari esok bergantung pada hari ini (baris berjumlah 1):

$$P = \begin{pmatrix} 0.88 & 0.10 & 0.02 \\ 0.60 & 0.32 & 0.08 \\ 0.50 & 0.35 & 0.15 \end{pmatrix} \tag{16.5}$$

$$\pi P = \pi, \pi_1 + \pi_2 + \pi_3 = 1 \tag{16.6}$$

Persamaan (16.5) memakai konvensi baris, berbeda dari Bab 12 — periksa jumlahnya. Persamaan (16.6): bagian jangka panjang setiap jenis hari, dengan metode pangkat dan dengan Gauss–Seidel. Banyaknya hujan pada hari hujan ringan diambil dari tabel kuantil catatan hujan. Buku: Subbab 16.3.3.

Saluran Tetes dan Neraca Harian

Aliran keluar bergantung pada tinggi air di atas penetes, ditambah tinggi dudukan tandon Z:

$$\frac{dV}{dt} = -K\sqrt{\frac{h(V)}{100} + Z}\,, K = 12\,\text{L/min per}\sqrt{\text{m}} \tag{16.7}$$

Setiap hari d: hujan r_d (mm) di atap, jatah pagi R_d, volume pompa sore W_d:

$$\begin{aligned} V' &= \min(C,\ V_d + 102\,r_d), \\ V'' &= V' - G_d, \qquad G_d = \text{drip session from } V' \text{ for } R_d \ (0 \text{ if } r_d \geq 10), \\ V_{d+1} &= V'' + \min(C - V'',\ W_d), \end{aligned} \tag{16.8}$$

Persamaan (16.7) adalah hukum tandon Bab 11; RK4 melangkahkannya, dan regula falsi menemukan saat jatah itu selesai. Persamaan (16.8): atap menangkap 102 liter per milimeter hujan, hari hujan 10 mm atau lebih tidak perlu disiram, dan kelebihan di atas kapasitas meluap. Buku: Subbab 16.3.4–16.3.5.

Menjalankannya: Setiap Bagian, Dua Cara

```
1. Tank: level for 6,500 liters (Chapters 2, 3, 6, 7)
   inverse spline h(V)           115.0174 cm
   Newton on V(h) = 6500         115.0144 cm in 4 steps
   bisection on V(h) = 6500      115.0144 cm
   straight lines through table  114.7929 cm
2. Pump: datasheet fit and operating point (Chapters 3, 4, 8)
   head = 31.9958 -0.023155 Q -0.00411310 Q^2  (m, Q in L/min)
   largest misfit on the datasheet 0.087 m
   operating point 59.752 L/min at 15.927 m
   1000 liters: 0.2231 kWh, Rp 322
3. Weather: long-run share of dry, light and heavy days (5, 7, 10, 12)
   power method, 40 steps   0.828813  0.138638  0.032550
   Gauss-Seidel, 13 sweeps   0.828813  0.138638  0.032550
```

Spline invers dan Newton pada spline maju berbeda 0,03 mm; garis lurus 2 mm — tidak berbahaya. Kuadratik pompa meleset dari lembar data paling banyak 9 cm tinggi angkat. Metode pangkat dan Gauss–Seidel sepakat sampai enam digit: 83% hari kering. Badainya terintegral menjadi 38,9 mm dengan trapesium dan 39,1 mm dengan Simpson. Buku: Subbab 16.5.1.

Menjalankannya: Sesi Penyiraman

```
5. Drip session for 2,520 liters (Chapters 2, 11)
   start   13000 L, euler dt 10.0:  2520.00 L in 114.5825 min
   start   13000 L, rk4   dt 10.0:  2520.00 L in 114.8071 min
   start   13000 L, rk4   dt 0.5:  2520.00 L in 114.8071 min
   start    4000 L, rk4   dt 10.0:  2520.00 L in 147.2514 min
   start    3000 L, rk4   dt 10.0:  2450.54 L in 150.0000 min
```

Baris-baris terpilih. Dari tandon penuh jatah itu butuh 114,8 menit; dari 4000 L, 147,3 menit, pas di dalam jendela 150 menit; dari 3000 L jendelanya tertutup dengan kekurangan 70 liter, padahal masih ada 550 liter di tandon — tekanannya terlalu rendah. RK4 dengan langkah 10 menit sama dengan langkah 0,5 menit sampai setiap digit; Euler meleset seperempat menit, berorde satu seperti diramalkan Bab 11. Buku: Subbab 16.5.1.

Model Tetes Melawan Sensor

```
46 readings, 0 to 90 min, level 175.5 -> 151.0 cm
step  2 min: 44 rates, K (first half) 11.819, test RMS 6.839 L/min, K (all) 11.856
step 10 min: 36 rates, K (first half) 11.910, test RMS 1.253 L/min, K (all) 11.899
model value K = 12.000 L/min per sqrt(m)
```

`sensor.py` menurunkan ketinggian yang dicatat (Bab 9), mengubahnya menjadi liter per menit, dan mencocokkan K dengan kuadrat terkecil pada separuh pertama, lalu mengujinya pada separuh kedua (Bab 8 dan 14). Dengan selisih 2 menit deraunya mendominasi, seperti diperingatkan Bab 9; dengan selisih 10 menit galat data yang disisihkan turun ke 1,25 L/menit dan K menjadi 11,9 — dalam 1% dari 12 milik model. Buku: Subbab 16.5.2.

Banyak Musim: Monte Carlo

$$\acute{X} = \frac{1}{N}\sum_{s=1}^{N} X_s, \mathrm{SE}(\acute{X}) = \frac{\sigma_X}{\sqrt{N}} \tag{16.9}$$

Kode 16.1 Dari `montecarlo.py`: setiap musim diberi benih dengan nomornya sendiri, sehingga hasilnya tidak bergantung pada banyaknya pekerja.

```
def chunk(job):
    capacity, first, count = job
    rows = []
    for i in range(first, first + count):
        tot = o.season(SEED0 + i, capacity)
        rows.append([tot[k] for k in KEYS])
    return rows
```

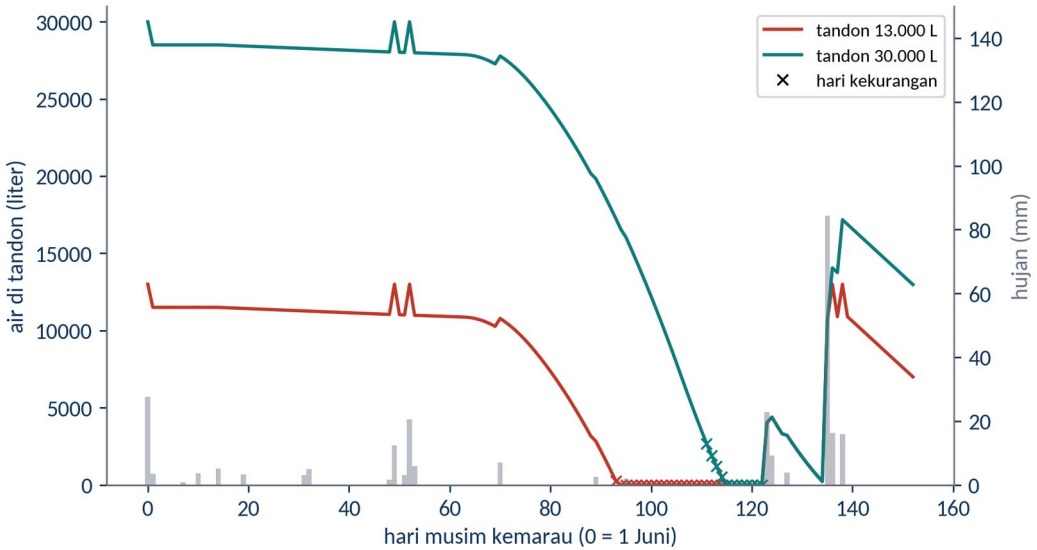
Persamaan (16.9): galat Monte Carlo hanya turun seperti $1/\sqrt{N}$ — empat kali banyaknya musim untuk separuh galat. Kode 16.1: benih berdasarkan nomor musim, bukan jam atau utas, adalah aturan keterulangan Bab 15; totalnya dijumlahkan dengan metode Kahan. Buku: Subbab 16.3.6 dan 16.4.

Menjalankannya: 13.000 Lawan 30.000 Liter

```
tank 13000 L, 400 seasons:
  rain caught     35870 L, overflow     20496 L
  pumped         258005 L, electricity 57.6 kWh = Rp 83192
  trees got      280146 L, missed 14166 L on 26.2 days on average
  seasons with any shortage: 390 of 400; 9 in 10 miss less than 24446 L
tank 30000 L, 400 seasons:
  rain caught     35870 L, overflow     19543 L
  pumped         258745 L, electricity 57.7 kWh = Rp 83431
  trees got      292307 L, missed 2005 L on 5.8 days on average
  seasons with any shortage: 145 of 400; 9 in 10 miss less than 7446 L
```

*Air atap, rata-rata 35.870 L, sepakat dengan taksiran tangan 35.757 dalam batas galat Monte Carlo — pemeriksaan pertama atas keseluruhan. Dengan tandon sekarang, **390 dari 400** musim kekurangan air. Dengan 30.000 L, 145 musim, dan kekurangan rata-rata turun dari 14.166 menjadi 2005 liter. Listriknya hampir tidak berubah: air yang diminum pohon toh harus dipompa. Buku: Subbab 16.5.3.*

Satu Musim, Dua Tandon



Gambar 16.2 Satu musim kemarau simulasi (cuaca yang sama) dengan tandon 13.000 liter dan 30.000 liter: air di tandon setelah penyiraman pagi, hari-hari ketika pohon mendapat kurang dari jatahnya (tanda silang), dan hujan harian (batang, sumbu kanan).

Gambar 16.2: sampai awal Agustus sumurnya masih sanggup. Lalu kedua tandon terkuras sekitar 900 liter sehari. Tandon kecil kosong pada 2 September dan pohon kekurangan setiap hari selama tiga minggu — persis pengalaman Pak Hadi tahun lalu. Tandon besar bertahan sampai satu masa kering pendek menjelang hujan pertama Oktober. Buku: Subbab 16.5.3.

Keputusannya: Biaya per Tahun

$$\text{cost}(C) = a\,C + e\,E[\text{kWh}] + p\,E[D(C)] \tag{16.10}$$

$a = 150$ rupiah per liter kapasitas per tahun (Rp1.500 selama sepuluh tahun), $e = 1445$ rupiah per kWh, $p = 500$ rupiah per liter yang tidak didapat pohon — taksiran Pak Hadi sendiri atas buah yang hilang.

Pada ukuran terbaik, satu liter tandon tambahan harus bernilai tepat sebesar yang dihematnya — **aturan newsvendor**:

$$a = p \cdot \Pr[D(C^*) > 0] \;\Leftrightarrow\; \Pr[\text{shortage with } C^*] = \frac{a}{p} = \frac{150}{500} = 0.30 \tag{16.11}$$

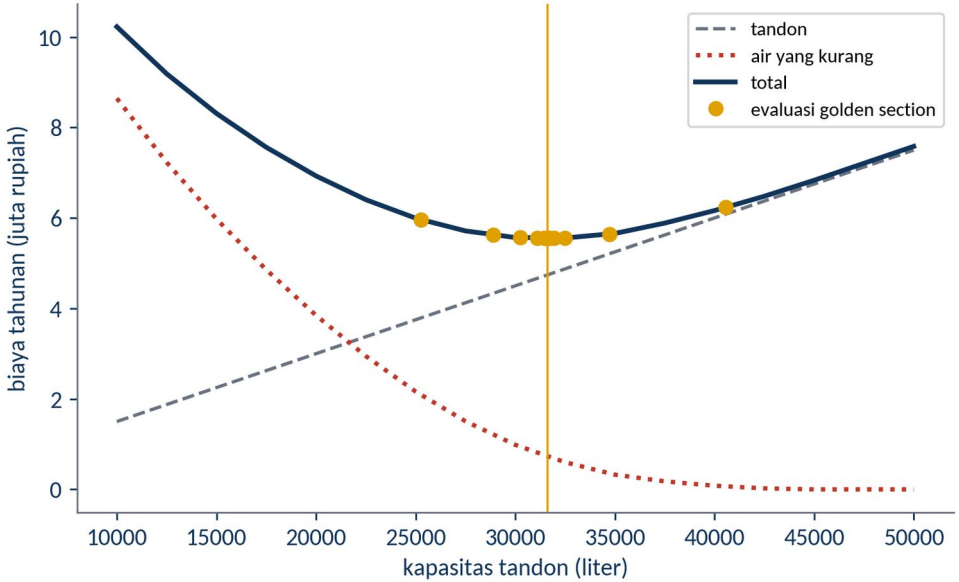
Persamaan (16.10): biaya tandon naik bersama C, air yang kurang turun. Nilai-nilai harapannya berasal dari simulasi, jadi setiap evaluasi adalah satu jalannya Monte Carlo dengan benih yang sama untuk setiap ukuran — bilangan acak bersama, agar perbandingannya bukan soal untung-untungan. Persamaan (16.11) meramalkan peluang kekurangan pada jawabannya sebelum kita menghitungnya. Buku: Subbab 16.3.7.

Menjalankannya: Golden Section pada Biaya

```
200 seasons per size; yearly cost in thousand rupiah
  tank (L)  P(short)   tank  power  missed   total
     25279     0.590   3792   83.4    2082    5958
     34721     0.175   5208   83.4     347    5639
     40557     0.060   6084   83.4      64    6231
     31115     0.315   4667   83.4     804    5555
...
     31564     0.305   4735   83.4     734    5552
best tank about 31602 L, yearly cost 5552 thousand; P(short) there 0.305
newsvendor check: P(short) should be near 0.300
```

*Dua belas kapasitas dicoba (baris-baris di antaranya tidak ditampilkan di sini). Sekitar **31.600 liter**, 5,55 juta rupiah setahun: 4,7 juta untuk tandon, 0,08 juta listrik, 0,7 juta buah yang hilang. Tandon sekarang akan kehilangan sekitar 7,1 juta buah, 9 juta seluruhnya. Peluang kekurangan pada optimumnya, 0,305, sesuai dengan 0,30 dari aturan newsvendor: teori dan simulasi saling membenarkan. Buku: Subbab 16.5.4.*

Optimum yang Datar



Gambar 16.3 Biaya tahunan terhadap kapasitas tandon dari 200 musim simulasi per ukuran: biaya tandon, nilai air yang kurang, dan totalnya, dengan kapasitas-kapasitas yang dicoba *golden-section search* dan ukuran yang dipilih.

> **WAWASAN KUNCI:** Biaya totalnya datar dari sekitar 29.000 sampai 34.000 liter. Laporkan “sekitar 30.000 liter”, bukan 31.602: datanya — tabel sumur lima angka, taksiran harga buah — tidak punya lima digit. Pak Hadi dapat membeli ukuran standar dalam rentang itu dan hampir tidak rugi apa pun.

Gambar 16.3: garis tandon naik terus, kurva air yang kurang turun tajam lalu mendatar, dan totalnya punya lembah yang lebar. Model memberi saran; Pak Hadi yang memutuskan. Buku: Subbab 16.5.4.

Membandingkan dengan SciPy

```
tank: level for 6500 L 115.0174 cm (ours 115.0174)
pump: polyfit 31.9958 -0.023155 Q -0.00411310 Q^2
      operating point 59.751951 L/min (ours 59.751951)
weather: eigenvector for eigenvalue 1 0.828813  0.138638  0.032550
storm: scipy simpson 39.1111 mm
drip: solve_ivp reaches 2520 L after 114.8071 min (ours 114.8071)
```

`CubicSpline`, `polyfit`, `brentq`, `eig`, `simpson`, dan `solve_ivp` dengan fungsi peristiwa menghitung ulang setiap bagian dan sepakat pada setiap digit yang dicetak. Simulasi yang dirakit dari bagian-bagian yang masing-masing sepakat dengan pustaka, dan yang bersama-sama sepakat dengan taksiran tangan dan aturan newsvendor, telah layak mendapat kepercayaan. Buku: Subbab 16.6.

Membuatnya Lebih Baik, dan Memakainya Sungguhan

Sisakan ruang untuk hujan. Lebih dari separuh air atap meluap karena pengisian sore menjaga tandon tetap penuh; isi hanya sampai 90% pada Agustus dan September.

Uji kepekaannya. Ubah hasil sumur bulan September, peluang cuaca, dan nilai buah yang hilang sebesar 10% lalu jalankan ulang `design.py`.

Modelkan sumur lebih baik. Ketinggian air sumur yang dicatat dapat dicocokkan sebagai persamaan diferensialnya sendiri.

Kebun lain, rumah tangga, sekolah, atau klinik kecil dengan tandon air hujan dapat memakai modelnya dengan mengubah konstanta bernama dan berkas data. Bagi rumah tangga yang membeli air dari truk seharga Rp30 per liter, $a/p = 5$ dan tidak ada tandon besar yang membayar dirinya sendiri dari sisi itu saja — aturan yang sama, jawaban yang berbeda. Buku: Subbab 16.7–16.8.

Metode Numerik dalam Praktik: Profitina

Platform AI Visibility Profitina, seperti kebun ini, adalah **sistem dari bagian-bagian**: satu pemindaian mengukur banyak sinyal, masing-masing dihitung dengan metodenya sendiri dan diperiksa sendiri-sendiri; sinyal-sinyal itu dinormalkan dan digabungkan dengan bobot menjadi skor.

Skornya diikuti dalam tren mingguan, dengan penjadwal yang memindai ulang setiap merek menurut irama tingkat langganannya.

“Periksa setiap bagian dengan metode yang independen sebelum memercayai keseluruhannya” adalah disiplin sebuah **gerbang rilis**: lebih dari seribu uji otomatis harus lulus, dan sebuah rilis baru dianggap berhasil ketika versi yang terpasang menjawab pemeriksaan kesehatannya.

Sinyal dan bobot persisnya milik Profitina; disiplinnya adalah bab ini. Seperti di setiap bab, angka memberi saran dan manusia yang memutuskan. Buku: Subbab 16.9.

Kesalahan Umum

Merakit sebelum memeriksa. Tanda yang salah dalam pencocokan pompa tetap mencetak angka yang tampak masuk akal.

Cuaca yang berbeda untuk setiap pilihan. Pakai benih yang sama untuk setiap ukuran, atau Anda membandingkan keberuntungan.

Memberi benih dari jam atau dari utas. Beri benih setiap musim dari nomornya.

Memercayai digit yang tidak dimiliki data. Laporkan rentang yang datar, bukan optimum yang persis.

Rata-rata saja. Laporkan juga peluang kekurangan dan persentil yang tinggi.

Setiap kesalahan ini sudah pernah kita temui, kini dalam lingkup yang lebih besar. Buku: Subbab 16.10.

Ringkasan

1. Pertanyaan nyata dijawab dengan simulasi yang dibangun dari metode seluruh mata kuliah (Tabel 16.1, Gambar 16.1).

2. Setiap bagian diperiksa dengan metode kedua: tandon (Persamaan (16.1)), pompa (Persamaan (16.2), Persamaan (16.3)), cuaca (Persamaan (16.5), Persamaan (16.6)), tetes (Persamaan (16.7)).

3. Neraca harian (Persamaan (16.8)) berjalan 153 hari per musim, 400 musim secara paralel, dapat diulang; galatnya turun seperti $1/\sqrt{N}$ (Persamaan (16.9)).

4. Tandon sekarang kekurangan air hampir di setiap musim; golden section pada Persamaan (16.10) memilih sekitar 31.600 liter, datar dari 29.000 sampai 34.000.

5. Peluang kekurangan di sana, 0,305, sesuai dengan aturan newsvendor (Persamaan (16.11)); Java, Python, dan SciPy sepakat.

Dari struk Bu Wati di Bab 1 sampai tandon Pak Hadi, pertanyaan yang sama mengalir di setiap metode: seberapa salah ini dapat terjadi, seberapa cepat ia konvergen, dan bagaimana kita tahu? Buku: Subbab 16.11.

Soal-soal Latihan

Cari distribusi cuaca stasioner dengan tangan dari persamaan keseimbangan dan syarat jumlah, lalu periksa dengan program.

Hentikan pengisian sore pada 90% kapasitas di Agustus dan September lalu jalankan ulang `montecarlo.py` untuk 30.000 liter: apa yang terjadi pada luapan dan air yang kurang?

Tanpa menjalankan apa pun, pakai aturan newsvendor untuk nilai buah Rp300 atau Rp1.000 per liter.

Soal lain menanyakan berapa musim yang memberi peluang kekurangan dengan galat baku 0,01, dan berapa lama itu dalam masing-masing bahasa. Penyelesaiannya ada di buku. Buku: Subbab 16.12–16.13.