Föreläsningsanteckningar i
Numerisk analys
MMG410 (MAN200, MAM240)
Thomas Ericsson, Matematik
Chalmers / GU
2007

Vad är numerisk analys?

Numerisk analys handlar om hur man löser beräkningsproblem på ett säkert och effektivt sätt med hjälp av dator.

Några viktiga komponenter:

- Problemets egenskaper
  - Problemen kommer från naturvetenskap, teknik, matematik etc.
  - Existerar det någon lösning?
  - Är den entydig?
  - Vad händer med lösningen när man ändrar indata något?
- Algoritmens egenskaper:
  - Hur snabb är metoden, implementationen?
  - Hur mycket minne går åt?
  - Vilka fel introduceras av algoritmen (avrundningsfel etc)?
- Beräkningsredskapets egenskaper:
  - Datorarkitektur. Möjligheter/begränsningar.
  - Parallellitet.
  - Programspråk och kompilatorer.

På följande sidor kommer några korta exempel på ovanstående.

Är det viktigt att räkna exakt?

Flertalet verkliga problem är för komplicerade att lösa exakt och även om det går så är det kanske inte intressant.
Maple kan beräkna:

$$\int x^{10}e^x\,dx = \begin{array}{l}\big[3628800 - 3628800\,x + 1814400\,x^2 - 604800\,x^3 + \\ 151200\,x^4 - 30240\,x^5 + 5040\,x^6 - 720\,x^7 + 90\,x^8 - \\ 10\,x^9 + x^{10}\big]\,e^x + \text{konstant}\end{array}$$

och

$$\int_0^1 x^{10}e^x\,dx = 1334961\,e - 3628800$$

Jämför i Matlab

```
>> i = quadl('x.^10 .* exp(x)', 0, 1)
i = 2.2800151548782251e-01
```

med ett fel mindre än $4 \cdot 10^{-10}$. I en verklig tillämpning duger det kanske med fyra siffror.

$$\int_0^1 \arctan(e^x)\,dx$$

kan ej uttryckas på något enkelt sätt. Dock

```
>> i = quadl('atan(exp(x))', 0, 1)
i =
     1.017430583002258e+00
```

Dessa problem är enkla jämfört med många verkliga problem. Ett system av differentialekvationer kan normalt endast lösas approximativt och även det kan vara svårt (väderprognoser).

Hur många siffror behöver man i tillämpningar?
Några exempel:

Säg att vi skall tillverka en stålstav med längden $1m$. Hur många siffror är det rimligt att ange? De finaste passbitarna (dyra och mycket noggrant polerade mätblock) har en avvikelse på omkring $\pm 10^{-6}m$. En väteatom har en storlek $\approx 10^{-10}m$.
Typen `double` i Java, C etc. tar 64 bitar, drygt 16 decimaler.

Vanliga elektriska motstånd har en osäkerhet om 5-10%, 0.1% för precisionsmotstånd.

Från NIST (National Institute of Standards and Technology) Special Publication 333, 2001 Edition, The International System of Units (SI) `http://physics.nist.gov/Pubs/pdf.html`

The unit of mass, the kilogram, is the mass of the international prototype of the kilogram kept at the BIPM (Bureau International des Poids et Mesures). It is a cylinder made of an alloy for which the mass fraction of platinum is 90 % and the mass fraction of iridium is 10 %. The masses of 1 kg secondary standards of the same alloy or of stainless steel are compared with the mass of the international prototype by means of balances with a relative uncertainty approaching 1 part in 10^9.

The mass of the international prototype increases by approximately 1 part in 10^9 per year due to the inevitable accumulation of contaminants on its surface. For this reason, the CIPM (Comité International des Poids et Mesures) declared that, pending further research, the reference mass of the international prototype is that immediately after cleaning and washing by a specified method (PV, 1989, 57, 104-105 and PV, 1990, 58, 95-97). The reference mass thus defined is used to calibrate national standards of platinum-iridium alloy (Metrologia, 1994, 31, 317-336).

In the case of stainless-steel 1 kg standards, the relative uncertainty of comparisons is limited to about 1 part in 10^8 by the uncertainty in the correction for air buoyancy. The results of comparisons made in vacuum, though unaffected by air buoyancy, are subject to additional corrections to account for changes in mass of the standards when cycled between vacuum and atmospheric pressure.

Mass standards representing multiples and submultiples of the kilogram can be calibrated by a conceptually simple procedure.

Slutsats: det räcker oftast med några få siffror.

Att förstå sitt problem

Ett generaliserat egenvärdesproblem: $Ax = \lambda Bx$.
A och B är stora och glesa matriser.

Saab: Ditt program gör fel. Vi får olika egenvärden varje gång.

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Vi ser att

$$\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

men

$$\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

för varje komplext tal λ. Singulärt matrisknippe.

Låt oss störa matriserna:

$$A = \begin{bmatrix} 2 & \epsilon \\ \epsilon & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & \delta \\ \delta & 0 \end{bmatrix}$$

Sekularekvationen, $\det(A - \lambda B) = 0$, blir:

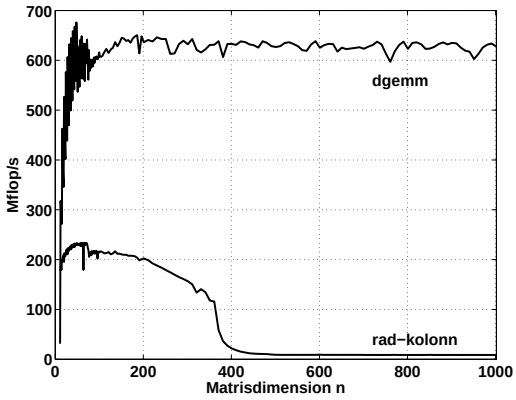
$$(\delta\lambda - \epsilon)^2 = 0$$

så att det dubbla egenvärdet är $\lambda = \epsilon/\delta$ för alla $\delta \neq 0$ och ϵ oavsett hur små de är.


```
>> A          % Ett linjärt ekvationssystem
A =
   -0.1537    0.8538    0.6535    0.1342
    0.6678   -0.1268   -0.1732   -0.1248
    1.5560   -1.0140   -1.0986   -0.5522
   -0.1248    0.0686    0.3664    0.3467

>> b
b =
    0.2042
   -0.1849
   -0.0358
   -0.9607

>> x = A \ b     % lös A x = b
x =
   -0.7033
    1.4745
   -1.4029
   -1.8333

>> f             % mätfel kanske
f =
   1.0e-10 *     % OBS: 1e-10
    0.3399
   -0.8218
    0.4035
    0.2154

>> A \ (b + f)
ans =
   1.0e+04 *     % OBS: 1e4
   -0.1257
   -2.1861
    3.5091
   -3.3217
```


Följande bild visar prestanda för två rutiner för matrismultiplikation, $A = BC$, där samtliga ingående matriser är kvadratiska med dimension n (x-axeln).

y-axeln visar antalet miljoner flyttalsoperationer (+, *) per sekund som flyttalsenheten (FPUn) i CPUn presterar. "dgemm" är hämtad från IBMs beräkningsbibliotek, ESSL (Engineering Scientific Subroutine Library). Rutinen ligger nära maskinens teoretiska toppfart.
"rad-kolonn" är algoritmen från kursen i linjär algebra. Det är inget fel på denna metod när man handräknar, men den är hopplöst ineffektiv på moderna datorer. Det beror på att FPUn är mycket snabbare än primärminnet: FPUn får vänta på matriselement att räkna med. I dgemm-rutinen har accessmönstret av matriselementen ändrats så att cache-minnena kan utnyttjas mer effektivt, vilket leder till inga eller korta väntetider.
Kodning blir mer komplicerad, vilket är en av flera anledningar till att vi inte ska studera verkliga implementationer.



Mål med kursen: numerisk allmänbildning.

Kunna svara på frågor som (för några problemklasser):

- Hur fungerar numeriska algoritmer?
- Hur ser typisk numerisk programvara ut?
- Vilka typer av problem, inom problemklassen, är möjliga att lösa?
- Är problemet svårt eller enkelt?
- Vilken programvara kan jag välja bland?
- Kan jag lita på resultatet?
- Tog beräkningen rimlig tid?
- Gick det åt för mycket minne?
- Hur stort problem av denna typ kan jag lösa?

Målet är inte att Du skall kunna skriva numerisk programvara. Vi kommer att studera principer, inte verkliga implementationer eftersom dessa är alltför tekniskt komplicerade.

Kursinnehåll

1. Konditionstal, stabilitet
2. Flyttalsaritmetik
3. $Ax = b$
4. Minstakvadratproblem
5. System av ickelinjära ekvationer
6. Interpolation
7. Kvadratur
8. Ordinära differentialekvationer
9. HPC (High Performance Computing)

Laborationer:

1. Flyttalsaritmetik, $Ax = b$
2. Minstakvadratproblem samt $f(x) = 0$
3. Kvadratur och ODE

Diverse felkällor

Fel som vi som numeriker inte kan göra så mycket åt

- modellfel, bortser från luftmotstånd, friktion
- mätfel, vågar etc. är inte exakta mellanavrundningar

Vi är intresserade av olika typer av beräkningsfel:

Avrundningsfel:

```
>> 49 * (1 / 49) - 1
ans = -1.1102e-16
```

Trunkeringsfel : $e^x \approx \sum_{k=0}^{N} \frac{x^k}{k!}$

Diskretiseringsfel : $f'(x) \approx \frac{f(x+h) - f(x)}{h}$

Viktigt att välja "lagom stort" h.

Om stort och smått

Låt $\hat{x}$ vara en approximation av det exakta värdet x.

- absoluta felet $= \hat{x} - x$
- relativa felet $= \frac{\hat{x}-x}{x}$, om $x \neq 0$

Absoluta fel är ointressanta om vi inte vet ungefär hur stort x är.

Är 1.4 ett stort absolut fel? Ja, om det exakta värdet är 2, men inte om det exakta värdet är 10^9.
De relativa felen är 0.7 respektive $1.4 \cdot 10^{-9}$.

På samma sätt kan det absoluta felet 10^{-20} vara stort eller litet. Det är viktigt att känna till problemets skalning.
Vilken storleksordning har de tal vi arbetar med?

Relativa fel säger något även om vi inte känner till problemets skalning. Vi kommer därför att vara mer intresserade av relativa fel än av absoluta fel.

Tema: nollställen till polynom

Beräkna rötterna till $(x-1)^5 = 0$ i Matlab (där vi räknar med 16 siffror). Matlab vill ha en vektor med koefficienter:

$$(x-1)^5 = x^5 - 5\,x^4 + 10\,x^3 - 10\,x^2 + 5\,x - 1$$

```
>> r = roots([1 -5 10 -10 5 -1]) % koefficienter

r = 1.0008 + 0.0006i              % rötterna
    1.0008 - 0.0006i
    0.9997 + 0.0009i
    0.9997 - 0.0009i
    0.9990

>> abs(r - 1)                     % felen?
ans = 1.0e-03 *
       0.9625
       0.9625
       0.9625
       0.9625
       0.9625
```

Varför? Lös

$$(x-1)^5 = \epsilon \quad \Rightarrow \quad x = 1 + \epsilon^{1/5}$$

Om $\epsilon = 10^{-15}$ så är $\epsilon^{1/5} = 10^{-3}$. Nollställena till polynomet $(x-1)^5$ är tydligen känsliga för störningar i koefficienterna.

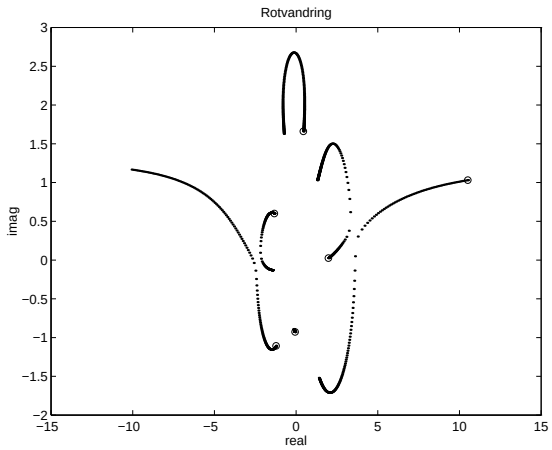
Är det alltid svårt att beräkna nollställen?

```
>> c = [1 -15 85 -225 274 -120];  % koefficienter
>> r = roots(c);  % de exakta röttena = 1, 2, 3, 4, 5
>> fel = sort(r) - (1:5)'
fel =
  -4.9960e-15
   6.6613e-14
  -1.5010e-13
   9.6811e-14
  -8.8818e-16
```

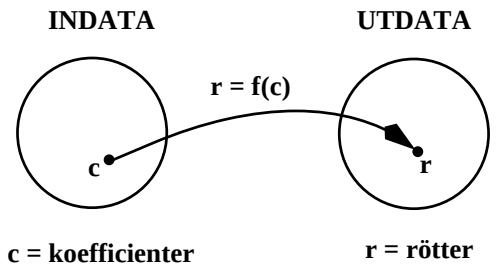
Rötter hos ett komplext 6-egradspolynom när vi stör x^5-termen.

$$x^6 + (c_5 + k\ \delta)x^5 + \cdots + c_0, \quad k = 0, 1, 2, 3 \ldots$$

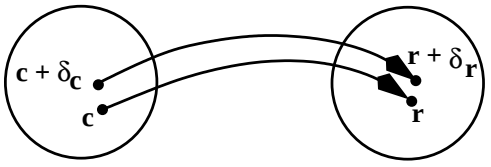
Ringarna visar rötternas begynnelsepositioner, $k = 0$.



Vi kan betrakta rötterna som funktioner av koefficienterna.



Vad händer när vi stör koefficienterna (indata i det allmänna fallet)? Det är viktigt att känna till om vårt problem är känsligt eller ej.



Om liten relativ ändring av indata ger en liten relativ ändring av resultatet säger man att det aktuella problemet är välkonditionerat. Om resultatet ändrar sig mycket är problemet illakonditionerat.

Konditionstalet är kvoten mellan de relativa förändringarna, dvs.

$$konditionstalet = \frac{|\delta_r|/|r|}{|\delta_c|/|c|}$$

Att beräkna konditionstalet är inte alltid möjligt; det kan vara lika svårt som att lösa det egentliga problemet. För vissa problemtyper är det överkomligt. Ibland är det dock möjligt att konstruera en uppskattning κ så att

$$\frac{|\delta_r|}{|r|} \leq \kappa \frac{|\delta_c|}{|c|}$$

Det räcker att känna till storleksordning på κ.
Är $\kappa \approx 10$ eller är $\kappa \approx 10^8$?

Exempel:
Hur känsliga är rötterna, till ekvationen $x^2 + ax + b = 0$, för ändringar i a och b?

Rötterna r_1 och r_2 är funktioner av a och b; $r_1(a,b)$, $r_2(a,b)$.

Låt r beteckna en av rötterna och låt $r + \delta_r$ beteckna den störda roten när vi ändrar koefficienterna med δ_a respektive δ_b. Vi har sambandet:

$$(r+\delta_r)^2 + (a+\delta_a)(r+\delta_r) + b + \delta_b = 0$$

Detta kan skrivas

$$\underbrace{r^2 + ar + b}_{0} + \delta_r(2r + a) + \delta_a r + \delta_b + \underbrace{\delta_r^2 + \delta_a\delta_r}_{\approx 0} = 0$$

Vi får det approximativa sambandet:

$$\delta_r \approx -\frac{\delta_a r + \delta_b}{2r + a} \quad \Rightarrow \quad |\delta_r| \lessapprox \frac{|\delta_a r| + |\delta_b|}{|2r + a|}$$

Eftersom r_1 och r_2 är rötter så gäller att:

$$\underbrace{(x - r_1)(x - r_2)}_{x^2 - (r_1 + r_2)x + r_1 r_2} \equiv x^2 + ax + b \;\Rightarrow\; -(r_1 + r_2) = a$$

alltså är $2r_1 + a = r_1 - r_2$, gapet. Låt $g = |r_1 - r_2|$, då gäller:

$$|\delta_r| \lessapprox \frac{|\delta_a r| + |\delta_b|}{g}$$

Slutligen vill vi ha relativa störningar:

$$\frac{|\delta_r|}{|r|} \lessapprox \frac{1}{|r|}\left[\frac{|r||a|}{g}\frac{|\delta_a|}{|a|} + \frac{|b|}{g}\frac{|\delta_b|}{|b|}\right] \leq \underbrace{\frac{|a| + |b/r|}{g}}_{\approx\ konditionstalet} \max\left[\frac{|\delta_a|}{|a|}, \frac{|\delta_b|}{|b|}\right]$$

Observera att detta är en uppskattning av konditionstalet. Det är inte heller beräkningsbart eftersom vi måste känna r_1 och r_2. I praktiken kan man (kanske) uppskatta r_1 och r_2 med de beräknade rötterna. En viktig lärdom är att vi nu vet att gapet mellan rötterna är viktigt.

Bakåtanalys

Vi har sett s.k. framåtanalys: givet δ_c vad blir

$$f(c + \delta_c) - f(c)$$

Detta kan, som vi har sett, ge väldigt pessimistiska svar.

Ett alternativ är följande: givet approximationen $\hat{r}$ till det exakta värdet r hur mycket måste vi ändra c för att $\hat{r}$ skall bli en exakt lösning till det störda problemet? Vi söker alltså δ_c sådant att

$$f(c + \delta_c) = \hat{r}$$

Man kallar detta bakåtanalys. Detta på grund av att vi tittar på indatasidan i stället för på resultatsidan.

Exempel: Låt

$$p(x) = (x - 1)(x - 1.0001) = x^2 - 2.0001\ x + 1.0001$$

Vi vet sedan tidigare att konditionstalet har storleksordningen $1/(1.0001 - 1) = 10^4$.

Antag att vi på något sätt har producerat de dåliga approximativa rötterna 1.11 och 0.895. De relativa felen är ungefär 11%.

Det störda polynomet (som har rötterna 1.11 och 0.895) är:

$$(x - 1.11)(x - 0.895) = x^2 - 2.005x + 0.99345 \;\Rightarrow\; \begin{cases} |\delta_a| \leq 5 \cdot 10^{-3} \\ |\delta_b| \leq 7 \cdot 10^{-3} \end{cases}$$

Detta innebär att vi har löst "nästan rätt problem"; vi har gjort ett relativt bra jobb med att beräkna rötterna. Att våra rötter är dåliga approximationer beror på att problemet är illakonditionerat.

Stabila algoritmer

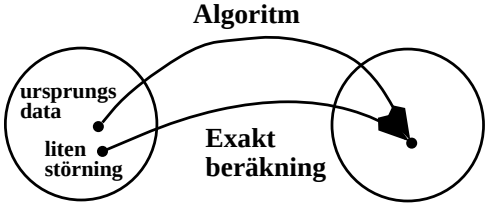
Man kan ofta lösa ett beräkningsproblem på olika sätt.

$$x^2 + ax + b = 0 \quad \text{har rötterna} \quad -\frac{a}{2} \pm \sqrt{\left(\frac{a}{2}\right)^2 - b}$$

Får vi ett bra program om vi skriver in ovanstående formler i koden? Nej, inte alltid. I en dator räknar vi med begränsat antal siffror. Om a är mycket större än b så kanske inte b kommer med när vi beräknar $(\frac{a}{2})^2 - b$. Detta svarar mot att $b = 0$ så att en beräknad rot blir noll. I detta fall är gapet stort så att rötterna är välkonditionerade.

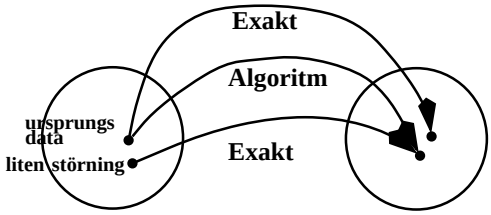
Det är algoritmen det är fel på! Det är i allmänhet ingen bra idé att kopiera formler direkt ur böcker.
Det finns bättre algoritmer; se övningarna.

En stabil algoritm genererar resultat som är exakta för ett lite stört problem.

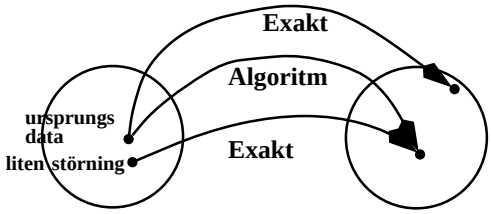


Om vi löser ett problem med en stabil algoritm får vi då ett resultat med ett litet fel?
Det beror på om det lilla bakåtfelet förstoras eller ej.

> Välkonditionerat problem + stabil algoritm ⇒ litet fel i resultatet.



Om vi applicerar en stabil algoritm på ett illakonditionerat problem löser algoritmen fortfarande nästan rätt problem.
Det lilla felet i indata kan dock ge upphov till ett stort fel i resultatet.



Flyttalsaritmetik

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e, \;\; 0 \le d_k \le \beta - 1, \;\; L \le e \le U$$

β bas (vi kommer att ha $\beta = 2$)
t precision
$[L, U]$ exponentomfång
$d_0 d_1 \cdots d_{t-1}$ mantissa

IEEE 754 (Institute of Electrical and Electronics Engineers, Inc.) definierar enkel och dubbel precision (bland annat).
Vi antar att $\beta = 2$ från och med nu:

bas	t	L	U	
2	24	-126	127	32 bitar
2	53	-1022	1023	64 bitar

Ett tal $\neq 0$ är normaliserat om $d_0 \neq 0$.
Om $\beta = 2$ så är då $d_0 = 1$ varför man inte lagrar d_0.

I minnet lagras ett 64-bitars flyttal med en teckenbit, en exponentdel till vilken man adderat 1023 (för att slippa tecken på exponentdelen), samt mantissan (utom den inledande ettan).

Exempel: -3.25 är $(-1)^1 \cdot 1.101 \cdot 2^1$ i binär form.
Teckenbiten är ett, exponenten (ett) lagras som $1 + 1023 = 2^{10}$ och av mantissan lagrar vi 101, varför vi bör hitta följande bitar i minnet:

```
1  100 0000 0000         1010 0000 ... 0000 0000
s  exponent 11 bitar     mantissa 52 bitar lagras
```

Hexadecimalt (bas 16) gruppera fyra bitar / hex-siffra

```
c00a000000000000  (a <-> 1010, c <-> 1100)
```

a=10, b=11, c=12, d=13, e=14 och f=15.

I Matlab

```
>> format hex
>> -3.25
ans = c00a000000000000
```

Det finns speciella bitformat för diverse specialfall, t.ex.:

```
>> [0, -0]
ans = 0000000000000000   8000000000000000
```

Antalet olika tal är: $2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$
dvs $4.2614 \cdot 10^9$ i enkel precision och $1.8429 \cdot 10^{19}$ i dubbel.
Att testa en funktion för alla flyttal i dubbel precision är nästan omöjligt. 10^9 tester per sekund ger 584.4 år.

Minsta positiva representerbara talet $= 2^L \approx 1.17 \cdot 10^{-38}$ i enkel och $\approx 2.2 \cdot 10^{-308}$ i dubbel.

Det största representerbara talet har största exponenten och ettor i hela mantissan. I enkel precision $\approx 3.4 \cdot 10^{38}$, i dubbel $1.8 \cdot 10^{308}$. Tal större än denna gräns ger overflow.

```
>> 1e-200^2        % e har INGET med exp att göra
ans = 0            % underflow

>> 1e200^2         % overflow
ans = Inf          % Infinity

>> log(0)
ans = -Inf         % -Infinity

>> sin(1/0)
ans = NaN          % Not a Number
```


Ett enkelt talsystem med $\beta = 2, t = 4, L = -3, U = 3$.

Samma system inzoomat kring nollan.

Att lägga märke till: ett minsta positivt tal, ett största tal, ej kontinuerligt, grupperat, gap kring nollan, ökande avstånd mellan talen.

Vad är avståndet, gapet, mellan två grannar?

Det absoluta gapet ökar med talets storlek.
I dubbel precision är det absoluta gapet $\approx 2.2 \cdot 10^{-16}$ kring ettan, men $\approx 2 \cdot 10^{292}$ kring overflowgränsen.

Den relativa skillnaden, gapet dividerat med talet, varierar något med talet, men är ungefär $1.1 \cdot 10^{-16}$ till $2.2 \cdot 10^{-16}$ i dubbel precision.

Vi räknar med ungefär 16 decimala decimaler i dubbel precision.

Exempel:
för vilka $k \ge 0$ kan 10^k lagras exakt i dubbel precision?

10^{22} kan representeras exakt.
10^{23} lagras som 99999999999999991611392.

Det absoluta felet verkar "stort":

$99999999999999991611392 - 10^{23} = -8388608$.

Det är nu inte så farligt som det verkar, eftersom det relativa felet är:

$8388608/10^{23} \approx 8.3 \cdot 10^{-17}$.

Negativa exponenter är "svårare", 0.1 kan t.ex. inte lagras exakt i binärt format med ändlig mantissa.

$$\sum_{k=1}^{\infty} \left[\frac{1}{2^{4k}} + \frac{1}{2^{4k+1}} \right] = \frac{3}{2} \sum_{k=1}^{\infty} \frac{1}{2^{4k}} = \frac{3}{2} \sum_{k=1}^{\infty} \frac{1}{16^k} = \frac{3}{2} \, \frac{1}{16} \, \frac{1}{1 - 1/16} = 0.1$$

varför den binära framställningen av 0.1 kan skrivas;

$$0.0001100110011001100110011\ldots$$

Detta är inget konstigt. $1/3 = 0.3333\cdots$ i basen 10 men 0.1 i basen 3.

Om x är ett godtyckligt reellt tal betecknar vi det avrundade flyttalet med $fl(x)$ (floating). Normalt (kan ändras) är $fl(x)$ det flyttal som ligger närmast x.

Exempel: Låt oss anta att vi räknar decimalt med fyra siffror.
$fl(\pi) = fl(3.141592653589...) = 3.142$.
$fl(31415926.53589...) = 3.142 \cdot 10^7$.

Hur stort kan det absoluta felet bli vid avrundning till närmaste flyttal? Maximalt en halv enhet i fjärde siffran. Så om vårt tal är $\pm s_1.s_2s_3s_4 \ldots \cdot 10^e$ (där $s_1, s_2, \ldots$ betecknar decimala siffror) är absolutbeloppet av absoluta felet maximalt $0.0005 \cdot 10^e$.

Relativa felet är maximalt (för ett normaliserat tal)

$$\left|\frac{fl(x) - x}{x}\right| \leq \frac{0.0005 \cdot 10^e}{1.0000 \ldots \cdot 10^e} = 0.0005$$

Denna begränsning kallas relativa maskinnoggrannheten och betecknas med ϵ_{mach}. Denna kvantitet bestäms av hur många siffror vi har i mantissan. Många siffror ger ett litet ϵ_{mach}.

Observera att ϵ_{mach} INTE HAR NÅGOT med exponentomfånget att göra (hur stora eller små tal vi kan arbeta med). Exponentomfånget ges ju av parametrarna L, U.

Om vi antar att vi räknar decimalt och $U = 100$ är det största talet vi kan lagra $9.999 \cdot 10^{100}$.

Om $L = -100$ är det minsta positiva representerbara talet $1.000 \cdot 10^{-100}$. Det minsta representerbara talet är INTE 0.0005.

I dubbel gäller att $\epsilon_{mach} = \frac{1}{2^t} \approx 1.11 \cdot 10^{-16}$ (16 siffrorna!) och i enkel precision $\epsilon_{mach} \approx 6 \cdot 10^{-8}$.

Vi kan skriva

$$\left|\frac{fl(x) - x}{x}\right| \leq \epsilon_{mach}$$

på ett annat sätt. Det gäller att:

$$fl(x) = (1 + \epsilon)x = x + \epsilon x \quad \text{med} \quad |\epsilon| \leq \epsilon_{mach}$$

OBS OLIKA ϵ.

Varför gäller detta? Antag $x \neq 0$.

$$fl(x) = (1 + \epsilon)x \Leftrightarrow fl(x) - x = \epsilon x \Rightarrow \underbrace{\left|\frac{fl(x) - x}{x}\right|}_{\leq \epsilon_{mach}} = |\epsilon|$$

OK även om $x = 0$ ty $fl(0) = 0$.

Enligt IEEE skall en enstaka `+, -, *, /` avrundas korrekt. Låt $\otimes$ beteckna någon av dessa operationer och låt x och y vara två flyttal. Då gäller att:

$$fl(x \otimes y) = (1 + \epsilon)(x \otimes y) = x \otimes y + \epsilon(x \otimes y), \quad |\epsilon| \leq \epsilon_{mach}$$

förutsatt att vi inte får `Inf` eller `NaN`.

Beloppet av absoluta felet vi denna beräkning är alltså:

$$|fl(x \otimes y) - (x \otimes y)| = |\epsilon(x \otimes y)| \leq \epsilon_{mach}|x \otimes y|$$

och det relativa felet (om $x \otimes y \neq 0$):

$$\left|\frac{fl(x \otimes y) - (x \otimes y)}{x \otimes y}\right| = |\epsilon| \leq \epsilon_{mach}$$

Talsystemet igen:

Lucka kring nollan. Offra "decimaler" för att få större exponentomfång. Vi använder denormaliserade eller subnormala tal.

```
1.010010011100 ... * 2^e         normaliserat
0.000010100011 ... * 2^(-1022)   denormaliserat
```

Har färre bitar i mantissan. Man talar om "gradual underflow".

Några vanliga problem med flyttalsräkning:

Utskiftning

Antag att vi räknar i dubbel precision:

```
>> a = 1e16;
>> b = a;                    % spara
>> a = a + 1
a = 1.000000000000000e+16

>> a - b
ans = 0
```

ger ingen förändring, ettan "trillar över kanten".

```
>> a = 1e16;
>> a = a + 123
a = 1.000000000000012e+16
>> a - b
ans = 124
```

Inte hela 123 kommer med.

Man bör undvika att addera eller subtrahera tal av mycket olika storleksordning.

$a + (b + c)$ behöver inte vara lika med $(a + b) + c$.
En kompilator får inte optimera för mycket.

```
>> 1 + (1e16 + (-1e16))
ans = 1

>> (1 + 1e16) + (-1e16)
ans = 0
```

Kancellation - subtraktion av två nästan lika stora tal

Antag att vi subtraherar följande två tal:

```
1.03678947f  f betecknar ett fel, en osäker siffra
1.03678935g  g betecknar ett fel, en osäker siffra
-----------
0.00000012t  t nytt fel
```

I de två första talen kommer felet i tionde siffran. I skillnaden finns felet redan i tredje siffran. Vi har alltså mycket större relativ osäkerhet i skillnaden än i någon av de ingående termerna. Vi känner alltså termerna mycket bättre än skillnaden; vi har förlorat information.

Vi får denna osäkerhet även om vi utför subtraktionen exakt.

Matrisfaktoriseringar

Vanligt i tillämpningar och teoretiskt arbete att skriva matriser som produkter av andra matriser (kallas matrisfaktoriseringar eller uppdelningar). Några exempel illustrerade med små "kryssmatriser":

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}}_{U}$$

L för "Lower triangular", undertriangulär och
U för "Upper triangular", övertriangulär.

Kallas LU-faktorisering. Används för att lösa $Ax = b$-problem. Matlab-kommando `lu`.

För att approximativt lösa överbestämda ekvationssystem (minstakvadratproblem) använder vi QR-faktorisering (`qr` i Matlab).

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}}_{R}$$

där Q är ortogonal, dvs. $Q^TQ = I$.

Om A är en sk diagonaliserbar matris kan vi använda Matlabs `eig`-kommando för att beräkna:

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{X^{-1}}$$

$\lambda_1, \lambda_2, \lambda_3$ är A:s egenvärden och de tre kolonnerna i X är motsvarande egenvektorer. Om A är en reell och symmetrisk matris så kan egenvektorerna väljas ortonormerade varför X är ortogonal och $X^{-1} = X^T$.

Singuläravärdesfaktoriseringen (i Matlab `svd`) är en slags generalisering av egenvärdesuppdelningen till ickekvadratiska matriser.

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{V^T}$$

där $U^TU = I$, $V^TV = I$ och $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Faktoriseringen existerar även för liggande matriser.

I numerisk analys har man dessutom varianter av en faktorisering beroende på matrisens egenskaper.
Detta för att spara datorminne och beräkningstid.

Några exempel för LU-faktoriseringen.

Om A är symmetrisk, $A^T = A$, så behövs halva minnet och beräkningstiden. Om A dessutom är positivt definit (positiva egenvärden) kan man förenkla metoden ytterligare.

Det finns motsvarande varianter om matrisen är komplex.

- $A^H = A$, A är Hermitsk ($A^H = \bar{A}^T$)
- $A^T = A$, A är komplexsymmetrisk

Många element i en matris kan vara noll, en gles matris. Detta kan man utnyttja för att spara minne och beräkningstid. Det är viktigt eftersom glesa matriser brukar vara stora, med en dimension om $10^4 - 10^6$ kanske. Ett specialfall av en gles matris är en sk bandmatris, här två små exempel:

$$\begin{bmatrix} \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \qquad \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ 0 & \times & \times & \times & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Den högra matrisen är ett exempel på en sk tridiagonal matris. Den kan vara osymmetrisk, symmetrisk, symmetrisk positivt definit, Hermitsk, komplexsymmetrisk etc. Dessutom brukar det finnas stöd för enkel- och dubbel precision liksom för komplex och dubbel komplex.

Man inser att det kan bli många varianter av LU-faktoriseringen (många olika rutiner i programbiblioteken, t.ex. Lapack).

LU-faktorisering

Now [let] there be
hemp 1 dŏu, wheat 3 dŏu, beans 2 dŏu, peas 8 dŏu, millet 5 dŏu, worth 95 coins.
hemp 2 dŏu, wheat 5 dŏu, beans 3 dŏu, peas 9 dŏu, millet 4 dŏu, worth 112 coins;
hemp 3 dŏu, wheat 5 dŏu, beans 7 dŏu, peas 6 dŏu, millet 4 dŏu, worth 116 coins;
hemp 7 dŏu, wheat 6 dŏu, beans 4 dŏu, peas 5 dŏu, millet 3 dŏu, worth 128 coins;
hemp 9 dŏu, wheat 7 dŏu, beans 3 dŏu, peas 2 dŏu, millet 5 dŏu, worth 140 coins;
Question: how much is 1 dŏu [of each] worth?

The answer says:
hemp 1 dŏu 7 coins,
wheat 1 dŏu 4 coins,
beans 1 dŏu 3 coins,
peas 1 dŏu 5 coins,
millet 1 dŏu 6 coins.

En dŏu $\approx$ 2 liter.

Detta exempel är hämtat ur kapitel 8 i den cirka 2000 år gamla boken "The Nine Chapters on the Mathematical Art (Jiuzhang Suanshu)" Boken behandlar 246 problem i 9 kapitel. Boken är den mest betydelsefulla kinesiska matematiska klassikern.

$$\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 2 & 5 & 3 & 9 & 4 \\ 3 & 5 & 7 & 6 & 4 \\ 7 & 6 & 4 & 5 & 3 \\ 9 & 7 & 3 & 2 & 5 \end{bmatrix} \begin{bmatrix} h \\ w \\ b \\ p \\ m \end{bmatrix} = \begin{bmatrix} 95 \\ 112 \\ 116 \\ 128 \\ 140 \end{bmatrix}$$

I Matlab:

```
>> x = A \ b
x =
    7.0000
    4.0000
    3.0000
    5.0000
    6.0000
```

Kineserna utnyttjar det som vi kallar Gausselimination ($\approx$ 1800). Kombinera rader så att vi till slut har en triangulär matris.

$$\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 2 & 5 & 3 & 9 & 4 \\ 3 & 5 & 7 & 6 & 4 \\ 7 & 6 & 4 & 5 & 3 \\ 9 & 7 & 3 & 2 & 5 \end{bmatrix} x = \begin{bmatrix} 95 \\ 112 \\ 116 \\ 128 \\ 140 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & -4 & 1 & -18 & -11 \\ 0 & -15 & -10 & -51 & -32 \\ 0 & -20 & -15 & -70 & -40 \end{bmatrix} x = \begin{bmatrix} 95 \\ -78 \\ -169 \\ -537 \\ -715 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 5 & 54 & 58 \\ 0 & 0 & 5 & 70 & 80 \end{bmatrix} x = \begin{bmatrix} 95 \\ -78 \\ 143 \\ 633 \\ 845 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 60 & 67 \end{bmatrix} x = \begin{bmatrix} 95 \\ -78 \\ 143 \\ 490 \\ 702 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 0 & \frac{62}{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 95 \\ -78 \\ 143 \\ 490 \\ 372/11 \end{bmatrix}$$

Vi löser det triangulära problemet med så kallad bakåtsubstitution:

$$\begin{aligned} x_5 &= (372/11)/(62/11) = 6 \\ x_4 &= (490 - 45x_5)/44 = 5 \\ x_3 &= (143 - 10x_4 - 13x_5)/5 = 3 \\ x_2 &= (-78 - (-1)x_3 - (-7)x_4 - (-6)x_5)/(-1) = 4 \\ x_1 &= (95 - 3x_2 - 2x_3 - 8x_4 - 5x_5)/1 = 7 \end{aligned}$$

Kan formulera eliminationen som en serie matrismultiplikationer.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 \\ -3 & 0 & 1 & 0 & 0 \\ -7 & 0 & 0 & 1 & 0 \\ -9 & 0 & 0 & 0 & 1 \end{bmatrix}}_{L_1} \underbrace{\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 2 & 5 & 3 & 9 & 4 \\ 3 & 5 & 7 & 6 & 4 \\ 7 & 6 & 4 & 5 & 3 \\ 9 & 7 & 3 & 2 & 5 \end{bmatrix}}_{A} = \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & -4 & 1 & -18 & -11 \\ 0 & -15 & -10 & -51 & -32 \\ 0 & -20 & -15 & -70 & -40 \end{bmatrix}$$

Elementen $-L_1(2:5,1)$ (dvs. $2,3,7,9$) kallas multiplikatorer.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -4 & 1 & 0 & 0 \\ 0 & -15 & 0 & 1 & 0 \\ 0 & -20 & 0 & 0 & 1 \end{bmatrix}}_{L_2} \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & -4 & 1 & -18 & -11 \\ 0 & -15 & -10 & -51 & -32 \\ 0 & -20 & -15 & -70 & -40 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 5 & 54 & 58 \\ 0 & 0 & 5 & 70 & 80 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}}_{L_3} \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 5 & 54 & 58 \\ 0 & 0 & 5 & 70 & 80 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 60 & 67 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{-15}{11} & 1 \end{bmatrix}}_{L_4} \begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 60 & 67 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 0 & \frac{62}{11} \end{bmatrix}}_{U}$$

Vi får

$$L_4 L_3 L_2 L_1 A = U$$

eller

$$A = (L_4 L_3 L_2 L_1)^{-1} U$$

Vi noterar (se övningarna):

- L_k är inverterbar
- L_k^{-1} ser nästan ut som L_k

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \alpha & 1 & 0 \\ 0 & \beta & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\alpha & 1 & 0 \\ 0 & -\beta & 0 & 1 \end{bmatrix}$$

- det är enkelt att multiplicera L_k-matriser:

```
L1 =
     1     0     0     0
     2     1     0     0
     3     0     1     0
     4     0     0     1
L2 =
     1     0     0     0
     0     1     0     0
     0    20     1     0
     0    30     0     1

>> L1*L2
     1     0     0     0
     2     1     0     0
     3    20     1     0
     4    30     0     1

>> L2*L1
     1     0     0     0
     2     1     0     0
    43    20     1     0
    64    30     0     1
```

Om vi sammanställer allt detta får vi med andra ord

$$\underbrace{\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 2 & 5 & 3 & 9 & 4 \\ 3 & 5 & 7 & 6 & 4 \\ 7 & 6 & 4 & 5 & 3 \\ 9 & 7 & 3 & 2 & 5 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 4 & 1 & 0 & 0 \\ 7 & 15 & 1 & 1 & 0 \\ 9 & 20 & 1 & \frac{15}{11} & 1 \end{bmatrix}}_{L_1^{-1}L_2^{-1}L_3^{-1}L_4^{-1}} \underbrace{\begin{bmatrix} 1 & 3 & 2 & 8 & 5 \\ 0 & -1 & -1 & -7 & -6 \\ 0 & 0 & 5 & 10 & 13 \\ 0 & 0 & 0 & 44 & 45 \\ 0 & 0 & 0 & 0 & \frac{62}{11} \end{bmatrix}}_{U}$$

Detta kallas LU-uppdelning. L är undertriangulär och U övertriangulär.

För att konstruera L plockar vi alltså in multiplikatorerna från de olika L_k på respektive plats i L.

Vad är det för fördel med detta jämfört med vanlig GE (Gausselimination)? Svar: enklare att hantera vid teoretiskt arbete. Gör det möjligt att lösa problem av typen $Ax_k = b_k$ där b_{k+1} beror av x_k. (Om alla högerleden är kända på en gång kan givetvis vanlig GE utnyttjas).

Så normalt löses $Ax = b$ i de tre stegen

1. beräkna L och U så att $A = LU$
   för att lösa $LUx = b$ inför vi beteckningen $z = Ux$ och får då problemet $Lz = b$
2. lös $Lz = b$ (framåtsubstitution)
3. lös $Ux = z$ (bakåtsubstitution)

Framåtsubstitution går till på samma vis som bakåtsubstitutionen fast man tar raderna i omvänd ordning.

Kostnad?
$A = LU$ tar ungefär $n^3/3$ vardera av $+$ och $*$
$Lz = b$ kostar $n^2/2$ vardera av $+$ och $*$
($Ux = z$ kostar lika mycket).

Ett numeriskt exempel

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 4 & -2 & 12 \\ 3 & -7 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

Vi startar med en "tom" $L = I$ och fyller i multiplikatorerna eftersom under diagonalen.

$$A = \begin{bmatrix} \boxed{1} & -1 & 2 \\ \boxed{4} & -2 & 12 \\ \boxed{3} & -7 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 & 2 \\ 0 & \boxed{2} & 4 \\ 0 & \boxed{-4} & -5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix} = U$$

Så, det som skall stå i L är symboliskt $\begin{smallmatrix} \square & \\ \square & \end{smallmatrix}$ och U är det som blir kvar av A efter trianguleringen.

Alltså:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \boxed{4} & 1 & 0 \\ \boxed{3} & \boxed{-2} & 1 \end{bmatrix} \quad \text{och} \quad U = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix}$$

Lös $Lz = b$, ger:

$$z = \begin{bmatrix} 3 \\ -6 \\ -12 \end{bmatrix}$$

Lös $Ux = z$, ger:

$$x = \begin{bmatrix} 16 \\ 5 \\ -4 \end{bmatrix}$$

Kontroll:

$$Ax = \begin{bmatrix} 1 & -1 & 2 \\ 4 & -2 & 12 \\ 3 & -7 & 1 \end{bmatrix} \begin{bmatrix} 16 \\ 5 \\ -4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix} = b, \quad \text{OK!}$$

Är detta en stabil algoritm?

Här följer en grov skiss som visar vad som kan gå fel.

Låt ϵ stå för ett litet tal. a_1, a_2 och a_3 markerar "medelstora" tal. LU-faktorisering blir då:

$$\underbrace{\begin{bmatrix} \epsilon & a_2 \\ a_1 & a_3 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ a_1/\epsilon & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \epsilon & a_2 \\ 0 & a_3 - a_2(a_1/\epsilon) \end{bmatrix}}_{U}$$

a_1/ϵ blir ett stort tal, vilket ger utskiftning i beräkningen av $u_{2,2} = a_3 - a_2(a_1/\epsilon)$. Låt oss anta att hela a_3 skiftas ut och att allt annat räknas ut exakt. Hur stort blir bakåtfelet?

$$\underbrace{\begin{bmatrix} 1 & 0 \\ a_1/\epsilon & 1 \end{bmatrix}}_{\text{ber. L}} \underbrace{\begin{bmatrix} \epsilon & a_2 \\ 0 & -a_2(a_1/\epsilon) \end{bmatrix}}_{\text{ber. U}} = \underbrace{\begin{bmatrix} \epsilon & a_2 \\ a_1 & 0 \end{bmatrix}}_{\text{faktoriserad matris}}$$

Vi har alltså faktoriserat en matris som avviker mycket från A i (2,2)-elementet. Algoritmen behöver inte vara stabil.

Det kan vi dock lätt fixa. Kasta om raderna i systemet (byt ordning på ekvationerna), dvs. studera matrisen $B = PA$:

$$B = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{P} A = \begin{bmatrix} a_1 & a_3 \\ \epsilon & a_2 \end{bmatrix}$$

LU-faktorisering blir nu:

$$\begin{bmatrix} a_1 & a_3 \\ \epsilon & a_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \epsilon/a_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_3 \\ 0 & a_2 - a_3(\epsilon/a_1) \end{bmatrix}$$

Notera att ϵ/a_1 är ett litet tal. Vi får alltså inte farlig utskiftning i $u_{2,2}$.

Låt oss anta att $a_3(\epsilon/a_1)$ skiftas ut:

$$\underbrace{\begin{bmatrix} 1 & 0 \\ \epsilon/a_1 & 1 \end{bmatrix}}_{\text{ber. L}} \underbrace{\begin{bmatrix} a_1 & a_3 \\ 0 & a_2 \end{bmatrix}}_{\text{ber. U}} = \underbrace{\begin{bmatrix} a_1 & a_3 \\ \epsilon & a_2 + a_3\epsilon/a_1 \end{bmatrix}}_{\text{faktoriserad matris}} = B + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & a_3\epsilon/a_1 \end{bmatrix}}_{\text{Fel}}$$

Detta förfarande kallas partiell pivotering och det får utföras i varje eliminationssteg. Här följer ett exempel:

$$A = \begin{bmatrix} -0.01 & 0.80 & 3.8 \\ -0.10 & -2.05 & 7.8 \\ 1.00 & 20.00 & 20.0 \end{bmatrix}, \quad P_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0.1 & 1 & 0 \\ 0.01 & 0 & 1 \end{bmatrix}}_{L_1} \underbrace{\begin{bmatrix} 1 & 20.00 & 20.0 \\ -0.1 & -2.05 & 7.8 \\ -0.01 & 0.80 & 3.8 \end{bmatrix}}_{P_1A} = \underbrace{\begin{bmatrix} 1 & 20.00 & 20.0 \\ 0 & -0.05 & 9.8 \\ 0 & 1.00 & 4.0 \end{bmatrix}}_{L_1P_1A}$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.05 & 1 \end{bmatrix}}_{L_2} \underbrace{\begin{bmatrix} 1 & 20.00 & 20.0 \\ 0 & 1.00 & 4.0 \\ 0 & -0.05 & 9.8 \end{bmatrix}}_{P_2L_1P_1A} = \underbrace{\begin{bmatrix} 1 & 20 & 20 \\ 0 & 1 & 4 \\ 0 & 0 & 10 \end{bmatrix}}_{U}$$

Så $L_2P_2L_1P_1A = U$. Kan visa att vi beräknat $P_2P_1A = \tilde{L}U$.

$$\underbrace{\begin{bmatrix} 1.00 & 20.00 & 20.0 \\ -0.01 & 0.80 & 3.8 \\ -0.10 & -2.05 & 7.8 \end{bmatrix}}_{P_2P_1A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -0.01 & 1 & 0 \\ -0.1 & -0.05 & 1 \end{bmatrix}}_{\tilde{L}=[P_2L_1^{-1}P_2]L_2^{-1}} \underbrace{\begin{bmatrix} 1 & 20 & 20 \\ 0 & 1 & 4 \\ 0 & 0 & 10 \end{bmatrix}}_{U}$$

Vi bildar givetvis aldrig permutationsmatriserna utan rader flyttas via tilldelning eller pekare.

LDU-faktoriseringen

L har ettor på diagonalen. Kan få ettor på U:s diagonal genom att "bryta ut" U:s diagonal (antar A ickesingulär).

$$D = \text{diag}(u_{1,1}, \ldots, u_{n,n}), \quad A = LU = LD(D^{-1}U)$$

Sätt $\hat{U} = D^{-1}U$ så blir $A = LD\hat{U}$ där både L och $\hat{U}$ har ettor på diagonalen.

Vi struntar i pivotering för att slippa bråk.

$$\begin{bmatrix} 2 & 6 \\ 4 & 15 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}$$

Vi kan utnyttja detta för att titta på två viktiga fall:

A symmetrisk: $A = A^T \Rightarrow \hat{U} = L^T$ så att $A = LDL^T$.

Innebär halva antalet operationer för faktoriseringen (förutsatt att vi utnyttjar symmetrin i vår algoritm).

Kräver halva lagringsutrymmet.

$$\begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 0 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

Problem med pivotering och symmetri ty partiell pivotering förstör symmetrin (finns andra pivoteringsalgoritmer).

Det andra viktiga fallet inträffar när D i $A = LDL^T$ har positiva diagonalelement.

Först ett exempel:

$$\underbrace{\begin{bmatrix} 4 & 8 \\ 8 & 25 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 4 & 8 \\ 0 & 9 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}}_{L^T} =$$

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{1/2}} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{1/2}} \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}}_{L^T} = \underbrace{\left[\underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{1/2}}\right]}_{C} \underbrace{\left[\underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{1/2}}\right]^T}_{C^T}$$

Så

$$\underbrace{\begin{bmatrix} 4 & 8 \\ 8 & 25 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 2 & 0 \\ 4 & 3 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} 2 & 4 \\ 0 & 3 \end{bmatrix}}_{C^T}$$

Detta kallas Choleskyfaktorisering och den existerar när A är symmetrisk och positivt definit: $x \neq 0 \Rightarrow x^T A x > 0$.

D har i detta fall positiva diagonalelement. Man kan visa att LU-faktorisering för en positivt definit matris är stabil även om vi inte pivoterar.

Positivt definita matriser är vanliga i tillämpningar.

Exempel: vi har partiklar med massorna m_1, m_2, m_3 och farterna v_1, v_2, v_3. Den totala kinetiska energin, E_{kin} är

$$\frac{m_1 v_1^2 + m_2 v_2^2 + m_3 v_3^2}{2} = \frac{1}{2} \underbrace{[v_1, v_2, v_3]}_{v^T} \underbrace{\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{v} = \frac{v^T M v}{2}$$

$E_{kin} > 0$ om någon massa rör sig, dvs. $v \neq 0 \Rightarrow \frac{v^T M v}{2} > 0$ så att M är positivt definit.

Konditionstalet för $Ax = b$-problemet
"Proof by example"

Låt oss se hur lösningen x ändrar sig när vi stör högerledet b. Vi studerar ett numeriskt exempel där A är diagonal.

$$\begin{bmatrix} 2 & 0 \\ 0 & 10^{-10} \end{bmatrix} x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \Rightarrow x = \begin{bmatrix} 1 \\ 10^{10} \end{bmatrix}$$

Vi stör nu b med f och får då lösningen y, dvs. $Ay = b + f$. Hur mycket ändras x, dvs. hur stor är $y - x$?

$$y = A^{-1}(b + f) = A^{-1}b + A^{-1}f = x + A^{-1}f$$

så att

$$y - x = A^{-1}f = \begin{bmatrix} 1/2 & 0 \\ 0 & 10^{10} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 0.5\ f_1 \\ 10^{10}\ f_2 \end{bmatrix}$$

Det är är inte alltid så här illa. Om vi i stället tar koefficientmatrisen

$$B = \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix}$$

så blir

$$y - x = B^{-1}f = \begin{bmatrix} 0.5\ f_1 \\ 10\ f_2 \end{bmatrix}$$

Slutsats: om A har ett eller flera diagonalelement nära noll, så kommer x att vara känslig för ändringar i högerledet.

A är "nästan singulär" i följande mening:

$$\underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 10^{-10} \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & -10^{-10} \end{bmatrix}}_{E} = \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}}_{\text{singulär}}$$

Den lilla störningen E (små element jämfört med det största elementet i A) gör A singulär. A ligger alltså nära en singulär matris.

B är inte nästan singulär eftersom E måste innehålla ett stort element, -0.1.

Allmänt gäller att x är känslig för störningar i b och A om A är nästan singulär. Om A är långt från att vara singulär, så är x relativt okänslig för störningar.

En nästan singulär matris har en invers där åtminstone något element är stort. Eftersom $y - x = A^{-1}f$ så kommer x att ändras mycket om A^{-1} innehåller stora element.

Om matrisen inte är diagonal får vi ett mer komplicerat uppträdande. Antag att $\delta > 0$ är nära noll.

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 1+\delta \end{bmatrix} \Rightarrow C^{-1} = \frac{1}{\delta} \begin{bmatrix} 1+\delta & -1 \\ -1 & 1 \end{bmatrix}$$

Vi ser att C^{-1} är proportionell mot $1/\delta$, så inversen är stor. C ligger också nära en singulär matris. ty om vi subtraherar δ från $c_{2,2}$ så blir matrisen singulär. Detta gäller allmänt.

Om C har element av storleksordningen ett:

storlek på $C^{-1} \approx 1$ / avståndet till närmaste singulära matris

För att göra riktiga satser krävs mer matematik, vektor- och matrisnormer.

Vektornormer

En vektornorm är en funktion som ger ett mått på storleken på elementen i en vektor. Om vektorn innehåller n element så sammanfattar vi storleken med ett ickenegativt tal, så normer kan vara trubbiga mätverktyg.

Det finns oändligt många normer. Vi kommer att använda tre så kallade L_p-normer som vi betecknar med $||\cdot||_p$:

$$||x||p = \left[\sum_{k=1}^{n} |x_k|^p\right]^{\frac{1}{p}}, \quad p > 0$$

- $p = 1$, $||x||_1 = \sum_{k=1}^{n} |x_k|$, ettnormen
- $p = 2$, $||x||_2 = \left[\sum_{k=1}^{n} x_k^2\right]^{\frac{1}{2}}$, tvånormen
- $p = \infty$, $||x||_\infty = \max_{1 \le k \le n} |x_k|$, maxnormen

Dessa tre normer (liksom alla vektornormer) uppfyller:

- $x \neq 0 \Rightarrow ||x|| > 0$ (positivitet), $||0|| = 0$
- $||\alpha x|| = |\alpha|\ ||x||$ för alla $\alpha \in \Re$ (homogenitet)
- $||x + y|| \le ||x|| + ||y||$ (triangelolikheten)

Normer är olika stora

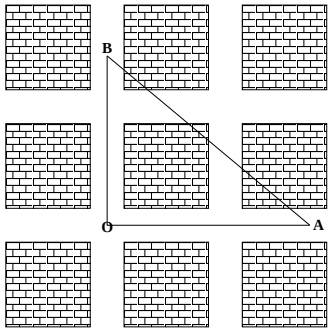
$$x = \begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix}, \quad ||x||_1 = 6, \quad ||x||_2 = \sqrt{14}, \quad ||x||_\infty = 3$$

Varför räcker det inte med den "vanliga längden" av en vektor, dvs. det vi kallar tvånormen?

Det beror på att olika problemställningar kräver olika sätt att mäta storlek. Dessutom kan det vara så att det går att skapa starkare satser för en viss norm.

Exempel:
Antag att vi befinner oss i en stad där kvarteren ligger i ett rutnät. Vi vill ta oss den kortaste vägen från A till B. Låt O beteckna origo. Om vi kan flyga så är avståndet $||B - A||_2$. Om vi måste följa gatorna så är avståndet $||B - A||_1$ (dvs. längden av $\overline{OA}$ plus längden av $\overline{OB}$).

$||B - A||_\infty$ kan tolkas som den största förflyttning som vi gör sidledes.



Innerprodukter

Vi kan definiera en norm givet en innerprodukt (skalärprodukt).

$$x \cdot y = (x, y) = \sum_{k=1}^{n} x_k y_k = x^T y$$

Så

$$||x||_2 = \sqrt{x^T x}$$

Notera att $x^T y$ är en skalär men $x y^T$ är en matris.

Exempel:

$$\begin{bmatrix} -1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = 4, \quad \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -2 & -1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Om $x \neq 0$ så säger vi att vektorn $x/||x||$ är normerad (har längd ett).

Definition av vinkel

$$x^T y = ||x||_2 ||y||_2 \cos\phi$$

Cauchy-Schwarz olikhet

$$|x^T y| \le ||x||_2 ||y||_2$$

Matrisnormer

Matrisnormer är funktioner från $\Re^{m \times n}$ till $\Re$ och uppfyller de tre vektornormsvillkoren ovan.

Användbara matrisnormer är dessutom submultiplikativa (konsistenta):

$$||AB|| \le ||A||\ ||B||$$

Tre olika normer kan vara inblandade ovan. Vi använder samma beteckning $||\cdot||$ för alla tre.

Vi kan bilda matrisnormer utgående från vektornormer.
En operatornorm norm mäter hur mycket multiplikation med en matris A kan förstora en vektor:

$$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

Vi noterar att $||I|| = 1$ om $||\cdot||$ är en operatornorm.

Operatornormerna som svarar mot våra tidigare vektornormer:

- $p = 1$, $||A||_1 = \max_k \sum_{r=1}^{m} |a_{r,k}|$, ettnormen, största kolonnsumman
- $p = 2$, $||A||_2 = \max \left[\lambda(A^T A)\right]^{\frac{1}{2}}$ tvånormen
- $p = \infty$, $||A||_\infty = \max_r \sum_{k=1}^{n} |a_{r,k}|$, maxnormen, största radsumman

Exempel:

$$A = \begin{bmatrix} 1 & -2 & -3 \\ 6 & 4 & 2 \\ 9 & -6 & 3 \end{bmatrix}, \quad ||A||_1 = 16, \quad ||A||_2 \approx 11.9042, \quad ||A||_\infty = 18$$

Konditionstal för $Ax = b$-problemet

Vi skall nu använda normer för att studera konditionstalet för $Ax = b$-problemet. Vi vill alltså studera vad som händer med x när vi ändrar A och b. Vi kommer endast att ändra b.

Sats: Antag att A är ickesingulär och att $Ax = b \neq 0$.
Om $Ay = b + f$ så gäller att

$$\frac{||x-y||}{||x||} \leq ||A||\ ||A^{-1}||\ \frac{||f||}{||b||}$$

Bevis:

$$Ay = b + f \ \text{ och } \ Ax = b \Rightarrow A(y-x) = f \Rightarrow$$
$$y - x = A^{-1}f \Rightarrow ||y-x|| = ||A^{-1}f|| \leq ||A^{-1}||\ ||f||$$

Men $Ax = b$ så att $||A||\ ||x|| \geq ||b||$ eller $1/||x|| \leq ||A||/||b||$. ■

Man kan bevisa likartade satser för fallen när A eller A och b störs. Normalt betecknas konditionstalet med kappa, dvs. $\kappa(A) = ||A||\ ||A^{-1}||$.

Antag att $||\cdot||$ är en operatornorm, då gäller:

- $\kappa(A) \geq 1$ för alla A, ty $1 = ||AA^{-1}|| \leq ||A||\ ||A^{-1}||$
- I är perfekt konditionerad ty $\kappa(I) = 1$
- konditionstalet är skalningsoberoende $\kappa(\alpha A) = \kappa(A)$
- $\kappa(A) = \infty$ om A är singulär

Om A är singulär kan det finnas ingen eller oändligt många lösningar. Vi förväntar oss problem om A är <u>nästan</u> singulär.

Om $\kappa(A)$ är stort så finns en matris, E, med liten norm $||E||$, så att $A + E$ är exakt singulär. A "ligger nära" mängden av singulära matriser, matrisen är "nästan" singulär.

Om $\kappa(A)$ är litet måste man ändra A mycket (stor E) för att $A + E$ skall bli singulär.

Man kan visa att de E som gör $A + E$ singulär och som har minsta norm uppfyller $||E|| = ||A||/\kappa(A)$.

Determinanten för A inte är något bra mått på nästan singulär.

Exempel:

$$\det(\alpha I) = \alpha^n, \quad \kappa(\alpha I) = 1$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}, \quad \det(A) = 0.1, \quad \kappa(A) = 10$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad \det(A) = 0.001, \quad \kappa(A) = 10$$

Exempel: Hur väl stämmer satsen?

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-8} \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \kappa_\infty(A) = 10^8$$

Tag

$$f = \begin{bmatrix} 10^{-9} \\ 10^{-9} \end{bmatrix} \Rightarrow y - x = \begin{bmatrix} 10^{-9} \\ 10^{-1} \end{bmatrix}, \quad \frac{||x-y||_\infty}{||x||_\infty} = \frac{0.1}{1} = 0.1$$

$$\kappa_\infty(A)\frac{||f||_\infty}{||b||_\infty} = 10^8\frac{10^{-9}}{1} = 0.1$$

Så likhet i gränsen. Tag nu i stället

$$f = \begin{bmatrix} 10^{-9} \\ 0 \end{bmatrix} \Rightarrow y - x = \begin{bmatrix} 10^{-9} \\ 0 \end{bmatrix}, \quad \frac{||x-y||_\infty}{||x||_\infty} = \frac{10^{-9}}{1} = 10^{-9}$$

$$\kappa_\infty(A)\frac{||f||_\infty}{||b||_\infty} = 10^8\frac{10^{-9}}{1} = 0.1$$

vilket ger en enorm överskattning.

Tolkning av satsen

Vi kan anta att x är normerad, ty:

$$\frac{||x-y||}{||x||} = \left|\left| \frac{x}{||x||} - \frac{y}{||x||} \right|\right|, \quad \left|\left| \frac{x}{||x||} \right|\right| = 1$$

Antag att elementen i x, y är av samma storleksordning, $x \approx \xi e$, $y \approx \eta e$, där e består av ettor.

$$\frac{||x-y||}{||x||} \approx \frac{||\xi e - \eta e||}{||\xi e||} = \frac{|\xi - \eta|}{|\xi|} = \left|1 - \frac{\eta}{\xi}\right|$$

Vi kan låtsas som om det exakta värdet är ett, vilket underlättar jämförelsen. Så om $\xi = 1$ och $\eta = 1 \pm 0.5 \cdot 10^{-3}$ så blir $|1 - \eta/\xi| \leq 0.5 \cdot 10^{-3}$.

Vi kan använda ungefär samma tolkning för $||f||/||b||$ (om vi antar att elementen har samma storleksordning):

$$\frac{||b-(b+f)||}{||b||} = \frac{||f||}{||b||} \approx \frac{||\phi\ e||}{||\beta\ e||} = \frac{|\phi|}{|\beta|}$$

Antag att $\kappa(A)\ ||f||/||b|| = 0.5 \cdot 10^{-3}$. Det approximativa värdet ligger då i intervallet $1 \pm 0.5 \cdot 10^{-3}$, så att vi har tre decimaler (och fyra signifikanta siffror totalt).

Vi får följande tumregel:

Om $\kappa(A) = 10^p$ så riskerar vi att tappa p siffror.

Antag nu att x innehåller element av olika storleksordning, t.ex. $x = [1, 10^{-3}]^T$ och att vi använder $||\cdot||_\infty$. Det gäller då att:

$$\max\{|1 - y_1|, |10^{-3} - y_2|\} \leq 0.5 \cdot 10^{-3}$$

så att

$$1 - 0.5 \cdot 10^{-3} \leq y_1 \leq 1 + 0.5 \cdot 10^{-3}$$

och

$$10^{-3} - 0.5 \cdot 10^{-3} \leq y_2 \leq 10^{-3} + 0.5 \cdot 10^{-3}$$

Normer kan vara trubbiga instrument.

Låt oss se på två fall.

<u>Exakta indata:</u> i detta fall får vi eventuellt avrundningsfel när $a_{j,k}$ och b_k lagras i datorns minne. Relativa felet (per komponent) är ungefär ϵ_{mach}. Vi får också avrundningsfel när vi löser $Ax = b$-problemet.

Vi kan nog tillåta tämligen stora $\kappa(A)$, men det beror givetvis på hur många siffror vi behöver. Om vi har stora krav eller om $\kappa(A)$ är mycket stort, så kan vi minska ϵ_{mach} genom att t.ex. använda Maple eller Mathematica (räkna med fler siffror). Att räkna med många siffror går dock mycket långsammare (mjukvara och inte hårdvara).

Att lösa ett osymmetriskt problem med $n = 100$ tar i Maple på en 900 MHz Sun.

`Digits`	tid(s)
20	12.8
40	26.0
60	34.1
80	31.4
100	34.3

Matlab tar 4.2 ms. Maple tar alltså flera tusen gånger mera tid.

Hur stort problem klarar Matlab på 10 sekunder? Jo, $n \approx 2150$.

Indata med osäkerhet (mätdata): detta fall ger normalt större begränsningar på hur stora $\kappa(A)$ som kan tillåtas.

Felet i indata varierar beroende på problem. Atomklockor kan uppvisa mycket små fel, kanske 10^{-15} s.
`http://www.boulder.nist.gov/timefreq/general/precision.htm` .

Ett elektriskt motstånd kan ha fel på 5% (finns bättre).

Att räkna med mindre ϵ_{mach} ger normalt inte en bättre lösning eftersom $\kappa(A)$ är måttligt stort och mätfelen helt dominerar över avrundningsfelen.

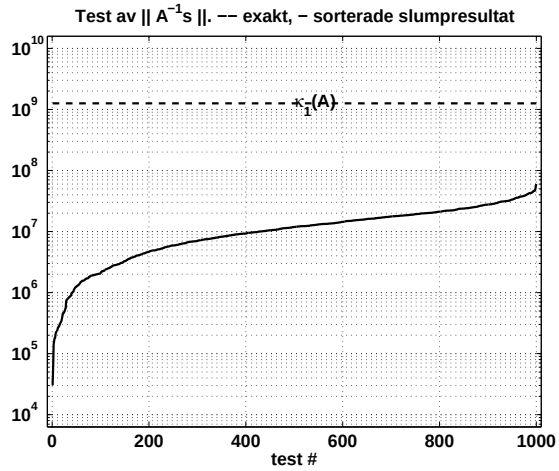
Hur kan vi uppskatta $\kappa(A)$?

Att beräkna A^{-1} tar mycket tid och minne om A är stor.
`cond` i Matlab använder `svd` för $||\cdot||_2$ och explicit `inv` för andra normer.
För stora matriser kan man använda `condest` som uppskattar $||A^{-1}||$ genom att lösa linjära ekvationssystem (+ listigheter).

Även LAPACK kan ge en sådan uppskattning när man löser $Ax = b$. Uppskattningen kostar nästan inget eftersom man utnyttjar den LU-faktorisering som redan beräknats.

Antag nu att man inte har tillgång till något av ovanstående. Fungerar följande? Tag en normerad slumpvektor, s, $||s|| = 1$, och lös $Az = s$ (dvs. beräkna $z = A^{-1}s$). Då gäller:

$$||z|| = ||A^{-1}s|| \leq ||A^{-1}||\ ||s|| = ||A^{-1}|| \Rightarrow \kappa(A) \geq ||A||\ ||z||$$



Varför stämmer det så dåligt? I exemplet är $n = 800$ och matrisen är en osymmetrisk slumpmatris.
Det stämmer bättre för mindre n.

Om $A = \text{diag}(1, \ldots, 1, 10^{-8})$ så måste slumpvektorn ha $s_n \neq 0$ som är tillräckligt stort, för att vi ska se att A är illa konditionerad.

Om s har ungefär lika stora komponenter i varje riktning, och $||s||_1 = 1$ torde $s_n \approx 1/n$ så att uppskattningen kanske blir $||A||_1 10^8/n$. Så uppskattningen borde bli sämre när n ökar.

Det existerar vektorer så att man märker det stora $\kappa(A)$, men dessa vektorer (singulära vektorer) är lika besvärliga att beräkna som egenvektorer. `condest`, som försöker uppskatta en singulär vektor, fungerar mycket bra.

Att bilda $z = A^{-1}s$ kan ses som första steget i "invers iteration", en metod för att beräkna en egenvektor till egenvärdet närmast noll.

Egenvärden är inte rätt verktyg för att mäta konditionstal.
Rätt verktyg är singulära värden. Låt A_n, av dimension n, vara den övertriangulära matrisen med ettor på diagonalen och -1 ovanför diagonalen. Egenvärdena är alla ett, men $\kappa_1(A) = n2^{n-1}$. Vi noterar att $\det(A_n) = 1$,

Ovanstående (att bilda $z = A^{-1}s$) kan ses som experimentell störningssanalys. Vi kan lösa $Ay = b + s$ där s är en slumpvektor med element av samma storleksordning som mätfelen. Eftersom $A^{-1}(b + s) - A^{-1}b = A^{-1}s$ ser man att risken, även här, är att man inte hittar störningen, s, som ger den största störningen.

Vad säger residualen $r = b - A\hat{x}$? $\hat{x}$ beräknad lösning.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-8} \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{x} = \begin{bmatrix} 1 \\ 10^4 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ -10^{-4} \end{bmatrix}$$

Kan visa:

$$(A + E)\hat{x} = b, \quad ||E||_2 = ||r||_2/||\hat{x}||_2$$

$||E||_2 \approx 10^{-8}$ i exemplet.

Så en liten residual betyder att vi löst nästan rätt problem.

I framåtriktningen kommer $\kappa(A)$ in (antag att A^{-1} existerar):

$$r = b - A\hat{x} = Ax - A\hat{x} = A(x - \hat{x}) \Leftrightarrow x - \hat{x} = A^{-1}r$$

Alltså gäller:

$$||x - \hat{x}|| \leq ||A^{-1}||\ ||r||$$

Om $b \neq 0$ gäller:

$$||b|| \leq ||A||\ ||x|| \Leftrightarrow \frac{1}{||x||} \leq \frac{||A||}{||b||}$$

Om vi kombinerar de två olikheterna erhåller vi:

$$\frac{||x - \hat{x}||}{||x||} \leq \kappa(A)\ \frac{||r||}{||b||}$$

Så felet i lösningen kan vara godtyckligt stort även om residualen är liten.

Minstakvadratproblem

Ett ofta återkommande problem är att vi har en matematisk modell och uppmätta värden, och vill bestämma parametrar i modellen.

En mycket vanlig modell är $b = ce^{\lambda t}$ (halveringstid, befolkningstillväxt, urladdning av kondensator). b kan vara befolkningen vid en given tidpunkt t. c är befolkningsmängden vid tiden $t = 0$.

Antag att vi vill bestämma parametern λ genom att mäta b vid m olika tidpunkter. Vi alltså har m par $(t_k, b_k), k = 1, \ldots, m$ av mätvärden. Låt oss anta att vi känner c.

Hur ska vi beräkna λ? Vi har ju m olika ekvationer

$$b_1 = ce^{\lambda t_1}, \quad b_2 = ce^{\lambda t_2}, \ldots, \quad b_m = ce^{\lambda t_m}$$

och vi lär inte kunna hitta ett λ som satisfierar alla ekvationerna. Det är inte intressant att få m olika λ-värden.

En rimlig kompromiss är att hitta ett λ som approximativt satisfierar alla ekvationerna, dvs:

$$b_1 \approx ce^{\lambda t_1}, \quad b_2 \approx ce^{\lambda t_2}, \ldots, \quad b_m \approx ce^{\lambda t_m}$$

Detta kan formuleras på följande vis:

Försök att göra alla avvikelserna (residualerna)

$$ce^{\lambda t_1} - b_1, \quad ce^{\lambda t_2} - b_2, \ldots, \quad ce^{\lambda t_m} - b_m$$

så små (när noll) som möjligt.

Vi hade lika gärna kunnat studera avvikelserna $b_k - ce^{\lambda t_k}$.

Vi kan definiera "små" på oändligt många sätt, till exempel:

$$\min_{\lambda} \sum_{k=1}^{m} \left| ce^{\lambda t_k} - b_k \right|$$

$$\min_{\lambda} \left[\sum_{k=1}^{m} \left[ce^{\lambda t_k} - b_k \right]^2 \right]^{1/2} \qquad \min_{\lambda} \max_{1 \le k \le m} \left| ce^{\lambda t_k} - b_k \right|,$$

Dessa förslag är inte slumpvis valda, utan låt oss införa en vektor, r, av alla residualerna (en residualvektor):

$$r = \begin{bmatrix} ce^{\lambda t_1} - b_1 \\ ce^{\lambda t_2} - b_2 \\ \vdots \\ ce^{\lambda t_m} - b_m \end{bmatrix}$$

Vår tre mått kan då skrivas:

$$\min_{\lambda} ||r||_1, \quad \min_{\lambda} ||r||_2, \quad \min_{\lambda} ||r||_\infty$$

Vi kommer normalt att få olika värden på λ beroende på vilken norm vi utnyttjar. Varje λ är dock bäst för den givna normen.

Det finns oändligt många frågor, varje fråga med sitt svar; varje svar är dock ett korrekt svar på den givna frågan.

Det finns normalt inte ett bästa λ-värde.

Man kan givetvis ha modeller med flera parametrar. Ett vanligt problem är att anpassa en serie mätpunkter till en rät linje.

Vår modell kan då skrivas; $b = x_1 + x_2 t$. Här är x_1 och x_2 parametrar och (t_k, b_k) är uppmätta värden.

Hur ser residualvektorn ut i detta fall?

$$r = \begin{bmatrix} x_1 + x_2 t_1 - b_1 \\ x_1 + x_2 t_2 - b_2 \\ \vdots \\ x_1 + x_2 t_m - b_m \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}}_{b}$$

Dvs. $r = Ax - b$. Vi vill alltså lösa minimeringsproblemet:

$$\min_{x} \; ||Ax - b||$$

i någon lämplig norm.

Observera att detta normalt inte är ett linjärt ekvationssystem. Vi löser inte $Ax = b$, ty detta går normalt inte, eftersom vi får avvikelser i alla ekvationerna. Lite slarvigt kan man skriva $Ax \approx b$. Om vi kan lösa $Ax = b$ så är ju residualvektorn $r = Ax - b$ nollvektorn, varför mätpunkterna följer modellen exakt. Notera också att matrisen A har fler rader än kolonner.

När residualvektorn kan skrivas $r = Ax - b$ säger vi att problemet är linjärt. Modellen har utseendet:

$$b = \text{uttryck}_1 \; \text{parameter}_1 + \ldots + \text{uttryck}_n \; \text{parameter}_n$$

där uttryck_k beror av mätvärdena och inte beror av någon parameter.

Vår första modell är ickelinjär eftersom parametern λ inte ingår linjärt i modellen.

I vissa fall kan vi via substitutioner eller andra transformationer skapa en linjär modell utifrån en ickelinjär sådan. Vår första modell är enkel att transformera, förutsatt att b och c har samma tecken. Låt oss anta att både b och c är positiva:

$$b = ce^{\lambda t} \Leftrightarrow \log b = \log c + \lambda t$$

λ ingår nu linjärt i modellen.

Om vi nu antar att c inte är känd (vi mätte aldrig b för $t = 0$) så är c en parameter som nu ingår ickelinjärt i modellen. Om vi sätter $x_1 = \log c$ har vi dock en linjär modell som är identisk med modellen för vår räta linje, $\log b = x_1 + \lambda t$.

För att göra analogin ännu tydligare sätter vi $x_2 = \lambda$ och får:

$$\min_{x} \left|\left| \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \log b_1 \\ \log b_2 \\ \vdots \\ \log b_m \end{bmatrix} \right|\right|$$

Efter att x är beräknad sätter vi så $c = e^{x_1}$ och $\lambda = x_2$.

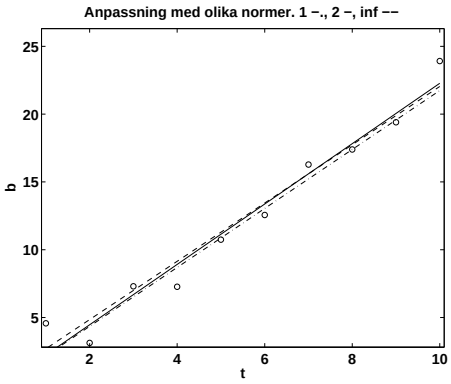
När vi gör transformationer på detta sätt ändrar vi (ibland) på normen. Logaritmering, till exempel, har en utjämnande verkan, och minskar de stora residualernas inflytande.
Detta kan jämföras med att minimera i en annan norm.
Vi ställer en annan fråga, men den kan ju vara lika relevant.

Ibland fäster vi olika stor vikt vid de olika residualerna.
Mätapparaturen kanske mäter olika noga i olika mätområden.
Det är då rimligt att ett osäkert värde får mindre inflytande än ett säkert. Vi kan åstadkomma detta med en viktad norm, t.ex.

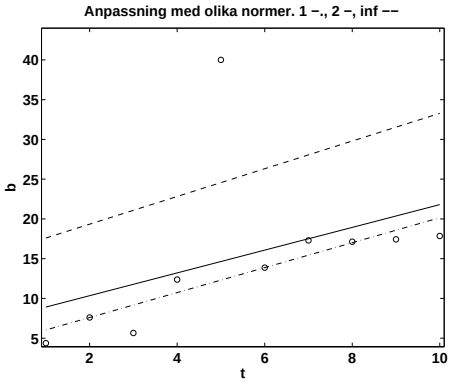
$$\min_{x} \; ||V(Ax - b)||, \quad V = \text{diag}(v_1, v_2, \ldots, v_m)$$

Residual, r_k, multipliceras alltså med vikten v_k.

Låt oss nu återvända till den räta linjen, $b = x_1 + x_2 t$.
Följande bild visar anpassning i våra tre normer.



Accentueras när man har utliggare (outliers).



Vi studerar nu det linjära minstakvadratproblemet:

$$\min_x ||Ax - b||_2$$

Det är enkelt att beskriva den optimala lösningen till detta problem. Vi ser på specialfallet när A har två kolonner, a_1 respektive a_2, men det kan enkelt generaliseras till ett godtyckligt fall.

För en godtycklig $x \in \Re^2$ gäller att $Ax = a_1x_1 + a_2x_2$ är en linjärkombination av As kolonner. När x varierar över alla vektorer med två element så kommer mängden $a_1x_1 + a_2x_2$ att bilda ett plan, As bildrum, $\mathcal{R}(A)$ (ty $\mathcal{R}(A) = \{Ax \mid x \in \Re^n\}$).

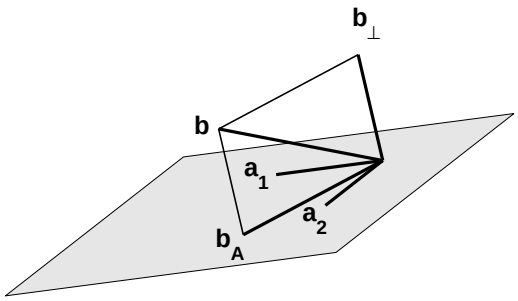
Om b tillhör detta plan så existerar (minst) ett x så att $Ax = b$ med likhet. Residualvektorn $r = Ax - b$ blir då noll. T.ex.

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

I exemplet är $x = [1,\ 1]^T$. Normalt bildar dock b en vinkel mot planet, tag till exempel $b = [2, 1, 2]^T$. Vektorn b kan då inte skrivas som en linjärkombination av As kolonner, men vi vill minimera avvikelsen, längden av residualvekorn $Ax - b$.

Dela upp b i två komponenter, b_A som ligger i planet och $b_\perp$ som är ortogonal mot planet. Oavsett hur vi väljer x så kan vi inte nollställa någon del av $b_\perp$, eftersom $b_\perp$ är ortogonal mot alla linjärkombinationer, Ax. Däremot kan vi nollställa b_A, eftersom b_A ligger i planet och den därmed är en linjärkombination av As kolonner, dvs. det existerar (minst) ett x så att $b_A = Ax$. Detta x är det x vi söker.

Residualvektorn blir $r = Ax - b = Ax - b_A - b_\perp = -b_\perp$.



Här följer samma resonemang med normer:

Pythagoras sats: om y och z är ortogonala vektorer gäller:

$$||y + z||_2^2 = ||y||_2^2 + ||z||_2^2$$

ty

$$||y+z||_2^2 = (y+z)^T(y+z) = y^Ty + \underbrace{y^Tz}_{0} + \underbrace{z^Ty}_{0} + z^Tz = ||y||_2^2 + ||z||_2^2$$

Det x som löser $Ax = b_A$ är optimalt. Ty om så inte vore fallet existerar $z \neq 0$ så att $x + z$ ger ett mindre värde på normen. Vi testar:

$$||A(x+z) - b||_2^2 = ||A(x+z) - (b_A + b_\perp)||_2^2 =$$

$$||\underbrace{Ax - b_A}_{0} + Az - b_\perp||_2^2 = ||Az||_2^2 + ||b_\perp||_2^2 \geq ||b_\perp||_2^2$$

Med minimum då $z = 0$ (om A har linjärt oberoende kolonner).

Residualvektorn, $r = -b_\perp$, är ju ortogonal mot bildrummet. Bildrummet utgörs av alla linjärkombinationer av a_1 och a_2 (i vårt specialfall) vilket medför att $a_1^Tr = a_2^Tr = 0$.
Vi kan skriva dessa likheter på följande form:

$$0 = \begin{bmatrix} a_1^Tr \\ a_2^Tr \end{bmatrix} = \begin{bmatrix} a_1^T \\ a_2^T \end{bmatrix} r = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T r = A^Tr = A^T(Ax - b)$$

vilket ger oss <u>normalekvationerna</u>:

$$A^TAx = A^Tb$$

rang$(A) = n \Rightarrow A^TA$ symmetrisk och positivt definit. Kan lösa normalekvationerna med hjälp av Choleskyfaktorisering.

Entydighet?

- om A har linjärt oberoende kolonner så har minstakvadratproblemet en entydig lösning. Matrisen har full rang.
- om A har linjärt beroende kolonner (är rangdefekt) så finns det oändligt många lösningar som ger samma residualvektor, ty tag $z \in \mathcal{N}(A)$ då gäller att $A(x + z) = Ax$.

Om A har nästan linjärt beroende kolonner, så är problemet illa konditionerat. Normalekvationerna förvärrar konditionen på problemet, ett elakt problem kan bli omöjligt att lösa. Det gäller att $\kappa(A^TA) = \kappa(A)^2$. Vi ska därför se på en bättre metod baserad på QR-faktorisering.

`x = A \ b` i Matlab använder QR-faktorisering.
Observera att operatorn \ är överlagrad. Om A är kvadratisk så används LU-faktorisering, annars används QR-faktorisering. Matlabkoderna för de två fallen har ingen gemensam del.

Kort om konditionstal för LS-problemet (LS = Least Squares)

Antag att x resp. y löser följande problem:

$$\min_x ||Ax-b||_2 \quad \text{resp.} \quad \min_y ||(A+F)y-(b+f)||_2$$

y är alltså lösningen till ett stört problem.

Vi vill begränsa $||y-x||_2/||x||_2$ i termer av $||F||_2/||A||_2$ och $||f||_2/||b||_2$.

Att göra detta allmänt är svårt. En första förenkling är att anta att A har full rang och att $||F||_2$ är tillräckligt liten så att $A+F$ har samma rang som A. Härledningen är nu avsevärt enklare, men ändå lite småbesvärlig, så på dessa sidor antar vi att $F=0$, precis som vi gjorde när vi analyserade $Ax=b$-problemet.

Eftersom A har full rang kan vi använda normalekvationerna och får $x=(A^TA)^{-1}A^Tb$ resp. $y=(A^TA)^{-1}A^T(b+f)$.

Lösningen till ett vanligt linjärt ekvationssystem, $Cx=b$, kan skrivas, $x=C^{-1}b$, så det verkar rimligt att betrakta $(A^TA)^{-1}A^T$ som en generaliserad invers. Detta gör man, och denna invers kallas pseudoinversen, beteckna A^+ och kan beräknas med Matlabkommandot `pinv`.

A^+ är ett matematiskt hjälpmedel och den brukar inte användas för att lösa minstakvadratproblem i praktiken. Vi ser att A^+ är en vänsterinvers, $A^+A=(A^TA)^{-1}A^TA=I$. Däremot är inte A^+ en högerinvers, så $AA^+\neq I$. Man kan definiera A^+ även om A är rangdefekt (men då gäller inte att $A^+=(A^TA)^{-1}A^T$).

Vi ser att

$$y-x=A^+(b+f)-A^+b=A^+f \Rightarrow ||y-x||_2 \leq ||A^+||_2\,||f||_2$$

Vi måste få en undre begränsning av $||x||_2$ och använder sambandet $Ax=b_A$, där b_A är den ortogonala projektionen av b på A:s bildrum. Antag vidare att $b_A\neq 0$ vilket medför att $x\neq 0$. Vi får

$$||b_A||_2=||Ax||_2 \leq ||A||_2\,||x||_2 \Rightarrow 1/||x||_2 \leq ||A||_2/||b_A||_2$$

Slutligen:

$$\frac{||y-x||_2}{||x||_2} \leq \underbrace{||A||_2\,||A^+||_2}_{\kappa_2(A)}\,\frac{||f||_2}{||b_A||_2}$$

Denna gräns liknar den för linjära ekvationssystem. En viktig skillnad är att det inte står $||f||_2/||b||_2$.
Låt oss skriva om uppskattningen:

$$\frac{||y-x||_2}{||x||_2} \leq ||A||_2\,||A^+||_2\,\frac{||b||_2}{||b_A||_2}\,\frac{||f||_2}{||b||_2}$$

Om modell och mätdata stämmer väl överens så kommer $||b||_2/||b_A||_2$ att vara nära ett (kvoten är alltid ≥ 1), men om modell och data inte passar ihop så kan kvoten bli stor. Extremfallet är att b är ortogonal mot A:s bildrum i vilket fall $b_A=0$ och kvoten är oändlig.

Skulle kvoten vara väldigt stor är det kanske inte så meningsfullt att lösa minstakvadratproblemet. Stör vi nu även A med F så tillkommer ytterligare en term i feluppskattningen och det visar sig att man även får en faktor $\kappa_2^2(A)\,||b_\perp||_2/||b_A||_2$ gånger de relativa störningarna.

När vi studerade $Ax=b$-problemet sa vi att $||A||/\kappa(A)$ är normen på den minsta störning, E, som gör $A+E$ singulär. Analogt gäller för minstakvadratproblemet att $||A||_2/\kappa_2(A)$ är tvånormen på den minsta E som gör att $A+E$ är rangdefekt ($A+E$ har linjärt beroende kolonner).

Exempel: låt $\epsilon,\mu>0$ vara små och antag att $0\leq\psi<\pi/2$. Sätt;

$$A=\begin{bmatrix}1 & 1\\ 0 & \epsilon\\ 0 & 0\end{bmatrix},\quad b=\begin{bmatrix}\cos\psi\\ 0\\ \sin\psi\end{bmatrix},\quad f=\begin{bmatrix}0\\ \mu\\ 0\end{bmatrix}$$

Det gäller att $\kappa_2(A)\approx 2/\epsilon$. Pseudoinversen blir:

$$A^+=(A^TA)^{-1}A^T=\frac{1}{\epsilon}\begin{bmatrix}\epsilon & -1 & 0\\ 0 & 1 & 0\end{bmatrix}$$

varför lösningarna x och y ges av

$$x=\begin{bmatrix}\cos\psi\\ 0\end{bmatrix},\quad y=\begin{bmatrix}\cos\psi-\mu/\epsilon\\ \mu/\epsilon\end{bmatrix}$$

och

$$\frac{||y-x||_2}{||x||_2}=\sqrt{2}\,\frac{\mu}{\epsilon\cos\psi}$$

Det gäller att

$$\frac{||f||_2}{||b||_2}=\frac{\mu}{1}=\mu$$

och att

$$\frac{||b||_2}{||b_A||_2}=\frac{1}{\cos\psi}$$

Vår uppskattning stämmer bra i detta exempel:

$$\underbrace{\frac{||y-x||_2}{||x||_2}}_{\sqrt{2}\,\mu/(\epsilon\cos\psi)} \leq \underbrace{\kappa_2(A)}_{2/\epsilon}\;\underbrace{\frac{||b||_2}{||b_A||_2}}_{1/\cos\psi}\;\underbrace{\frac{||f||_2}{||b||_2}}_{\mu}$$

Om $\psi\approx\pi/2$ så är b nästan ortogonal mot A:s bildrum (ψ är i själva verket vinkeln som b bildar mot bildrummet) och felet ökar med (den då stora) faktorn $1/\cos\psi$.

Alternativ till normalekvationerna

Först ett exempel som visar en nackdel med normalekvationerna.

$$A=\begin{bmatrix}1 & 1\\ 0 & \epsilon\\ 0 & 0\end{bmatrix},\ \text{med } \epsilon>0.\ A^TA=\begin{bmatrix}1 & 0 & 0\\ 1 & \epsilon & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ 0 & \epsilon\\ 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1\\ 1 & 1+\epsilon^2\end{bmatrix}$$

Om $0<\epsilon\leq\sqrt{\epsilon_{\text{mach}}}$ så är $fl(1+\epsilon^2)=1$ varför A^TA blir singulär och $A^TAx=A^Tb$ har inte entydig lösning. Minstakvadratproblemet, $\min_x||Ax-b||_2$ har dock entydig lösning så länge som $\epsilon\neq 0$.

Idé: vi utnyttjar att tvånormen är unitärt invariant, dvs.

$$||QAP||_2=||A||_2,\ \text{om } Q^TQ=I,\ P^TP=I$$

förutsatt att P är kvadratisk (Q behöver dock inte vara kvadratisk). Speciellt kan A vara en vektor, v säg, så:

$$||Qv||_2=||v||_2$$

En komplex matris, Q, är unitär då $Q^HQ=I$. Så unitär är motsvarigheten till ortogonal för reella matriser.

Bevis av $||Qv||_2=||v||_2$. Utnyttja att $||\cdot||_2\geq 0$ och att

$$||Qv||_2^2=(Qv)^TQv=v^TQ^TQv=v^TIv=v^Tv=||v||_2^2$$

Sats: Antag att A har linjärt oberoende kolonner. A har då en QR-faktorisering: $A=QR$ där $Q^TQ=I$ och R är övertriangulär med positiva diagonalelement.

Bevis: Beviset är konstruktivt (men ger inte en lämplig algoritm). Låt R definieras av $A^TA=R^TR$, Choleskyfaktoriseringen av A^TA. Denna existerar eftersom A har full kolonnrang, och R är övertriangulär med positiva diagonalelement. Sätt nu $Q=AR^{-1}$. Inversen existerar och Q blir ortogonal, ty: $(AR^{-1})^TAR^{-1}=R^{-T}A^TAR^{-1}=R^{-T}R^TRR^{-1}=I$.

QR-faktoriseringen ovan är av "economy size"(som det står i `help qr` i Matlab).

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}}_{R}$$

Följande bild visar den fullständiga varianten:

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{Z} \underbrace{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} R \\ 0 \end{bmatrix}}$$

eller mer kortfattat

$$A = [Q, Z] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Varför går detta? Givet en bas, Q, för ett underrum, kan man utvidga den (med Z) så att man får en bas för hela rummet.

Vi använder nu den fullständiga faktoriseringen för att lösa $\min_x ||Ax - b||_2$.

$$||Ax - b||_2 = || \, [Q, Z]^T (Ax - b)||_2 = \left|\left| \begin{bmatrix} Q^T(Ax - b) \\ Z^T(Ax - b) \end{bmatrix} \right|\right|_2 =$$

$$\left|\left| \begin{bmatrix} Rx - Q^Tb \\ 0 - Z^Tb \end{bmatrix} \right|\right|_2 = \left[\, ||Rx - Q^Tb||_2^2 + ||Z^Tb||_2^2 \, \right]^{1/2} \geq ||Z^Tb||_2$$

där minimum antas när $Rx = Q^Tb$.

Vi behöver inte Z för att lösa $Rx = Q^Tb$. Vi behöver inte ens Q utan endast Q^Tb. Q är en matris men Q^Tb är en vektor.

En av nackdelarna med normalekvationerna är ju att $\kappa(A^TA) = \kappa(A)^2$. Man kan visa att $\kappa(R) = \kappa(A)$.

Hur ska vi beräkna QR-faktoriseringen på ett bra sätt?

- Klassisk Gram-Schmidt (enkelt men inte så bra). GS.
- Modifierad Gram-Schmidt (mindre dåligt). MGS. Läs själv.
- Householderspeglingar. Bra!
- Householderspeglingar med pivotering (ännu bättre; tar jag inte upp). Standardmetoden, används i Matlab till exempel.

Klassisk Gram-Schmidt via exempel

Låt oss se på ett exempel där A har två kolonner: Gör följande ansats:

$$\underbrace{\begin{bmatrix} a_1 & a_2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} q_1 & q_2 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix}}_{R}$$

Vi ser att $a_1 = q_1 r_{1,1}$ så att $||a_1||_2 = ||q_1||_2 \, |r_{1,1}|$. Men $||q_1||_2 = 1$ och vi kan välja $r_{1,1} > 0$ så att $r_{1,1} = ||a_1||_2$ och $q_1 = a_1 / r_{1,1}$. Med andra ord: $r_{1,1}$ är längden av a_1 och q_1 är den vektor man får när a_1 normeras.

Nu till nästa kolonn. $a_2 = q_1 r_{1,2} + q_2 r_{2,2}$. Vi får:

$$q_1^T a_2 = \underbrace{q_1^T q_1}_{1} r_{1,2} + \underbrace{q_1^T q_2}_{0} r_{2,2}$$

Alltså är $r_{1,2} = q_1^T a_2$. Notera att $a_2 - q_1 r_{1,2} = q_2 r_{2,2}$ och q_1 är ortogonal mot q_2. Vi kan betrakta $r_{1,2}$ som det värde som gör $a_2 - q_1 r_{1,2}$ ortogonal mot q_1. $r_{2,2}$ är längden av $a_2 - q_1 r_{1,2}$.

Här ett numeriskt exempel:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}, \; r_{1,1} = \sqrt{2}, \quad q_1 = a_1 / r_{1,1} = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$

$$r_{1,2} = q_1^T a_2 = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1/\sqrt{2}$$

$$a_2 - q_1 r_{1,2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} 1/\sqrt{2} = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

Notera att denna vektor är ortogonal mot q_1. Slutligen:

$$r_{2,2} = 1/\sqrt{2}, \quad q_2 = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix} / (1/\sqrt{2}) = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$

QR-faktoriseringen kan alltså skrivas:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

Säg att vi vill lösa $\min_x ||Ax - b||_2$ där $b^T = [1, \; 1, \; 1]$.

$$\underbrace{\begin{bmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}}_{R} x = \underbrace{\begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix}}_{Q^T} \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{b} \quad \Rightarrow \quad x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Z, som vi inte behöver, ges av

$$Z = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \end{bmatrix}$$

Householdermatriser (speglingar)

Låt u vara en kolonnvektor och definiera:

$$H = I - 2 \frac{uu^T}{u^Tu}$$

Notera att uu^T är en ytterprodukt (matris) och u^Tu är en innerprodukt (skalär).

Övning: visa att $H = H^{-1} = H^T$ så H är symmetrisk och ortogonal och lika med sin egen invers.

H kallas spegling ty H speglar varje vektor i planet $\text{span}(u)^\perp$, ortogonala komplementet till $\text{span}(u)$ (mängden av vektorer ortogonala mot u). Övning: visa detta. Låt a vara godtycklig och sätt $a = \alpha u + v$ där $u^Tv = 0$. Studera Ha.

Påstående: låt a vara en godtycklig vektor med n element. Vi kan hitta H så att element $2, \ldots, n$ i Ha är noll.

Använder vi standardbeteckningen e_j för j-te kolonnen i enhetsmatrisens gäller alltså att $Ha = \alpha \, e_1$ för något α.

Vad är α? Jo, $||Ha||_2 = ||\alpha e_1||_2$ så att $\alpha = \pm ||a||_2$.
Hur ser H ut, dvs. hur skall u väljas?

$$\alpha e_1 = \underbrace{\left[I - 2 \frac{uu^T}{u^Tu} \right]}_{H} a = a - u \underbrace{\left[\frac{2u^Ta}{u^Tu} \right]}_{\text{skalär}}$$

så u måste vara en multipel av $a - \alpha e_1$ (observera att $uu^T/(u^Tu)$ är skalningsoberoende).

Övning: visa att $u = a - \alpha e_1$ faktiskt fungerar.
Vi tar $\alpha = -\text{sign}(a_1)||a||_2$ för att slippa kancellation.

Det kräver lite minne och få operationer för att bilda Ha. Så här kan man göra:

1. $\alpha = -\text{sign}(a_1)||a||_2$.
2. $u_1 = a_1 - \alpha$, $u_k = a_k, k = 2, \ldots, n$.
3. $\beta = 2(u^T a)/(u^T u)$ en skalär.
4. $Ha = a - \beta u$, en linjärkombination av två vektorer.

Det enda extraminne vi behöver är u, en vektor.

Punkt 3 kan alternativt skrivas: Skala om u, $u = u/||u||_2$. Notera att $H = I - 2uu^T$ med detta u. $Ha = a - (2u^T a)u$. Notera att vi aldrig bildar H.

Att applicera H på en matris A är heller inte svårt. Antag att A har tre kolonner. $HA = H[a_1,\ a_2,\ a_3] = [Ha_1,\ Ha_2,\ Ha_3]$.

Vi är nu redo att att beräkna den fullständiga QR-faktoriseringen. När vi räknade ut LU-faktoriseringen multiplicerade vi med L_k-matriser så att A överfördes till övertriangulär form, U. Så, $L_{n-1} \cdots L_2 L_1 A = U$ så att $A = (L_{n-1} \cdots L_2 L_1)^{-1} U$ som ger faktoriseringen. Nu gör vi ungefär på samma sätt med L_k ersatt av H_k. Låt oss anta att A har tre kolonner. Vi kommer att välja H_1, H_2, H_3 så att

$$H_3 H_2 H_1 A = \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad A = \underbrace{(H_3 H_2 H_1)^{-1}}_{[Q,Z]} \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Produkter av ortogonal matriser är ortogonal så att $(H_3 H_2 H_1)^{-1}$ existerar och är ortogonal.

Vi antar att A är en 5×3-matris. Vi börjar med att välja H_1 så att $H_1 a_1 = \alpha_1 e_1$, dvs. så att H_1 applicerat på $\square$-vektorn blir en multipel av e_1. $\times$, $\square$ etc. markerar godtyckliga element (som ej behöver vara noll).

$$\underbrace{H_1 \begin{bmatrix} \square & \times & \times \\ \square & \times & \times \\ \square & \times & \times \\ \square & \times & \times \\ \square & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \alpha_1 & \times & \times \\ 0 & \triangle & \times \\ 0 & \triangle & \times \\ 0 & \triangle & \times \\ 0 & \triangle & \times \end{bmatrix}}_{A^{(1)}}$$

Tag nu H_2 så att H_2 applicerat på $\triangle$-vektorn blir en multipel av e_1. Observera att denna vektor har 4 och inte 5 element och att H_2 är en 4×4-matris. Vi multiplicerar givetvis inte med ettor eller nollor utan arbetar endast med element under linjen och i andra och tredje kolonnen. Jag har bytt beteckningar så H_2 och H_3 är inte samma matriser som på föregående sida.

$$\underbrace{\left[\begin{array}{c|cc} 1 & 0 & \\ \hline 0 & \multicolumn{2}{c}{H_2} \end{array}\right] \left[\begin{array}{cc} \alpha_1 & \times \\ \hline 0 & \triangle \\ 0 & \triangle \\ 0 & \triangle \\ 0 & \triangle \end{array}\right]}_{A^{(1)}} = \underbrace{\left[\begin{array}{cc} \alpha_1 & \times \\ \hline 0 & \alpha_2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right]}_{A^{(2)}}$$

Vi väljer nu H_3 som applicerat på $\diamond$-vektorn blir en multipel av e_1. Observera att denna vektor har 3 element och att H_3 är en 3×3-matris.

$$\underbrace{\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & \multicolumn{1}{c}{H_3} \end{array}\right] \left[\begin{array}{cc} \alpha_1 & \times \\ 0 & \alpha_2 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right]}_{A^{(2)}} = \underbrace{\left[\begin{array}{cc} \alpha_1 & \times \\ 0 & \alpha_2 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right]}_{A^{(3)}} = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right]$$

Vi sammanställer och får:

$$\underbrace{\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & \multicolumn{1}{c}{H_3} \end{array}\right] \left[\begin{array}{c|cc} 1 & 0 & \\ \hline 0 & \multicolumn{2}{c}{H_2} \end{array}\right] H_1}_{H} \quad A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

så att

$$A = H^{-1} \begin{bmatrix} R \\ 0 \end{bmatrix} = H^T \begin{bmatrix} R \\ 0 \end{bmatrix} = [Q, Z] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Q består av de tre första kolonnerna i H^T och de resterande två bildar Z.

Om vi bara vill lösa minstakvadratproblemet behöver vi aldrig bilda Q (eller H) explicit utan det räcker att känna R och $Q^T b$ (vi löser ju $Rx = Q^T b$). Så i vårt exempel ges R av de tre första raderna i $A^{(3)}$ och $A^{(3)}$ har vi ju bildat utan att explicit bilda H.

$Q^T b$ utgörs av de tre första raderna i Hb. Varför? $H^T = [Q, Z]$ så att:

$$Hb = [Q, Z]^T b = \begin{bmatrix} Q^T \\ Z^T \end{bmatrix} b = \begin{bmatrix} Q^T b \\ Z^T b \end{bmatrix}$$

Men vi kan ju bilda Hb genom att applicera H_1, H_2 och H_3 på b och sedan plocka ut de tre första elementen. Vi behöver således inte Q här heller.

Notera att vi inte måste spara alla u-vektorer som bildar H-matriserna heller. I praktiken brukar man skriva över A med R (plus nollor) så vi behöver inget extra matris-minne.

Illakonditionerade problem, regularisering

Tyvärr räcker inte alltid ovanstående metoder till. Antag att vi har beräknat en QR-faktorisering och att R har utseendet (hårt avrundat):

$$R = \begin{bmatrix} 2.4 & -1.7 & 3.2 \\ 0 & 1.2 \cdot 10^{-5} & 3 \\ 0 & 0 & 2.3 \cdot 10^{-10} \end{bmatrix}$$

I detta fall är R (och därmed A) mycket illa konditionerad. $\kappa(A) \approx 7 \cdot 10^{14}$.

När vi löser $Rx = Q^T b$ kommer vi (implicit) att invertera R. De små diagonalelementen kommer då att bestämma hur lösningen ser ut eftersom R^{-1} kommer att innehålla inversen av dessa element (bland annat). T.ex. gäller att $(R^{-1})_{kk} = 1/r_{kk}$.

Frågan är nu hur mycket man litar på de små r_{kk}. Är de säkra värden eller består de bara av mätfel och avrundningsfel?

Vi litar nog mycket på r_{11}-elementet, det är ju relativt stort. r_{22} tror vi kanske rätt mycket på, men $r_{33} = 2.3 \cdot 10^{-10}$ kanske vi tvivlar på. Problemet är att $1/r_{33}$ i stor utsträckning kommer att bestämma utseendet på lösningen x (om inte b är väldigt speciell).

Det säkra värdet, 2.4 bidrar knappast något till lösningen, eftersom $1/2.4$ är så litet. Hur mycket vi litar på värdena beror på mätfel och modell.

Om man tror att ett värde helt eller delvis består av brus är det ingen mening att bara räkna på. Den lösning man får fram är nog tämligen meningslös. GIGO = Garbage In, Garbage Out.

Hur kan ett sådant illa konditionerat problem uppstå? Det finns flera orsaker. På nästa följer några exempel.

Exempel: En olämplig modell. Säg att vi har modellen

$$b = x_1 t + x_2 \sin t$$

Problemet är att $\sin t \approx t$ för $t \approx 0$. $\sin t = t - t^3/6 + \cdots$. Vi har därför nästan modellen $b = (x_1 + x_2)t$ och motsvarande minstakvadratproblem har inte entydig lösning.

Valet av fysikaliska enheter kan orsaka problem. Antag att vi har modellen $b = x_1 + x_2 t + x_3 t^2$ och att $A = QR$ där R är välkonditionerad.

Ett enhetsbyte för t svarar mot en skalning σt. Sambandet $A = QR$ övergår då i $AD = Q(RD)$, $D = \mathrm{diag}(1, \sigma, \sigma^2)$.
Det gäller att $\kappa(A) = \kappa(R) = ||R||_2 \; ||R^{-1}||_2$ varför

$$\kappa(AD) = \kappa(RD) = ||RD||_2 \; ||(RD)^{-1}||_2 \leq$$

$$||R||_2 \; ||D||_2 \; ||R^{-1}||_2 \; ||D^{-1}||_2 = \kappa(R)\kappa(D) = \kappa(R) \; \max(\sigma^{-2}, \sigma^2)$$

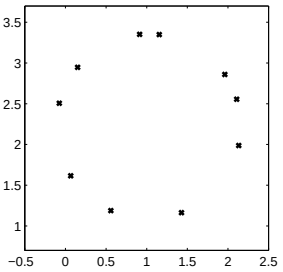
Så att byta från t.ex. sekunder till ms kan öka konditionstalet med 10^6. Lika illa är att byta till ks.

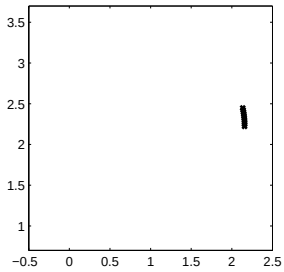
Vissa problem är illa konditionerade i sig (redan innan vi gjort någon olämplig matematik), det gäller många sk inversa problem, t.ex. bildrekonstruktion. Detta används inom bland annat kriminologi och astronomi.

Framåtriktningen är då att vi tar ett fotografi, vi avbildar ett motiv. Den omvända riktningen (inversen) att rekonstruera motivet givet ett fotografi som är suddigt (av rörelseoskärpa t.ex.). Problemet är illa konditionerat pga att flera motiv kan passa ihop med ett suddigt fotografi. En annan viktig tillämpning är datortomografi.

Ett enkelt exempel: vi har punkter i planet som vi tror ligger på en cirkel och vi vill bestämma cirkelns centrum och radie.
Efter variabeltransformation kan problemet skrivas som ett linjärt minstakvadratproblem.

Vi får en säker bestämning om vi har mätpunkter fördelade över cirkeln. Om punkterna grupperar ihop sig utmed en liten del av cirkeln, är problemet att bestämma den illa konditionerat. Om punkterna sammanfaller finns oändligt många lösningar (cirklar).





Ibland kan man inte välja hur man ska mäta, utan man får nöja sig med de data man har. För att kunna få ett vettigt svar måste lösningsprocessen stabiliseras med sk regularisering.

Antag att $||R||_2 \approx 1$ och $||b||_2 \approx 1$. När R är illakonditionerad så är $||R^{-1}||_2$ stor vilket medför att $||x||_2$ blir stor. Ett indirekt sätt att minska inflytandet av de små osäkra värdena i R är att begränsa längden på lösningen, en enkel variant är:

$$\min_x \; ||Ax - b||_2 \quad \text{med} \quad ||x||_2 \leq \alpha$$

ett annat sätt är att ändra direkt i R. α måste sättas av den som löser problemet. Hur man gör detta på ett vettigt sätt tar jag inte upp.

Ickelinjära ekvationer

$$f(x) = 0, \quad f : \Re \to \Re$$

Vi kan också ha system av ekvationer:

$$\begin{cases} f(x,y,z) = 0 \\ g(x,y,z) = 0 \\ h(x,y,z) = 0 \end{cases}$$

Exempel:

$$\begin{cases} x^2 + y^2 - 2 = 0 \\ x - y = 0 \end{cases}$$

med rötter $(1,1)$ och $(-1,-1)$.

En ickelinjär ekvation kan ha $0, 1, 2, 3, \ldots, \infty$ lösningar.

Ett linjärt problem ($Ax = b$) har $0, 1$ eller ∞ många.

Det kan tänkas att f är definierad via en procedur.

$$f(x) = \int_{-4}^{x} (1+t)e^{-t^2} \sin t \; dt$$

Flertalet metoder:

- Startas med en (eller flera) approximation(er).
- Skapar en sekvens av approximationer som förhoppningsvis konvergerar mot nollstället.
- Kan divergera.
- Försöker att hitta ett nollställe åt gången.

Halveringsmetoden (bisektionsmetoden)

Givet en kontinuerlig funktion f och $p, n \in \Re$ med $f(n) < 0$, $f(p) > 0$.

```
while |n − p| > tol do
    m = (n + p)/2
    if f(m) < 0 then     ! borde ta hand om exakt likhet också
        n = m
    else
        p = m
    endif
end
```

Om begynnelseintervallet har längden τ har intervallet längden

$$\frac{\tau}{2^k}$$

efter k iterationer.

Halveringsmetodens fördelar

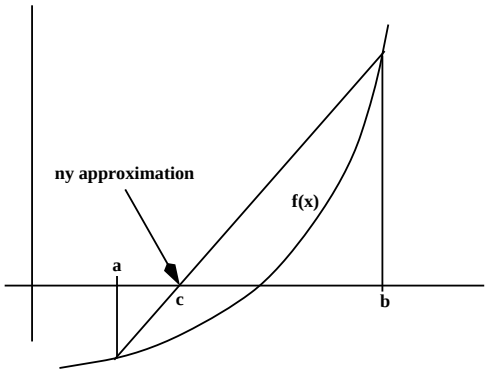
- konvergerar alltid
- räcker att f är kontinuerlig
- får ett intervall där roten ligger
- deterministisk i antal steg

och nackdelar

- kan ej generaliseras till system
- långsam
- kan vara svårt att hitta p och n

"långsam men säker"

Snabbare metoder: lös ett svårt problem genom att lösa en sekvens av enklare problem.
Linjärisering, approximera f med en linjär funktion.



Sekanten (den räta linjen) har ekvationen

$$y(x) = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

varför c ges av

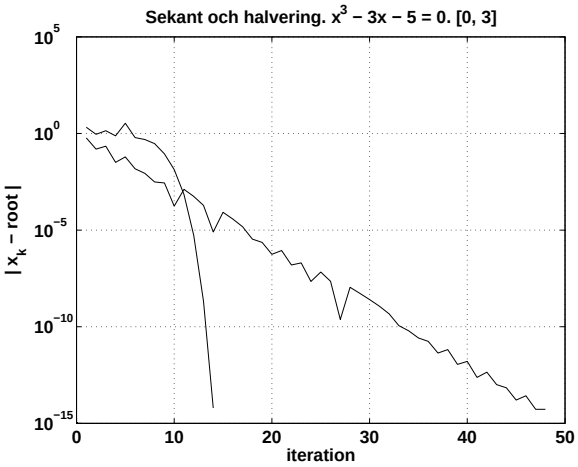
$$c = a - f(a)\frac{b - a}{f(b) - f(a)} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

Iterera: givet två startvärden x_0, x_1

$$x_{k+1} = x_k - f(x_k)\frac{x_{k-1} - x_k}{f(x_{k-1}) - f(x_k)}, \quad k = 1, 2, \ldots$$

Om f är linjär ger detta nollstället i ett steg.

Exempel: $f(x) = x^3 - 2x - 5$.



Sekantmetoden konvergerar i en iteration om funktion är linjär, ett faktum som använts i flera hundra år.

'A discussion of sheep' is a problem taken from Robert Record's "Grounde of Artes" (London, 1542?).

There is supposed a lawe made that (for furtheryng of tyllage) every man that doth kepe shepe, shall for every 10 shepe eare and sowe one acre of grounde, and for his allowance in sheepe pasture there is appointed for every 4 shepe one acre of pasture. Nowe is there a ryche shepemaister whyche hath 7000 akers of grounde, and would gladlye kepe as manye sheepe as he myght by that statute. I demaunde howe many shepe shall he kepe?

Fyrste I suppose he maye kepe 500 shepe, and for them he shall have in pasture, after the rate of 4 shepe to an acre, 125 akers, and in earable grounde 50 acres that is 175 in all, but this errour is to litell by 6825. Therefore I gesse agayn that he maye kepe 1000 shepe, that is in pasture 250 akers, and in tyllage 100 akers, which maketh 350, that is to lytle by 6650.

These bothe erroures with theyr positions I sette downe as you see, and multiply in crosse 6825 by 1000, and it maketh 6825000. Then I multiply 6650 by 500, and it doth amount to 3325000, which sum I do subtract out of the fyrst, & there remaineth 3500000, as the dividende. Also I doo subtract the lesser errour out of the greater, and so remayneth 175, by which I divide the said dividende, and the quotient will be 20000, so that I see that by this rate he that hathe 7000 acres of ground may keepe 20000 shepe: & therby I conjecture that many menne may kepe so many shepe: for many men (as the common talke is) have so many acres of grounde.

The 'equals' symbol '=' appears in Recorde's book "The Whetstone of Witte" published in 1557. He justifies using two parallel line segments "bicause noe 2 thynges can be moare equalle".

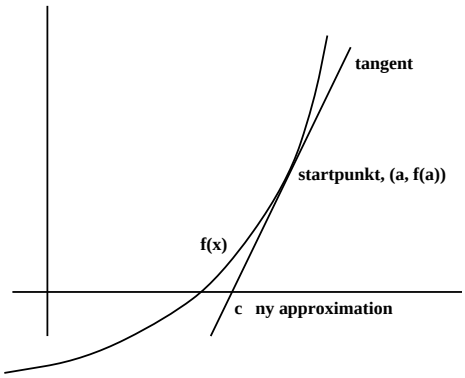
Problem 12, kapitel 7 i "Nio kapitel om matematik".
(1 cùn = $\frac{1}{10}$ chǐ)

Two rats gnawing towards each other

Jīn yǒu héng hòu wǔ chǐ,	Now let there be a wall of thickness 5 chǐ,
liǎng shǔ duì chuān.	two rats gnawing towards (each other).
Dà shǔ rì yī chǐ,	The big rat gnaws on the first day 1 chǐ,
xiǎo shǔ yì rì yī chǐ,	the small rat also on the first day 1 chǐ.
Dà shǔ rì zì bèi,	The big rat gnaws everyday twice itself,
xiǎo shǔ rì zì bàn,	The small rat gnaws everyday half itself.
Wèn	Question:
jǐ hé rì xiāng féng,	(In) how many days (do they) meet,
gè chuāng jǐ hé.	how much has each one gnawn?
Dá yuē:	The answer says:
èr rì shí qī fēn rì zhī èr.	(In) $2\frac{2}{17}$ days.
Dà shǔ chuān	The big rat has gnawn
sān chǐ sì cùn shí qī fēn cùn zhī shí èr,	3 chǐ $4\frac{12}{17}$ cùn,
xiǎo shǔ chuān	the small rat has gnawn
yī chǐ wǔ cùn shí qī fēn cùn zhī wǔ.	1 chǐ $1\frac{5}{17}$ cùn.
Shù yuē:	The rule says:
Jiǎ lìng èr rì,	Assuming 2 days,
bù zú wǔ cùn.	the deficit is 5 cùn.
Lìng zhī sān rì,	Let it (be) 3 days,
yǒu yú sān chǐ qī cùn bàn.	there is 3 chǐ $7\frac{1}{2}$ cùn in excess.

Newtons metod

Kan approximera med tangenten i stället för med sekanten (Newton-Raphson, 1690).



Tangenten har ekvationen:

$$y = f(a) + f'(a)(x - a)$$

När $x = c$ är $y = 0$

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

Iterera

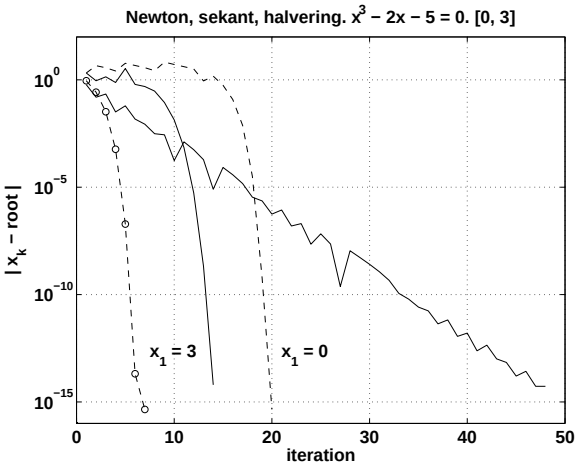
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Kräver endast ett startvärde, men måste ha derivatan.

Exempel: Newtons eget exempel $x^3 - 2x - 5 = 0$.
Iterationen blir

$$x_{k+1} = x_k - \frac{x_k^3 - 2x_k - 5}{3x_k^2 - 2}$$

I bilden nedan har vi testat med de två startvärdena $x_0 = 0$ respektive $x_0 = 3$.



Startvärdet viktigt! (roten = 2.0946)

Hur skall vi karakterisera de olika konvergenshastigheterna?

Om $f(x^*) = 0$ och $\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^r} = C$ konstant $< \infty$

så säger vi att metoden har konvergensordning r

- om $r = 1$ och $C < 1$ så har vi linjär konvergens
- om $r > 1$ så har vi superlinjär konvergens
- om $r = 2$ så har vi kvadratisk konvergens

Vad innebär $r = 1$? Om x_0 ligger tillräckligt när x^* så gäller att:

$$|x_1 - x^*| \approx C|x_0 - x^*|, \quad |x_2 - x^*| \approx C|x_1 - x^*| \approx C^2|x_0 - x^*|$$

Dvs. $|x_k - x^*| \approx C^k|x_0 - x^*|$

Exempel: $|x_{k+1} - x^*| \approx C|x_k - x^*|^r$. Antag att $|x_0 - x^*| = 0.1$ och $C = 0.1$, då är $|x_k - x^*|$:

k	linjär r = 1	superlinjär r = 1.618	kvadratisk r = 2
0	1e-1	1e-1	1e-1
1	1e-2	2e-3	1e-3
2	1e-3	6e-6	1e-7
3	1e-4	3e-10	1e-15
4	1e-5	5e-17	1e-31

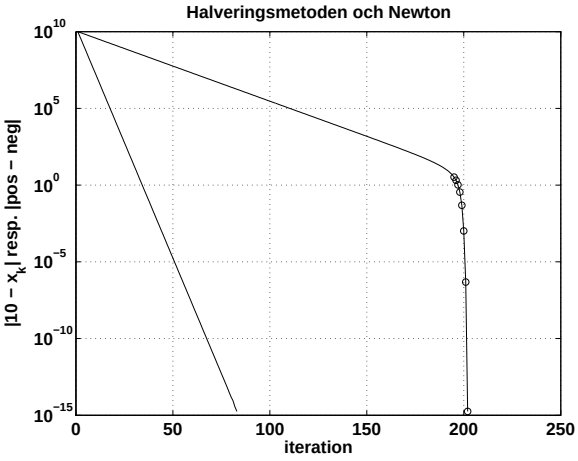
Normalt (nära lösningen för sekant och Newton) är:

- Halveringsmetoden linjär med $C = 0.5$.
- Sekantmetoden superlinjär med $r = (1 + \sqrt{5})/2 \approx 1.618$
- Newtons metod kvadratiskt konvergent (om enkelrot)

Exempel: lös med Newtons metod och halveringsmetoden

$$x^{10} - a = 0, \quad a = 10^{10}$$

Använd det urusla startvärdet a ($[0, a]$ för halveringsmetoden). Uruselt eftersom $x^* = 10$.



Konvergensordningen är definierad som ett gränsvärde. Det kan krävas många steg innan x_k ligger så nära x^* att den kvadratiska konvergensen sätter igång.

Om den kommer igång. Både Newtons metod och sekantmetoden kan divergera. "Snabba men osäkra."

Hybridmetoder: använd "dyra" Newton där det lönar sig och en billig metod för övrigt.

Vilken metod är billigare, Newton eller sekant?

Sekantmetoden kräver ett funktionsvärde i varje steg.
Newton kräver både ett funktionsvärde och en derivata men konvergerar snabbare (nära nollstället).

Vi är normalt inte intresserade av att minimera antalet iterationer. Det viktiga är det totala körtiden och minnesbehovet.

- få komplexa iterationer
- många enkla iterationer

Den metodoberoende feluppskattningen

Givet approximationen $\hat{x}$ och det exakta värdet x^* hur skall vi uppskatta $|\hat{x} - x^*|$?
Det vi kan beräkna är "residualen" $f(\hat{x})$.

Medelvärdessatsen:

$$f(\hat{x}) = f(x^*) + (\hat{x} - x^*)f'(\xi), \quad \xi \in (\hat{x}, x^*)$$

Antag att $f'(\xi)$ är kontinuerlig med $M = \min |f'(\xi)|, \xi \in [\hat{x}, x^*]$.

Om då $M > 0$ gäller att:

$$|\hat{x} - x^*| \leq |f(\hat{x})|/M$$

M kan vara noll. Tag $f(x) = x^2$ (så nollan är dubbelrot).
Då är både $f(0) = 0$ och $f'(0) = 0$.

Exempel:

$$f(x) = 1/x - 1/10, \quad \hat{x} = 11, \quad f(\hat{x}) = 1/11 - 1/10 = -1/110,$$

$$|f'(\xi)| = 1/\xi^2, \quad |\hat{x} - x^*| \leq \frac{|1/11 - 1/10|}{1/11^2} = 1.1$$

f är strängt avtagande med $f(9) > 0$ och $f(11) < 0$ varför $(9, 11)$ innehåller precis en rot. Beloppet av derivatan är $1/x^2$ som är strängt avtagande. Det minsta värdet på derivatan, i intervallet, är därför $1/11^2$.

Avbrottskriterium

I sekantmetoden får vi inte en sekvens av intervall som innehåller roten. Metoden kan ju till och med divergera. Så, hur vet vi när vi skall avsluta iterationen? Vi har ett avbrottskriterium som kontrollerar:

- k, för att undvika oändliga loopar (divergens, eller för små toleranser)
- $|x_k - x_{k-1}|$, borde gå mot noll vid konvergens, men litet värde kan betyda att det går långsamt
- $|f(x_k)|$, noll i lösningen (tänk också på $|f(x_k)|/M$)

Första försöket: avsluta om ($\vee$ = eller):

$$k > maxit \quad \vee \quad |x_k - x_{k-1}| \leq tol_x \quad \vee \quad |f(x_k)| \leq tol_f$$

$maxit$ (max antal iterationer), tol_x och tol_f ges av användaren. Man kan givetvis kräva att $|x_k - x_{k-1}| \leq tol_x$ & $|f(x_k)| \leq tol_f$. Inte skalningsoberoende: $10^5 f(x) = 0$ borde helst fungera lika bra som $f(x) = 0$. Motsvarande för $f(10^5 x) = 0$. Toleranserna måste skalas efter problemet.

Andra försöket: avsluta om:

$$k > maxit \quad \vee \quad |x_k - x_{k-1}| \leq tol_x|x_0| \quad \vee \quad |f(x_k)| \leq tol_f|f(x_0)|$$

Fungerar inte om $x_0 = 0$. Vi skulle kanske kunna uppskatta derivatan för att få något i stil med $|\hat{x} - x^*| \leq |f(\hat{x})|/M$.

Det är inte enkelt att utforma ett säkert och effektivt kriterium. Ett kriterium går alltid att lura eftersom vi endast känner funktionen (och kanske derivatan) i ett ändligt antal punkter. Det finns oändligt många funktioner som går genom dessa punkter.

Newton för system

Repetition av Taylors formel.

$$\begin{bmatrix} f(a+h, b+k) \\ g(a+h, b+k) \end{bmatrix} = \begin{bmatrix} f(a,b) \\ g(a,b) \end{bmatrix} + \begin{bmatrix} \frac{\partial f(a,b)}{\partial x}h + \frac{\partial f(a,b)}{\partial y}k \\ \frac{\partial g(a,b)}{\partial x}h + \frac{\partial g(a,b)}{\partial y}k \end{bmatrix} + \ldots =$$

$$\begin{bmatrix} f(a,b) \\ g(a,b) \end{bmatrix} + \begin{bmatrix} \frac{\partial f(a,b)}{\partial x} & \frac{\partial f(a,b)}{\partial y} \\ \frac{\partial g(a,b)}{\partial x} & \frac{\partial g(a,b)}{\partial y} \end{bmatrix} \begin{bmatrix} h \\ k \end{bmatrix} + \ldots$$

Matrisen av partiella derivator kallas Jacobianen.

Vi står i (x_j, y_j) och vill hitta korrektioner, (h, k), så att $f(x_j + h, y_j + k) = 0$ och $g(x_j + h, y_j + k) = 0$.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} f(x_j + h, y_j + k) \\ g(x_j + h, y_j + k) \end{bmatrix} \approx \begin{bmatrix} f(x_j, y_j) \\ g(x_j, y_j) \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{\partial f(x_j,y_j)}{\partial x} & \frac{\partial f(x_j,y_j)}{\partial y} \\ \frac{\partial g(x_j,y_j)}{\partial x} & \frac{\partial g(x_j,y_j)}{\partial y} \end{bmatrix}}_{J(x_j,y_j)} \begin{bmatrix} h \\ k \end{bmatrix}$$

Om Jacobianen, J, är ickesingulär kan vi få de approximativa korrektionerna:

$$\begin{bmatrix} h \\ k \end{bmatrix} \approx -J^{-1}(x_j, y_j) \begin{bmatrix} f(x_j, y_j) \\ g(x_j, y_j) \end{bmatrix}$$

Iterera!

$$\begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} - J^{-1}(x_j, y_j) \begin{bmatrix} f(x_j, y_j) \\ g(x_j, y_j) \end{bmatrix}$$

Jämför med det skalära fallet:

$$x_{j+1} = x_j - f(x_j)/f'(x_j)$$

Vi räknar naturligtvis inte ut inversen utan löser systemet:

$$J(x_j, y_j) \begin{bmatrix} h \\ k \end{bmatrix} = - \begin{bmatrix} f(x_j, y_j) \\ g(x_j, y_j) \end{bmatrix}$$

Exempel : $\begin{cases} x^2+y^2-2=0 \\ xy-\frac{1}{2}=0 \end{cases}$

Våra funktioner är alltså: $\begin{cases} f(x,y)=x^2+y^2-2 \\ g(x,y)=xy-\frac{1}{2} \end{cases}$

Newtons metod blir:

$$\begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} 2x_j & 2y_j \\ y_j & x_j \end{bmatrix}^{-1} \begin{bmatrix} x_j^2+y_j^2-2 \\ x_jy_j-\frac{1}{2} \end{bmatrix}$$

Om vi startar med $x_0=-3$ och $y_0=10$ så får vi följande approximationer:

```
  -3.0000e+00  -1.4121e+00  -5.4236e-01  -1.4188e-02
   1.0000e+01   5.1264e+00   2.8033e+00   1.8081e+00

   2.7380e-01   3.5877e-01   3.6597e-01   3.6603e-01
   1.4593e+00   1.3733e+00   1.3661e+00   1.3660e+00

   3.6603e-01   3.6603e-01   3.6603e-01
   1.3660e+00   1.3660e+00   1.3660e+00

>> fel =
  -3.3660e+00  -1.7781e+00  -9.0838e-01  -3.8021e-01
   8.6340e+00   3.7603e+00   1.4373e+00   4.4208e-01

  -9.2230e-02  -7.2583e-03  -5.1931e-05  -2.6966e-09
   9.3297e-02   7.2586e-03   5.1931e-05   2.6966e-09

            0            0            0
            0            0            0
```


Om man arbetar med stora system kan man inte ha variabler $x, y, z, w, \ldots$ utan vi får använda vektorer, analogt för funktionerna.

Exemplet kan skrivas på följande vis. x och y får vara elementen x_1 respektive x_2 i vektorn (kolonnmatrisen) x.

$$\begin{cases} x_1^2+x_2^2-2=0 \\ x_1x_2-\frac{1}{2}=0 \end{cases}$$

Vår funktion, f, med två komponenter är:

$$\begin{cases} f_1(x_1,x_2)=x_1^2+x_2^2-2 \\ f_2(x_1,x_2)=x_1x_2-\frac{1}{2} \end{cases}$$

Normalt skriver vi bara $f(x)=0$ där f, x och 0 är vektorer. f är alltså en vektorvärd funktion som beror av en vektor.

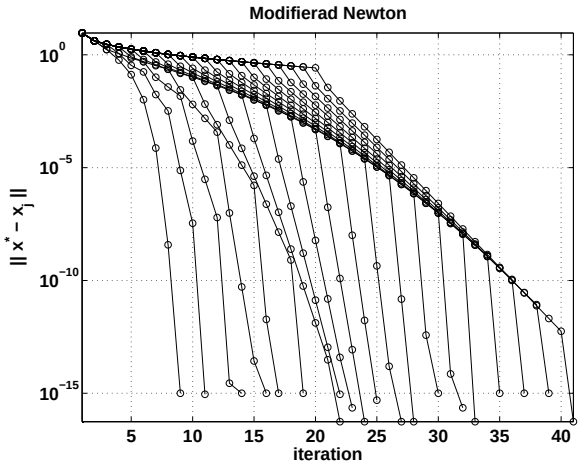
Newtons metod blir (notera placeringen av iterationsindex):

$$\begin{bmatrix} x_1^{(j+1)} \\ x_2^{(j+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix} - \begin{bmatrix} 2x_1^{(j)} & 2x_2^{(j)} \\ x_2^{(j)} & x_1^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} (x_1^{(j)})^2+(x_2^{(j)})^2-2 \\ x_1^{(j)}x_2^{(j)}-\frac{1}{2} \end{bmatrix}$$

Allmänt:

$$x^{(j+1)} = x^{(j)} - J^{-1}(x^{(j)})f(x^{(j)})$$

Dyrt och komplicerat att beräkna $J(x)$. Alternativ? Modifierad Newton: Beräkna $J(x^{(j)})$ då och då (inte i varje iteration).



Differensapproximation av J; slipper beräkna de n^2 derivatorna explicit. Om f är given via en algoritm kanske det inte är möjligt att beräkna derivatorna. Välj ett lämpligt tal δ (se övning):

$$f(x+\delta e_i) \approx f(x) + \delta J e_i$$

eller

$$Je_i \approx \frac{f(x+\delta e_i)-f(x)}{\delta}$$

Fixpunkter och lite teori

Upprepade tryckningar på `cos`-knappen. Tre olika startvärden.

```
  -5.0000e+00            0   2.0000e+01
   2.8366e-01   1.0000e+00   4.0808e-01
   9.6004e-01   5.4030e-01   9.1788e-01
   5.7349e-01   8.5755e-01   6.0750e-01
   8.4001e-01   6.5429e-01   8.2108e-01
   6.6745e-01   7.9348e-01   6.8143e-01
   7.8540e-01   7.0137e-01   7.7667e-01
   7.0711e-01   7.6396e-01   7.1325e-01
   7.6025e-01   7.2210e-01   7.5624e-01
   7.2467e-01   7.5042e-01   7.2742e-01
   7.4872e-01   7.3140e-01   7.4689e-01
   7.3256e-01   7.4424e-01   7.3380e-01
   7.4346e-01   7.3560e-01   7.4263e-01
   7.3613e-01   7.4143e-01   7.3669e-01
   7.4107e-01   7.3751e-01   7.4070e-01
   7.3774e-01   7.4015e-01   7.3800e-01
   7.3999e-01   7.3837e-01   7.3982e-01
   7.3848e-01   7.3957e-01   7.3859e-01
   7.3949e-01   7.3876e-01   7.3942e-01
   7.3881e-01   7.3930e-01   7.3886e-01
```

Så, i varje kolumn har vi $\cos(\cos(\cos(\cos(\ldots\cos(x_0)\ldots))))$. Detta kan skrivas på formen $x_{k+1}=\cos x_k$.

- Iterationen verkar konvergera
- Gör den alltid det?
- Hur snabbt konvergerar det?
- Kan vi använda detta till något?

Om vi får konvergens gäller, i vårt exempel, att $x = \cos x$, dvs. gränsvärdet är lösningen till en ekvation.

```
>> [x, cos(x), x - cos(x)]
ans = 7.3909e-01   7.3909e-01   0
```

Låt oss trycka x^2-knappen istället. Vi noterar först att om $x_0 \leq 0$ så är alla efterföljande värden ickenegativa. Det räcker att studera ickenegativa värden med andra ord.

Tre olika saker kan inträffa:

1. om $0 \leq x_0 < 1$, så konvergerar värdena mot 0.
   T.ex. $0.1, 0.01, 0.0001, \ldots$
2. $x_0 = 1$ medför att vi stannar i ett
3. $x_0 > 1$ medför att $x_k \to \infty$

Punkten ett är "repulsiv" i den meningen att oavsett hur nära vi startar ett (om vi inte startar precis i ettan) så stöts vi därifrån.

Nollan "attraherar". Om $|x_0| < 1$ så konvergerar följden mot noll.

Vi kommer att studera iterationer av typen $x_{k+1} = g(x_k)$
(inte enbart "knapptryckningsfunktioner").
Om x_k konvergerar, $x_k \to x^*$, gäller att $x^* = g(x^*)$.
Vi kallar g fixpunktsiteration och x^* fixpunkt.
Startar vi i en fixpunkt får vi tillbaks den.

Vi har två syften med de följande sidorna:

- givet en ekvation, $f(x) = 0$, hitta en fixpunktsiteration, g, som har en attraktiv fixpunkt, x^*, sådan att $f(x^*) = 0$.
- vi vill förstå vilka egenskaper hos g som ger konvergens

Newtons metod är en speciell fixpunktsiteration, ty

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad x_{k+1} = g(x_k) \quad \text{med} \quad g(x) = x - \frac{f(x)}{f'(x)}$$

Om Newtons metod konvergerar mot x^* gäller i gränsen att

$$x^* = x^* - \frac{f(x^*)}{f'(x^*)}$$

dvs $f(x^*) = 0$. Så fixpunkten är en lösning till vårt problem.

När konvergerar en fixpunktsiteration?
Dvs om det existerar x^* så att $x^* = g(x^*)$, när gäller att

$$\lim_{k\to\infty} |x_k - x^*| = 0 \ ?$$

Idé: konvergens medför att felet, $|x_k - x^*|$, minskar dvs.
$|x_{k+1} - x^*| < |x_k - x^*|$, så låt oss studera felet.

$$x_{k+1} - x^* = g(x_k) - x^* = g(x^* + x_k - x^*) - x^* =$$
$$g(x^*) + (x_k - x^*)g'(\theta_k) - x^* = g'(\theta_k)(x_k - x^*), \quad \theta_k \in (x_k, x^*)$$

Så,

$$|x_{k+1} - x^*| = |g'(\theta_k)|\ |x_k - x^*|$$

Ett steg till:

$$|x_{k+2} - x^*| = |g'(\theta_{k+1})|\ |x_{k+1} - x^*| = |g'(\theta_{k+1})|\ |g'(\theta_k)|\ |x_k - x^*|$$

Alltså:

$$|x_k - x^*| = |g'(\theta_{k-1})| \cdots |g'(\theta_1)|\ |g'(\theta_0)|\ |x_0 - x^*|$$

Så om det finns ett tal, λ, där alla $|g'(\theta_k)| \leq \lambda < 1$ får vi konvergens.

$$|x_k - x^*| \leq \lambda^k |x_0 - x^*|$$

Följande villkor garanterar konvergens:

- x_0 tillräckligt nära x^*
- g kontinuerligt deriverbar med $|g'(x^*)| < 1$

Den andra punkten medför att det existerar ett intervall, $I = [x^* - \delta, x^* + \delta]$ sådant att $|g'(x)| \leq \lambda < 1, x \in I$.

Om vi ser till att starta tillräckligt nära x^*, så stannar alla x_k kvar i intervallet. Detta medför att alla $\theta_k \in I$.

Första steget: Om $x_0 \in I$ så gäller att $\theta_0 \in I$, varför $|g'(\theta_0)| \leq \lambda$ vilket medför att $x_1 \in I$. Induktion!

Normalt linjär konvergens; ju mindre $|g'(x^*)|$ desto snabbare konvergens:

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} \to |g'(x^*)|$$

Newton?

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

så

$$g'(x^*) = 1 - \frac{(f'(x^*))^2 - f''(x^*)f(x^*)}{(f'(x^*))^2} = 0, \quad \text{om } x^* \text{ enkelrot}$$

Innebär (minst) kvadratisk konvergens (inte att det konvergerar i ett steg).

Några exempel:

$g(x) = x^2$ har vi redan analyserat. $x_{k+1} = g(x_k)$ eller $x_{k+1} = x_k^2$.
Fixpunkter? $g(x^*) = x^*$ eller $(x^*)^2 = x^*$ så $x^* = 0$ eller $x^* = 1$.
Konvergens? $g'(x) = 2x$ och $g'(0) = 0$ så bättre än linjär konvergens $g'(1) = 2$ divergens.

$x_0 = 10^{-1}, x_1 = 10^{-2}, x_2 = 10^{-4}, \ldots$.

$g(x) = x/2$. Fixpunkter: $x^* = x^*/2$ så $x^* = 0$. Konvergens?
$g'(x^*) = 1/2$. Linjär konvergens: $x_0 = 1, x_1 = 1/2, x_2 = 1/4, \ldots$.

$g(x) = \cos x$. Fixpunkter: $x^* = \cos x^*$ så $x^* \approx 0.739$.
Konvergens? $g'(x^*) = -\sin x^*$ och $|-\sin x^*| \approx 0.674 < 1$
så linjär konvergens.

Lös $x^2 - 2 = 0$. Vi kan ju använda Newtons metod, men låt oss testa med omskrivningen $[x^2 - 2]/\alpha + x = x$ och tag $g(x) = [x^2 - 2]/\alpha + x$. Fixpunkterna är rötterna till ekvationen.
Konvergens? $g'(x) = 2x/\alpha + 1$. Tar vi t.ex. $\alpha = -3$ så får vi rätt snabb konvergens ty $|g'(\sqrt{2})| = |-2\sqrt{2}/3 + 1| \approx 0.05719$.

```
>> x = 1; for k=1:9, x(k+1)=x(k)-(x(k)^2 - 2) / 3; end
>> d = x - sqrt(2)  % editerat
d = -4.1e-01 -8.0e-02 -6.8e-03 -4.0e-04 -2.3e-05
    -1.3e-06 -7.5e-08 -4.3e-09 -2.4e-10 -1.4e-11

>> abs(d(2:end) ./ d(1:end-1))
ans = 1.9526e-01  8.4151e-02  5.9460e-02  5.7326e-02
      5.7199e-02  5.7191e-02  5.7191e-02  5.7191e-02
      5.7192e-02
```

Interpolation

Exempel:
Gymnasiet på "den gamla goda tiden", räknesticka och tabeller.

Vi vill beräkna $\sqrt{1.245}$ och har en tabell över
$y = \sqrt{t}$ där $t = 0, 0.01, 0.02, \ldots, 9.99, 10.00$.
y-värdena är givna med fem siffror.

Mer realistiskt, nu för tiden, vore en tabell, $t_k, y_k, k = 1, \ldots, n$, där vi av någon anledning inte kan beräkna $y(t)$ för alla t (mättekniska problem, gamla data).

Hur ska vi gå tillväga. I min skoltabell fanns röda tal mellan y-värdena, differenser, för att underlätta linjär interpolation

t	y
...	
1.22	1.1045_{45}
1.23	1.1091_{46}
1.24	1.1136_{45}
1.25	1.1180_{44}
1.26	1.1225_{45}
1.27	1.1269_{44}

44 i 1.1180_{44} ska tolkas som $10^4(1.1180 - 1.1136)$. Så

$$\sqrt{1.245} \approx 1.1136 + \frac{1.245 - 1.24}{1.25 - 1.24} 0.0044 = 1.1158, \text{ fel} \approx -4.3 \cdot 10^{-6}$$

Andra tillämpningar som nyttjar interpolation är kvadratur (integration), lösning av randvärdesproblem, förenkling av funktioner, härledning av metoder (t.ex. sekantmetoden).
Allmänt har vi (t_k, y_k), $k = 1, \ldots, m$ med $t_1 < t_2 < \cdots < t_m$ och vill hitta en funktion (polynom i denna kurs), p, så att $p(t_k) = y_k$, $k = 1, \ldots, m$. Ibland lägger man dessutom krav på derivator, sk Hermite-interpolation.

Låt oss anta att det finns en bakomliggande funktion, f, (i exemplet $\sqrt{\ }$) som vi vill interpolera. Denna funktion är inte alltid känd.

Vi känner y_1, y_2 som är approximationer av f i två punkter $t_1 < t_2$, $y_1 = f(t_1) + \delta_1$ samt $y_2 = f(t_2) + \delta_2$ och vi vill approximera $f(t)$ där $t_1 < t < t_2$.

Vi bestämmer nu interpolationspolynomet, p, som uppfyller interpolationsvillkoren: $p(t_1) = y_1$ samt $p(t_2) = y_2$. Två villkor bestämmer en konstant- eller en linjär funktion så vi kräver att p har grad ≤ 1. Ansätt $p(t) = x_1 + x_2 t$, vilket ger följande linjära ekvationssystem:

$$\begin{cases} x_1 + x_2 t_1 = y_1 \\ x_1 + x_2 t_2 = y_2 \end{cases} \Rightarrow \begin{cases} x_1 = (t_2 y_1 - t_1 y_2)/(t_2 - t_1) \\ x_2 = (y_2 - y_1)/(t_2 - t_1) \end{cases}$$

så att

$$p(t) = \frac{t_2 y_1 - t_1 y_2}{t_2 - t_1} + \frac{y_2 - y_1}{t_2 - t_1} t = y_1 + (t - t_1)\frac{y_2 - y_1}{t_2 - t_1}$$

Observera att den andra omskrivningen direkt svarar mot tabellräkningen. Felet $p(t) - f(t)$ kan skrivas enligt:

$$p(t) - f(t) = f(t_1) + \delta_1 + (t - t_1)\frac{f(t_2) - f(t_1) + \delta_2 - \delta_1}{t_2 - t_1} - f(t) =$$

$$\underbrace{f(t_1) + (t - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1}}_{p_f(t)} - f(t) + \underbrace{\delta_1 + (t - t_1)\frac{\delta_2 - \delta_1}{t_2 - t_1}}_{p_\delta(t)}$$

Låt oss införa de två polynomen p_f och p_δ sådana att $p_f(t_1) = f(t_1)$, $p_f(t_2) = f(t_2)$ resp. $p_\delta(t_1) = \delta_1$, $p_\delta(t_2) = \delta_2$. Då är tydligen $p = p_f + p_\delta$. Detta kan man direkt se från det linjära ekvationssystemet, lösningen x beror ju linjärt på högerledet.

$$p(t) - f(t) = p_f(t) + p_\delta(t) - f(t) = p_f(t) - f(t) + p_\delta(t)$$

De två delarna i felet kan tolkas som följer: $p_f(t) - f(t)$ anger hur väl p_f approximerar funktionsvärdena om de hade varit utan fel. $p_\delta(t)$ svarar mot felet i tabellvärden.

Låt oss nu uppskatta felet $e(t) = f(t) - p_f(t)$ (denna härledning kan rätt enkelt generaliseras till polynom av högre gradtal). Vi antar att $t \neq t_1, t_2$ ty $e(t_1) = e(t_2) = 0$. Inför

$$g(z) = e(z) - \frac{(z - t_1)(z - t_2)}{(t - t_1)(t - t_2)} e(t)$$

där vi betraktar t som en fix punkt, g beror alltså av z. Det gäller att $g(t_1) = g(t_2) = 0$ och dessutom är $g(t) = 0$. g har alltså tre distinkta nollställen varför, enligt medelvärdessatsen, $g'(z)$ har två distinkta nollställen. $g''(z)$ har alltså ett nollställe, kalla det $\theta \in (t, t_1, t_2)$ (det minsta intervall som innehåller t, t_1, t_2). Vi deriverar nu g (med avseende på z) och får (ty grad $p_f \leq 1$):

$$g''(z) = e''(z) - \frac{2\ e(t)}{(t - t_1)(t - t_2)} = f''(z) - \frac{2\ e(t)}{(t - t_1)(t - t_2)}$$

Eftersom $g''(\theta) = 0$ kan vi lösa ut $e(t)$ och får

$$e(t) = \frac{f''(\theta)}{2}\ (t - t_1)(t - t_2)$$

Antag att vi tar med fler punkter och interpolerar med ett polynom av högre gradtal. Kommer felet i approximationen att minska?

Vi kan först se på den allmänna satsen: Om p_f interpolerar f för de n t-värdena $t_1 < t_2 < \ldots < t_n$ så gäller att

$$f(t) - p_f(t) = \frac{f^{(n)}(\theta)}{n!}(t - t_1)(t - t_2)\cdots(t - t_n)$$

där $\theta \in (t, t_1, t_2, \ldots, t_n)$.

$n!$ ser lovande ut, men resten är inte så lätt att bedöma (θ känner vi inte t.ex.), så vi ser på vårt exempel i stället.

I exemplet kommer inte felet att minska eftersom det ser ut som:

$$p(t) - f(t) = p_f(t) - f(t) + p_\delta(t)$$

så även om vi kan göra $p_f(t) - f(t)$ mindre, så kommer $p_\delta(t)$, som svarar mot avrundningsfelet i tabellvärdena, att vara tämligen konstant, $10^{-5} - 10^{-6}$.

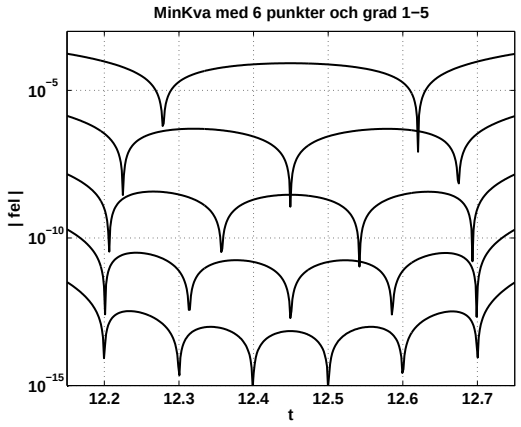
Situationen ändras om tabellvärdena hade givits med mindre fel, anta att $\delta_1 = \delta_2 = 0$. I exemplet hade då felet i approximationen varit 10^{-6} med två punkter (vårt förstagradspolynom), $\approx 10^{-8}$ med tre punkter (andragradspolynom), $\approx 10^{-10}$ med fyra punkter och $\approx 10^{-12}$ med fem punkter.

Observera att felet beror på hur t-punkterna ligger relativt den punkt där vi vill approximera f. I exemplet har jag lagt punkterna symmetriskt kring detta värde.

Så det kan löna sig att höja gradtalet förutsatt att tabellvärdena inte är behäftade med för stora fel. Polynom av höga gradtal är dock inte så lätthanterliga, mer om detta senare.

Kan vi använda polynomet för att extrapolera (gå utför $[t_1, t_n]$)? Vi vet att $|p(t)| \to \infty$ när $|t| \to \infty$ (om inte p är konstant), så det kan vara vanskligt. Polynom kan växa snabbt!

Hur passar minstakvadratanpassning in i detta sammanhang? Antag att vi anpassar mer än två (t_k, y_k)-punkter till ett förstagradspolynom. Kommer vi då att få en bättre approximation av det sökta värdet? Knappast. I bilden nedan har jag anpassat sex punkter, $t_k = 12.2 : 0.1 : 12.7$, ("exakta" y-värden) till polynom av grader 1-5 (när graden är fem har vi interpolation). Den lodräta axeln visar $|p_k(t) - f(t)|$, $k = 1, \ldots, 5$ (grad $p_k = k$).



Vi noterar att felet ökar utanför intervallet (extrapolation). Dipparna i felen svarar mot att polynomen interpolerar f i vissa punkter. I det sista fallet kräver vi detta, i de övriga fallen "råkar" det bli så. Approximationen ligger knappast helt på ena sidan om f eftersom felen kan minskas om approximationen skär f. Vi kan alltså notera att minstakvadrapolynomet p_1 svarar mot ett interpolationspolynom av grad ett. p_1 interpolerar dock inte f i något av t_j-värdena.

Varför använder man då minstakvadratanpassning?
Problemställningen är ofta en annan. Modellen och antalet parametrar är givna och man vill få en så säker bestämning av parametrarna som möjligt. I polynomapproximationen ovan ändrade vi antalet punkter och därmed ändrades även antalet parametrar (koefficienter i polynomet).

Skulle man t.ex. ha två parametrar och endast använda två mätpunkter så får man en mycket osäker bestämning av parametrarna.

Det finns en annan typ av approximation där vi använder polynom men inte kräver interpolation. Säg att vi vill approximera $\sin t$ (används av olika programspråk, Java, Fortran, C/C++ etc.). På en Sundator är approximationskoden skriven i C (och åtkomlig via www) och den kompilerade koden finns i `libm`-biblioteket.

Först gör man argumentreduktion, man reducerar t så att det ligger i ett litet intervall kring origo (sin är ju periodisk). Ju kortare intervall man har desto enklare blir det att approximera.

Man kan också utnyttja att sin är udda. Det visar sig (se min FAQ på www) att det räcker att approximera $\sin t, t \in [0, \pi/4]$.

På detta intervall använder man ett polynom, av grad 13, och med enbart udda potenser. Koefficienterna, x_k, är valda så att

$$\max_{0\le t\le \pi/4} \left| \frac{\sin t}{t} - \left(1 + x_1 t^2 + x_2 t^4 + x_3 t^6 + x_4 t^8 + x_5 t^{10} + x_6 t^{12}\right) \right|$$

minimeras, ett sk minimax-problem. Man vill alltså minimera den maximala relativa avvikelsen.

Till skillnad från Taylorutveckling försöker man sprida ut felet över hela intervallet. Taylorutvecklingar har ett litet fel i den punkt där man gör utvecklingen. För att få ett litet fel över hela intervallet måste man då ta med onödigt många termer.

Några sätt att bestämma interpolationspolynomet

Det står polynomet i bestämd form, detta pga att det alltid existerar och är entydigt.

Interpolationsproblemet: hitta ett polynom p med grad högst $n-1$ sådant att $p(t_k) = y_k, k = 1, \ldots, n$, där alla (t_k, y_k) är givna och $t_1 < t_2 < \cdots < t_n$.

Låt oss anta att existensen är given och studera entydigheten. Antag att det finns ett annat polynom, q, av grad $\le n-1$ som interpolerar data. Det gäller då att $p(t_k) - q(t_k) = 0, k = 1, \ldots, n$, vilket säger att polynomet $p-q$ av grad $\le n-1$ har n distinkta nollställen. $p-q$ måste därför vara nollpolynomet och $p = q$.

Polynomet behöver inte alltid ha grad $n-1$. Om vi t.ex. väljer punkterna $y_k = t_k^2, k = 1, \ldots, 10$ så klarar vi oss med $p(t) = t^2$ fastän $n = 10$.

Nu till existensen. Den går att bevisa på flera olika sätt. Vi kommer att använda ett konstruktivt bevis. Antag att vi har $n = 3$ punkter. Här följer interpolationspolynomet på Lagranges form:

$$p(t) = y_1 \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} + y_2 \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} + y_3 \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)}$$

En fördel med denna form på polynomet är att det är lätt att ställa upp och den kan vara användbar vid teoretiskt arbete. Formen lämpar sig dock inte så väl för numeriska beräkningar (många operationer). Det finns också risk för under- overflow om man inte tänker sig för.

Ett annat sätt att konstruera polynomet är att sätta upp ett ekvationssystem som vid gjorde i det linjära fallet. Så vi ansätter $p(t) = x_1 + x_2 t + x_3 t^2$. Interpolationsvillkoren ger då problemet:

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

I linjäralgebrakursen brukar man visa att en sådan matris, en Vandermonde-matris, är ickesingulär om alla t-värdena är distinkta.

Detta system är lätt att formulera men relativt dyrt att lösa (kubisk kostnad) men det finns snabbare metoder som utnyttjar matrisens speciella utseende. Det är dock billigt och stabilt att beräkna $p(t)$. Man använder normalt Horners metod för detta.

Exempel med $n = 4$:

$$x_1 + x_2 t + x_3 t^2 + x_4 t^3 = x_1 + t(x_2 + t(x_3 + t x_4))$$

Detta skrivs lämpligen i en loop, men har jag använt sekventiell kod: $p = x_4$, $p = x_3 + tp$, $p = x_2 + tp$, $p = x_1 + tp$. Detta kräver $n - 1 +$ respektive *.

Man kan se Vandermonde-härledningen som ett speciallfall av följande. Vi ansätter

$$p(t) = x_1\phi_1(t) + x_2\phi_2(t) + \cdots + x_{n-1}\phi_{n-1}(t) + x_n\phi_n(t)$$

ϕ_k kallas basfunktion och i Vandermonde-matrisen använde vi $\phi_k(t) = t^{k-1}$.

Ett problem med Vandermondematriser är att de kan bli illakonditionerade.

Exempel: Antag att $n = 4$ och tag t-värdena $0.1, 0.2, 0.3, 0.4$. Matrisen kan då skrivas:

$$\begin{bmatrix} 1 & 10^{-1} & 10^{-2} & 10^{-3} \\ 1 & 2\cdot 10^{-1} & 4\cdot 10^{-2} & 8\cdot 10^{-3} \\ 1 & 3\cdot 10^{-1} & 9\cdot 10^{-2} & 27\cdot 10^{-3} \\ 1 & 4\cdot 10^{-1} & 16\cdot 10^{-2} & 64\cdot 10^{-3} \end{bmatrix}$$

Konditionstalet är $\approx 2\cdot 10^3$. Anledningen till det stora konditionstalet är att basfunktionerna liknar varandra (kolonnerna blir nästan linjärt beroende).

Ett sätt att få ner konditionstalet är att använda andra basfunktioner. Låt oss ta bokens förslag.

$$\phi_k(t) = \left(\frac{t-(t_1+t_n)/2}{(t_n-t_1)/2}\right)^{k-1}$$

Den transformerade variabeln ligger i intervallet $[-1,1]$:

$$-1 \le \frac{t-(t_1+t_n)/2}{(t_n-t_1)/2} \le 1, \quad t \in [t_1, t_n]$$

Denna transformation leder till det nya konditionstalet ≈ 8 i vårt exempel.

Det finns ytterligare en vanlig framställning av interpolationspolynomet, nämligen Newtons form. Den är en kompromiss mellan de två tidigare. Det är relativt billigt både att konstruera polynomet och att sedan evaluera det. Dessutom är möjligt att lägga till nya punkter utan att börja om med polynomberäkningen.

Den allmänna formen är;

$p(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + \cdots + x_n(t-t_1)(t-t_2)\cdots(t-t_{n-1})$

Låt oss se på specialfallet när $n = 3$.

$$p(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2)$$

Vi får det undertriangulära ekvationssystemet:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & t_2-t_1 & 0 \\ 1 & t_3-t_1 & (t_3-t_1)(t_3-t_2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

som ju är enkelt att lösa (framåtsubstitution). Vi ser också att det går att lägga till en punkt (en rad underst i matrisen) och vi behöver inte lösa systemet från början.

Exempel: Finn p som interpolerar $(1,1)$, $(2,4)$ samt $(3,11)$.

1) Vandermondes form. Vi ansätter $p(t) = x_1 + x_2 t + x_3 t^2$.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 11 \end{bmatrix}$$

som har lösning $x = [2, -3, 2]^T$ varför $p(t) = 2 - 3t + 2t^2$ eller $p(t) = 2t^2 - 3t + 2$.

2) Lagranges form:

$$p(t) = 1\frac{(t-2)(t-3)}{(1-2)(1-3)} + 4\frac{(t-1)(t-3)}{(2-1)(2-3)} + 11\frac{(t-1)(t-2)}{(3-1)(3-2)}$$

Förenklar vi detta uttryck får vi (givetvis) $p(t) = 2t^2 - 3t + 2$.

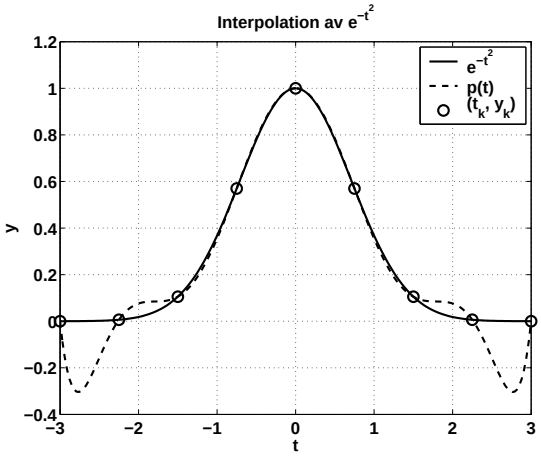
3) Newtons form: $p(t) = x_1 + x_2(t-1) + x_3(t-1)(t-2)$. Lös:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 2-1 & 0 \\ 1 & 3-1 & (3-1)(3-2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 11 \end{bmatrix}$$

så att $x = [1, 3, 2]^T$ varför $p(t) = 1 + 3(t-1) + 2(t-1)(t-2)$ som också kan förenklas till $p(t) = 2t^2 - 3t + 2$.

Problem med interpolation

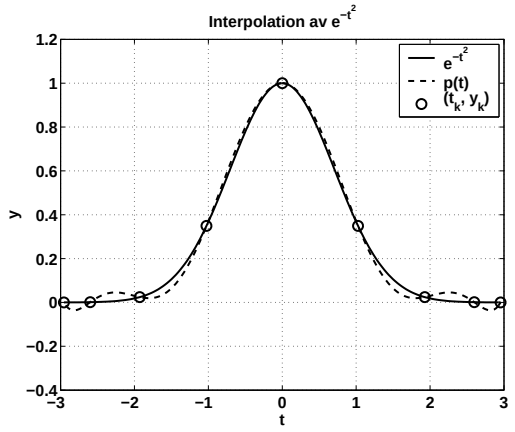
I följande exempel interpolerar $p(t)$, $f(t) = e^{-t^2}$ i nio punkter.



Det stämmer inte bra och det är stora fel i intervallets ändar. För vissa funktioner accentueras detta fenomen (Runges fenomen) när vi ökar antalet punkter (p behöver inte alltid konvergera mot f utan felet kan öka med ökande antal punkter).

Det är inte ovanligt att polynom av högt gradtal svänger kraftigt när man använder ekvidistant interpolation (samma avstånd mellan t_k-värdena).

Vi kan försöka att "hålla nere" polynomet i ändarna genom att lägga punkterna tätare där. Här har jag också använt nio punkter, men de ligger tätare mot intervallets ändpunkter. Polynomet svänger avsevärt mindre.



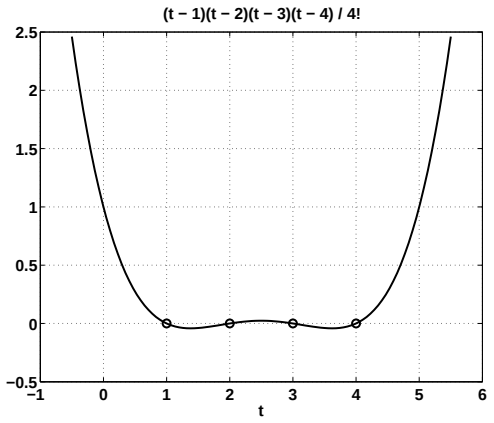
Vad är ett bra sätt att välja punkterna (om vi får välja)? Låt oss studera felets utseende igen (vi kan tänka oss exakta data, så att $p_f = p$):

$$f(t) - p(t) = \frac{f^{(n)}(\theta)}{n!}(t-t_1)(t-t_2)\cdots(t-t_n)$$

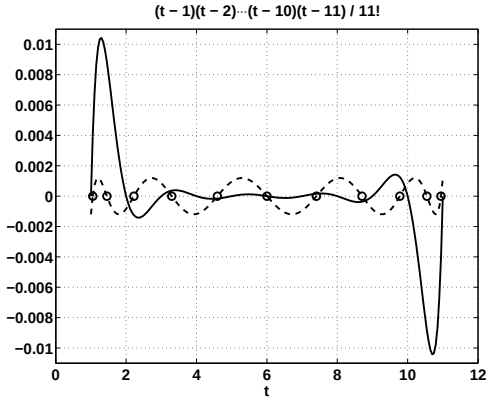
där $\theta \in (t, t_1, t_2, \ldots, t_n)$. Antag att $|f^{(n)}(\theta)| \le M$ för alla $\theta \in (t, t_1, t_2, \ldots, t_n)$. Vi har då

$$|f(t) - p(t)| \le \frac{M}{n!}\,|(t-t_1)(t-t_2)\cdots(t-t_n)|$$

Låt oss specialstudera funktionen $(t-t_1)(t-t_2)\cdots(t-t_n)/n!$. Den växer snabbt utanför $[t_1, t_n]$. I bilden på nästa sida är $n = 4$ och sedan 11. Extrapolation är farligt.



Den kan vara orolig inom intervallet också:



Den heldragna kurvan, i andra bilden på föregående sida, svarar mot ekvidistanta punkter den streckade (bättre) utnyttjar Chebyshevpunkterna. Dessa punkter har egenskapen att göra det maximala värdet av $|(t-t_1)(t-t_2)\cdots(t-t_n)|$ så litet som möjligt.

Sats:

$$\max_{-1\le t\le 1} |(t-t_1)(t-t_2)\cdots(t-t_n)|$$

minimeras då

$$t_k = -\cos\left[\frac{(2k-1)\pi}{2n}\right],\ k=1,2,\ldots,n$$

Det maximala värdet på $|(t-t_1)(t-t_2)\cdots(t-t_n)|$ är då $1/2^{n-1}$.

När t ligger i ett annat intervall, $[\alpha,\beta]$ säg får vi göra en linjär avbildning av Chebyshevpunkterna till detta intervall. Vi ser att

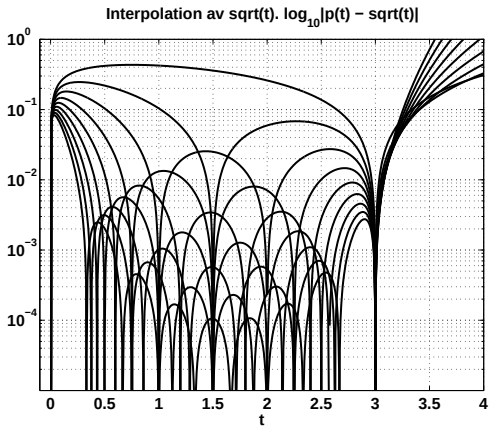
$$\frac{\beta-\alpha}{2}[-1,\ 1]+\frac{\alpha+\beta}{2}=[\alpha,\ \beta]$$

så de transformerade Chebyshevpunkterna blir

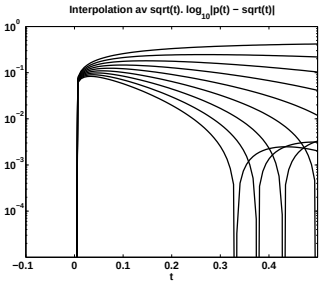
$$-\frac{\beta-\alpha}{2}\cos\left[\frac{(2k-1)\pi}{2n}\right]+\frac{\alpha+\beta}{2}$$

I bland är det ändå problem. Det kan tänka sig att M, begränsningen av $|f^{(n)}(\theta)|$ ej existerar. Exempel: $f(t)=\sqrt{t}$ på intervallet $[0,3]$. Redan $f'(0)$ är ju obegränsad, man säger att derivatan har en singularitet. I vissa fall visar sig singulariteten först i högre derivator (ex $f(t)=t^{5/2}$).

På nästa sida visas felet vid interpolation av $\sqrt{t}, t\in[0,3]$ för ökande n. Chebyshevpunkter ger obetydligt bättre resultat.



Här inzoomat:



Anledningen till att det inte konvergerar vid $t=0$ är att $\sqrt{t}$ där har lodrät tangent, något ett polynom aldrig kan ha.

Om en funktion har $n+1$ antal kontinuerliga derivator så kan den utvecklas i en Taylorutveckling:

$$f(t)=f(a)+\frac{f'(a)}{1!}(t-a)+\frac{f''(a)}{2!}(t-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(t-a)^n+R(t)$$

där resttermen $R(t)=c(\xi)(t-a)^{n+1}, \xi\in(a,t)$ och $|c(\xi)|$ är uppåt begränsad. Detta innebär att en sådan funktion (som har Taylorutveckling) liknar ett polynom på ett tillräckligt litet intervall.

Om inte alla $f^{(k)}(a)=0, k=0,\ldots,n$ kan vi göra $R(t)$ godtyckligt liten jämfört med resten av Taylorutvecklingen, genom att ta $|t-a|$ tillräckligt litet. På ett stort intervall behöver inte funktionen likna ett polynom.

$\sqrt{t}$ har ingen Taylorutveckling kring $a=0$. Däremot har ju $\sqrt{t}$ en utveckling kring alla $a>0$ och det är inga problem att approximera funktionen för positiva t.

Man kan naturligtvis approximera med annat än polynom. Exempel: Approximera $f(t)=(\sin t-1)/(\cos t-1)$ kring $t=0$. Problem, i detta fall har ju f (inte bara derivatorna) en singularitet. Kan använda rationell approximation (Padé).

$$\frac{\sin t-1}{\cos t-1}=\frac{12-12t+t^2+t^3}{6t^2}+R(t),\quad R(t)=\frac{t^2}{120}+\cdots$$

Så för $t\approx 0$ och med $r(t)=(12-12t+t^2+t^3)/(6t^2)$:

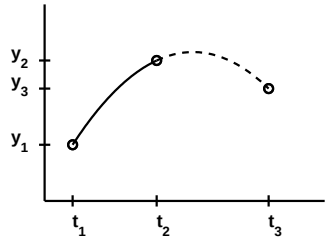
$$\left|\frac{f(t)-r(t)}{f(t)}\right|=\left|\frac{R(t)}{f(t)}\right|\approx\frac{t^4}{240}$$

Ett annat alternativ är att använda en generaliserad potensserie:

$$f(t)=\frac{2}{t^2}-\frac{2}{t}+\frac{1}{6}+\frac{t}{6}+\frac{t^2}{120}+\frac{t^3}{360}+\frac{t^4}{3024}+\cdots$$

Splinefunktioner

Polynom av höga gradtal är svårhanterliga men har samtidigt lokalt goda approximationsegenskaper. Dessutom är ju polynom "trevliga" funktioner. Enkla att beskriva, lagra, beräkna, integrera, derivera etc. Så en vanlig kompromiss är styckvisa polynom av låga gradtal. Man behåller polynomens enkelhet men slipper svängningarna.



I bilden ovan är den heldragna linjen ett polynom och den streckade ett annat. Heldragen plus streckad kurva tillsammans utgör dock inte (nödvändigtvis) ett polynom.

Def: En interpolerande splinefunktion av grad j är en funktion som interpolerar $(t_k, y_k), k = 1, \ldots, n$ och som består av styckvisa polynom, på intervallen $[t_1, t_2], [t_2, t_3], \ldots$
Dessutom är splinefunktionen $j-1$ gånger kontinuerligt deriverbar i knutpunkterna (dvs. i (t_k, y_k)).

Det är inga problem med kontinuiteten av derivatorna av varje enskilt polynom (i varje delintervall).

Om $j = 1$ så har vi ingen kontinuerlig derivata utan bara kontinuitet hos splinefunktionen.
Delpolynomen har högst grad ett.

Om $j = 2$ så är delpolynomen (högst) andragradspolynom. Splinefunktionen är kontinuerlig och är kontinuerligt deriverbar (förstaderivatan är kontinuerlig).

Det vanligaste är dock $j = 3$, kubiska splines, där delpolynomen är kubiska (högst) och splinefunktionen är kontinuerlig liksom dess första- och andraderivator.

Låt oss se varför detta verkar vara möjligt att åstadkomma och varför man inte kan kräva kontinuerlig tredjederivata.

En kubisk spline kan skrivas $p_k(t) = a_k t^3 + b_k t^2 + c_k t + d_k$ på intervallet $[t_k, t_{k+1}]$. Antag att vi har n stycken t-värden. Detta ger $n-1$ intervall (lika många polynom), så antalet obestämda koefficienter är $4(n-1)$. Hur många villkor har vi?

Interpolationskravet ger $2(n-1)$ villkor (ty varje polynom måste interpolera 2 knutpunkter). Detta ger oss kontinuiteten gratis.

Kontinuerlig förstaderivata ger $n-2$ villkor (inre punkter) och lika många för andraderivatan. Så summa $2(n-1)+n-2+n-2 = 4n-6$ villkor.

Det innebär att vi saknar två villkor som måste bestämmas på något sätt. Här är några vanliga tilläggsvillkor
(s är splinefunktionen):

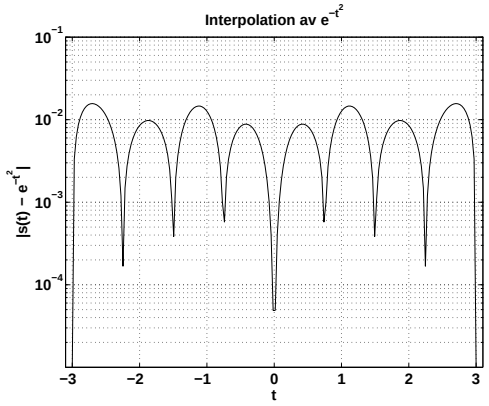
$s''(t_1) = s''(t_n) = 0$ sk naturliga splines (minimerar $\int_{t_1}^{t_n} (s''(t))^2\, dt$)

$s'(t_1) = f'(t_1)$ och $s'(t_n) = f'(t_n)$ komplett spline

$s'(t_1) = s'(t_n)$ samt $s''(t_1) = s''(t_n)$ periodisk första- och andraderivata (kanske rimligt med $y_1 = y_n$ i detta fall).

$p_1(t) = p_2(t), t \in [t_1, t_3]$ och $p_{n-2}(t) = p_{n-1}(t), t \in [t_{n-2}, t_n]$, not-a-knot; medför att s''' kontinuerlig i $t = t_2$ och $t = t_{n-1}$). Det är alltså ett tredjegradspolynom i $[t_1, t_3]$ (och ett (annat) i $[t_{n-2}, t_n]$).

Om vi återvänder till e^{-t^2}-exemplet har vi inget problem att göra en bra approximation med kubiska splines. Jag har ritat felet snarare än de två kurvorna, eftersom de ligger så nära varandra.



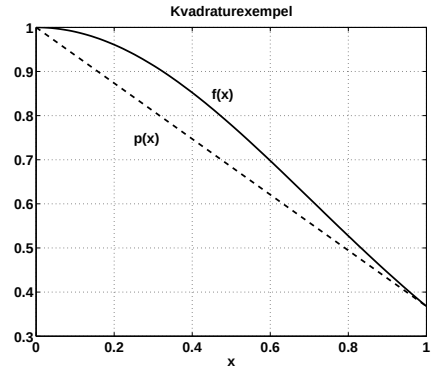
Om $f^{(4)}$ är begränsad (över intervallet) så konvergerar splinefunktionen mot f med hastigheten $\max_k h_k^4$.

Kvadratur - numerisk integration

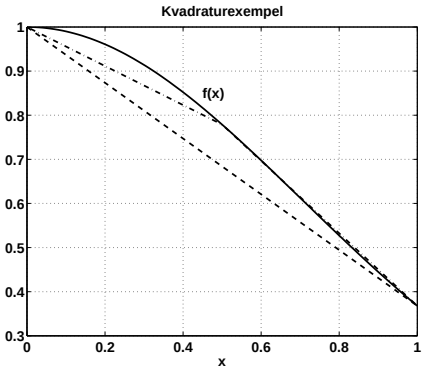
Vill beräkna: $\int_a^b f(x)\, dx$. Inte alltid möjligt att uttrycka en primitiv funktion i elementära funktioner (inte alltid bekvämt heller).

Grundidé: approximera $f(x)$ med en funktion $p(x)$ som har bra approximationsegenskaper, och som är enkel att beräkna och integrera.
Enkelt exempel: vi vill approximera $\int_0^1 e^{-x^2}\, dx$.
Facit: $\int_0^1 e^{-x^2}\, dx \approx 0.74682413281$.

Approximera $f(x)$ med en linjär funktion, $p(x) = 1 + (e^{-2} - 1)x$.



Från bilden framgår att vår approximation måste vara rätt dålig, 0.68394. Låt oss dela upp integrationsintervallet i två delintervall, $[0, 0.5], [0.5, 1]$ och approximera med en linjär funktion på varje delintervall:



Den andra halvan borde stämma rätt bra, absoluta felet är ≈ 0.001. Det är fortfarande rätt stort fel i det vänstra intervallet. Approximationen av integralen är nu: 0.73137. Vi kan fortsätta med att halvera intervallen, men det verkar lite bortkastat att fortsätta med högra halvan. Vi vill ha en adaptiv metod som försöker anpassa sig till felet.

Från bilden ser man att approximationen kommer att konvergera mot det exakta värdet (om vi bortser från avrundningsfel).

Ett annat alternativ är att approximera med ett polynom av högre gradtal. Om vi integrerar interpolationspolynomet, av grad fyra, som interpolerar e^{-x^2} för $x = 0, 0.25, 0.5, 0.72, 1$ blir felet i integralen $\approx 10^{-5}$.

Mer om Trapetsmetoden

Trapetsmetoden: approximation av f med ett linjärt interpolationspolynom på varje delintervall. På intervallet $[a, b]$ approximerar vi integralen med arean av en parallelltrapets (därav namnet):

$$\int_a^b f(x)\ dx \approx \frac{h}{2}(f(a) + f(b)), \quad h = b - a$$

Vi delar nu in $[a, b]$ i $n-1$ lika långa delintervall (en del författare börjar med x_0):

$$x_k = a + (k-1)h, \quad k = 1, \ldots, n, \quad h = (b-a)/(n-1)$$

så att $x_1 = a$ och $x_n = b$.

Beteckna den approximation vi får med $T_n(f)$. Den blir:

$$\frac{h}{2}\left[(f(x_1) + f(x_2)) + (f(x_2) + f(x_3)) + \ldots + (f(x_{n-1}) + f(x_n))\right] =$$

$$h\left[\frac{f(x_1)}{2} + f(x_2) + f(x_3) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

Gör man ovanstående i vårt exempel verkar felet ha utseendet ch^2. Kan man bevisa det?

Från interpolationsteorin vet vi att:

$$f(x) - p(x) = \frac{f''(\theta_x)}{2}(x-a)(x-b), \quad \theta_x \in (a, b)$$

med ett intervall. Alltså

$$\int_a^b f(x)\ dx - \int_a^b p(x)\ dx = \int_a^b \frac{f''(\theta_x)}{2}(x-a)(x-b)\ dx =$$

$$\frac{f''(\xi)}{2}\int_a^b (x-a)(x-b)\ dx = -\frac{(b-a)^3\ f''(\xi)}{12}, \quad \xi \in (a, b)$$

Detta följer av integralkalkylens medelvärdessats ($(x-a)(x-b)$ byter inte tecken på $[a, b]$). I det allmänna fallet, med $n-1$, intervall får vi summera felen:

$$\int_a^b f(x)\ dx - T_n(f) = -\sum_{k=1}^{n-1} \frac{(x_{k+1} - x_k)^3\ f''(\xi_k)}{12} = -\frac{h^3}{12}\sum_{k=1}^{n-1} f''(\xi_k)$$

Om vi antar att f'' är kontinuerlig så antar f'' min/max på $[a, b]$ så att

$$\min_{a \le x \le b} f''(x) \le \frac{1}{n-1}\sum_{k=1}^{n-1} f''(\xi_k) \le \max_{a \le x \le b} f''(x)$$

så att (en kontinuerlig funktion antar mellanliggande värden):

$$\frac{1}{n-1}\sum_{k=1}^{n-1} f''(\xi_k) = f''(\xi)$$

Alltså:

$$\int_a^b f(x)\ dx - T_n(f) = -\frac{h^3(n-1)f''(\xi)}{12} = -\frac{(b-a)h^2 f''(\xi)}{12}, \ \xi \in [a, b]$$

ty $h(n-1) = b - a$.

Så om andraderivatan är begränsad i $[a, b]$ och om vi räknar exakt gäller att $T_n(f) \to \int_a^b f(x)\ dx, \ n \to \infty$.

Observera att om man inte vet något om hur f'' ser ut kan man inte garantera konvergens.

Det är enkelt att lura avbrottskriteriet i kvadraturprogram. Det enda vi känner är ju $(x_k, f(x_k)), k = 1, \ldots, n$ men det finns oändligt många funktioner som interpolerar dessa punkter (med olika värden på integralen).
Detta är ett allmänt beräkningsproblem (ändliga punktmängder från oändliga punktmängder).

Newton-Cotes-kvadratur

Man kan generalisera trapetsmetoden. Att integrera interpolationspolynom ger Newton-Cotes metoder. Man skiljer mellan öppna metoder där ändpunkterna ej är med resp. slutna, där ändpunkterna tas med.
Enklaste metoden är mittpunktsmetoden (rektangelmetoden) där vi approximerar $f(x)$ med $f((x_k + x_{k+1})/2)$ i intervallet $[x_k, x_{k+1}]$. Så om vi bara ser på intervallet $[a, b]$ så har vi approximationen:

$$\int_a^b f(x)\ dx \approx (b-a)f\left(\frac{a+b}{2}\right)$$

Vi har tittat på trapetsmetoden där man använder en linjär approximation. Använder man en kvadratisk approximation får man Simpsons formel:

$$\int_a^b f(x)\ dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Om man härleder felen för de sammansatta metoderna (mer än ett intervall) har mittpunktsmetoden felet

$$(b-a)h^2 f''(\xi)/24$$

vilket lustigt nog är mindre än för trapetsmetoden som ju har högre ordning på polynomet.

Dessutom har både mittpunkts- och trapetsmetod polynomiellt gradtal ett (exakt för alla polynom upp till och med grad ett). Detta beror på att vi inte primärt är intresserade av att approximera f (då är normalt en allmän linjär funktion bättre än en konstant) utan att vi vill approximera en integral.

En linjär approximation av t.ex. $f(x) = x$ över $[-1, 1]$ ger felet noll och en exakt integral. Approximationen av samma funktion med $f(0) = 0$ ger stora fel i funktionsanpassningen men en exakt integral pga att approximationsfelen i integralen precis tar ut varandra.

Simpsons formel, som också har ett udda antal punkter (jämn grad på polynomet), har felet $(b-a)h^4 f^{(4)}(\xi)/180$ som också uppvisar mindre fel än först förväntat (tre punkter ger h^4 och $f^{(4)}$).

Allmänt kan en kvadraturmetod skrivas

$$\int_a^b f(x)\ dx \approx \sum_{k=1}^{n} w_k f(x_k)$$

w_k kallas vikter och x_k abscissor.

Hur ser Simpsons formel ut på mer än ett intervall? Dela in $[a,b]$ i sex lika långa delintervall där vi använder metoden på: $[x_1,x_3]$, $[x_3,x_5]$ och $[x_5,x_7]$.

$$\int_{x_1}^{x_3} f(x)\ dx + \int_{x_3}^{x_5} f(x)\ dx + \int_{x_5}^{x_7} f(x)\ dx \approx$$
$$\frac{x_3-x_1}{6}\left[f(x_1)+4f\left(\frac{x_1+x_3}{2}\right)+f(x_3)\right] +$$
$$\frac{x_5-x_3}{6}\left[f(x_3)+4f\left(\frac{x_3+x_5}{2}\right)+f(x_5)\right] +$$
$$\frac{x_7-x_5}{6}\left[f(x_5)+4f\left(\frac{x_5+x_7}{2}\right)+f(x_7)\right]$$

$\frac{x_1+x_3}{2} = x_2$ etc. och $h = x_{k+1} - x_k$ så att approximationen blir:

$$\frac{2h}{6}[f(x_1)+4f(x_2)+2f(x_3)+4f(x_4)+2f(x_5)+4f(x_6)+f(x_7)]$$

eftersom ändpunkterna i delintervallen sammanfaller parvis.

Testar man detta på $f(x)=e^{-x^2}$ och kräver ett absolut fel $\le 1.2\cdot 10^{-9}$ tar trapetsmetoden 7150 funktionsevalueringar, mittpunktsmetoden 5055 och Simpsons formel 52. Matlabs `quadl`, som är adaptiv, tar 18.

Det spelar alltså stor roll vilken metod man använder och h^m-faktorn är mycket viktig. För att exemplifiera, låt oss anta att vi har en uppsättning metoder med feltermer:

$$c\ (b-a)\ h^m\ f^{(m)}(\xi)$$

där c är en konstant och m ett postivt heltal som varierar mellan metoderna. $h = 1/(n-1)$ som vanligt. Om $f^{(m)}(\xi)$ är konstant (inte sannolikt) kan felet skrivas Ch^m för en annan konstant C. För att feltermen skall bli $\approx \tau$, en given tolerans, krävs alltså:

$$Ch^m \approx \tau,\quad n \approx \frac{1}{(\tau/C)^{1/m}},\quad n \propto \frac{1}{\tau^{1/m}}$$

Med $\tau = 10^{-9}$ och $C=1$ så får vi denna tabell:

m	$\propto n$
2	31623
3	1000
4	178
5	63
6	32
7	19

Om någon av f:s lägre derivator har en singularitet i $[a,b]$ kan dock metoderna konvergera avsevärt långsammare.

Exempel:
Trapetsmetoden på $f(x)=x^p$, $0<p<1$, $[a,b]=[0,1]$.

Vi kan ej använda feluppskattningen på hela intervallet eftersom f' och f'' har en singularitet i nollan. Vi kan dock räkna ut skillnaden mellan integral och approximation för $x\in[0,h]$:

$$\int_0^h x^p\ dx - \frac{h\ [0^p+h^p]}{2} = \frac{(1-p)}{2(1+p)}h^{1+p}$$

Man skulle kunna använda feluppskattningen på $[h,1]$ för att visa konvergens (felet går mot noll när $h\to 0$), men det blir ett väldigt svagt resultat.

Använder man uppskattningen på $[h,2h]$, $[2h,3h]$ etc. får man ett bra resultat som visar att felet uppför sig som h^{1+p}.

Det förväntar man sig även för de övriga metoderna. Antag att feltermen över det första intervallet (som innehåller nollan) har utseendet $ch^{m+1}f^{(m)}(\xi)$ där c är en konstant och $\xi > 0$ är en multipel av h, $\xi = \mu h$ säg (†). Med vår funktion så blir

$$ch^{m+1}f^{(m)}(\xi) = c_1(\mu)h^{m+1}h^{p-m} = c_1(\mu)h^{1+p}$$

för någon annan konstant c_1 som beror av μ. Denna konstant är givetvis viktig, så detta resonemang visar bara hur vi förväntar oss att beroendet av h ändras. Vi får alltså bara h^{1+p} som kan kräva många funktionsberäkningar (enligt vår tabell).

Tar vi $p=0.3$ med samma tolerans som i föregående exempel, så kräver Simpson inte 52 funktionsberäkningar utan 1 697 157.

Problemet är väsentligen av samma slag som när vi interpolerade $\sqrt{t}$ kring $t\ge 0$.

Vad kan man göra? I enkla fall kan man kanske byta parametrisering av f och betrakta x som funktion av y (givetvis förutsatt att f^{-1} existerar lokalt) och sedan integrera i y-led (lite mer fixande krävs för att få rätt integral).

$y=\sqrt{x}$ övergår då i triviala $x=y^2$. Man kan skaffa sig ett interpolationspolynom genom att anpassa x-värden till y-värden (sk invers interpolation).

(†) På varje intervall $[\delta,h]$, $\delta>0$ gäller feluppskattningen. Under svaga villkor på f och metod kan skillnaden mellan integralen över $[0,\delta]$ och metoden begränsas av konstant $\cdot\,\delta$, vilket gör att feltermen bestämmer utseendet på felet.

Adaptivitet

Normalt vill vi inte ha ekvidistanta punkter, utan vi vill att metoden automatiskt ska anpassa sig efter funktionens utseende och använda tätare med punkter där så behövs. Vi behöver då en uppskattning av felet.

Att direkt uppskatta feltermen gör man normalt inte.
En vanlig metod är att räkna ut resultatet med två metoder (en med mindre fel) och jämföra resultaten. Kostnaden bör vara som för en metod. Man kan också använda samma metod men med olika antal punkter.

I boken används den senare varianten med trapetsmetoden (Simpson, eller bättre, är vanligare). Här följer en genomgång. Vi börjar med intervallet $[a,b]$, räknar ut trapetsapproximationen med två punkter. Vi lägger sedan till mittpunkten, $m=(a+b)/2$ och räknar ut en ny approximation, nu med tre punkter. Observera att detta kräver ett nytt funktionsvärde, $f(m)$.

Vi fortsätter nu så rekursivt på intervallen $[a,m]$ och $[m,b]$. När felet över ett intervall är tillräckligt halverar vi inte detta intervall vidare.

Antag att vi har kommit ner till ett delintervall av längd h. Approximationerna kan skrivas (I är det exakta värdet av integralen över detta delintervall)

$$I = T_h - h^3 f''(\xi)/12 \quad\text{resp.}\quad I = T_{h/2} - h(h/2)^2 f''(\theta)/12$$

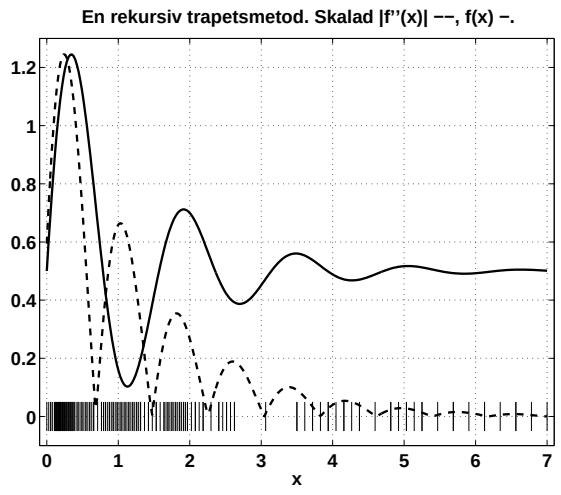
Antag att $c = -f''(\xi) \approx -f''(\theta)$ (behöver inte vara sant).
Då gäller

$$0 \approx T_h - T_{h/2} + ch^3(1-1/4)/12 = T_h - T_{h/2} + 3ch^3/(4\cdot 12)$$

Men felet i $T_{h/2}$ är ju $ch(h/2)^2/12$. Alltså

$$I \approx T_{h/2} + \frac{T_{h/2}-T_h}{3}$$

Så här kan det se ut (med rätt stor tolerans):



Man kan notera att formeln ovan även ger upphov till en ny metod. Om vi lägger till feluppskattningen får vi

$$I \approx (4T_{h/2} - T_h)/3$$

och bakom denna formel döljer sig Simpsons formel.

Man kan betrakta härledningen vi har gjort som ett specialfall av sk Richardsonextrapolation.

Man kan visa att det existerar en serieutveckling av felet

$$\left(\int_a^b f(x)\ dx =\right) I = T_h + a_1h^2 + a_2h^4 + a_3h^6 + \cdots$$

Vi halverar nu h och får

$$I = T_{h/2} + a_1h^2/4 + a_2h^4/16 + a_3h^6/64 + \cdots$$

Genom att kombinera de två uttrycken kan vi bli av med h^2-termen (och därmed minska felet):

$$4I - I = 4T_{h/2} - T_h + (4a_1h^2/4 - a_1h^2) + (4a_2h^4/16 - a_2h^4) + \cdots$$

så att

$$I = \frac{4T_{h/2} - T_h}{3} - \frac{a_2h^4}{4} + \cdots$$

Detta kan man nu upprepa (med $T_{h/4}$) för att bli av med h^4-termen. Denna process (upprepad Richardsonextrapolation) kallas Rombergkvadratur.

Richardsonextrapolation kan användas närhelst man har en utveckling av felet. Exempel, approximation av derivata.

$$\frac{f(x+h) - f(x)}{h} = \frac{1}{h}\sum_{k=0}^{\infty}\frac{h^k f^{(k)}(x)}{k!} - \frac{f(x)}{h} = f'(x) + \sum_{k=2}^{\infty}\frac{h^{k-1}f^{(k)}(x)}{k!}$$

så att

$$f'(x) = \frac{f(x+h) - f(x)}{h} - hf''(x)/2 - h^2f'''(x)/6 - \cdots$$

$$f'(x) = \frac{f(x+h/2) - f(x)}{h/2} - (h/2)f''(x)/2 - (h/2)^2f'''(x)/6 - \cdots$$

och

$$f'(x) = 2\frac{f(x+h/2) - f(x)}{h/2} - \frac{f(x+h) - f(x)}{h} + h^2f'''(x)/12 + \cdots$$

$$f'(x) = \frac{-3f(x) + 4f(x+h/2) - f(x+h)}{h} + h^2f'''(x)/12 + \cdots$$

Gausskvadratur

Antag att vi vill beräkna $\int_a^b f(x)\ dx$ och tillåts göra tre funktionsberäkningar, $f(x_1)$, $f(x_2)$ samt $f(x_3)$. Om vi väljer $x_1 = a$, $x_2 = (a+b)/2$ samt $x_3 = b$ så kommer Simpsons formel att vara optimal när det gäller polynomiellt gradtal. Dvs om vi vill att metoden ska vara exakt för polynom av grad $0, 1, \ldots, m$ för så stort m som möjligt så är Simpsons metod det bästa valet ($m = 3$).

Det visar sig dock att vi kan få större m genom att välja andra x_k-värden. Detta är kärnan i Gausskvadratur, att välja både x_k-värden och vikter för att maximera m.

Låt oss ta intervallet $[-1, 1]$. Vi ska nu välja x_1, x_2, x_3 samt vikter w_1, w_2, w_3 så att

$$\int_{-1}^{1} x^k\ dx = w_1x_1^k + w_2x_2^k + w_3x_3^k, k = 0, 1, \ldots, m$$

för maximalt m. Integralens värde blir 0 om k är udda och $2/(k+1)$ annars. Vi får följande ickelinjära ekvationssystem att lösa:

$$\begin{cases} 2 &= w_1 + w_2 + w_3 & k = 0 \\ 0 &= w_1x_1 + w_2x_2 + w_3x_3 & k = 1 \\ 2/3 &= w_1x_1^2 + w_2x_2^2 + w_3x_3^2 & k = 2 \\ 0 &= w_1x_1^3 + w_2x_2^3 + w_3x_3^3 & k = 3 \\ 2/5 &= w_1x_1^4 + w_2x_2^4 + w_3x_3^4 & k = 4 \\ 0 &= w_1x_1^5 + w_2x_2^5 + w_3x_3^5 & k = 5 \end{cases}$$

Det verkar inte rimligt att ta med en ekvation till. Vi har ju $3+3 = 6$ obekanta och vi kan då kanske satisfiera sex ekvationer. För att lösa systemet kan man använda "brute force", men det verkar rimligt att punkterna måste uppvisa viss symmetri. Vi antar sålunda att $x_1 < x_2 < x_3$ med $x_2 = 0$ och $x_1 = -x_3$.

Detta val leder ($k = 1$) till att $w_1 = w_3$ och satisfiering av fallen $k = 3, 5$. Kvarstår då ekvationerna $2 = 2w_1 + w_2$, $2/3 = 2w_1x_1^2$ samt $2/5 = 2w_1x_1^4$. Vi får $x_1 = -\sqrt{3/5}$ och $w_1 = 5/9$. Metoden blir alltså:

$$\int_{-1}^{1} f(x)\ dx \approx \frac{5}{9}f\left(-\sqrt{3/5}\right) + \frac{8}{9}f\left(0\right) + \frac{5}{9}f\left(\sqrt{3/5}\right)$$

Man ser att metoden inte är exakt för $m = 6$ så det polynomiella gradtalet är 5 (det var 3 för Simpsons metod). Eftersom integration är en linjär operation så är metoden exakt för alla polynom av grad högst 5.

För en Gausskvadraturformel har vi gradtalet $2n - 1$ med n punkter. Vi har dock offrat i enkelhet. Härledningen kan dock förenklas (man använder teorin för ortogonala polynom och kan blanda in egenvärdesproblem för tridiagonala matriser).

En annan nackdel är att värdena måste skrivas in i ett program (stora tabeller). Det allvarligaste problemet är dock att man inte kan återanvända funktionsvärden när man gör adaptiva metoder.

Det finns dock varianter, Gauss-Kronrodkvadratur där man har en kompromiss mellan optimaliteten i Gausskvadratur och kravet på återanvändning av funktionsvärden, se boken.

Hur ser vår metod ut på intervallet $[a, b]$, $\int_a^b f(z)\ dz$? Sätt $z = \alpha x + \beta$ där $\alpha = (b-a)/2$ och $\beta = (a+b)/2$. $z \in [a, b] \to x \in [-1, 1]$. $dz = \alpha\ dx$. Alltså:

$$\int_a^b f(z)\ dz = \int_{-1}^{1} f(\alpha x + \beta)\alpha\ dx = \sum_{k=1}^{3}(\alpha w_k)f(\alpha x_k + \beta)$$

Ordinära differentialekvationer

Vi kommer enbart att studera begynnelsevärdesproblem, t.ex.

$$y'(t) = t^2 + \sin y(t), \quad 3 < t \leq 10, \quad y(3) = 4$$

Derivatan, $y'(t)$, är tagen med avseende på t ("tiden").

$3 < t \leq 10$ anger det intervall där vi vill beräkna lösningen (approximativt). $y(3) = 4$ är ett begynnelsevärde som anger y:s värde, 4, vid tiden $t = 3$. Normalt (i övningar och anteckningar) skriver vi aldrig ut t, i $y(t)$. Vi struntar även i intervallet (tiden i begynnelsevärdet är vänster ändpunkt, och Du får anta något lämpligt slutvärde). Problemet kan då formuleras:

$$y' = t^2 + \sin y, \quad y(3) = 4$$

Normalt vill vi studera ett generellt problem, vi skriver:

$$y' = f(t, y), \quad y(t_0) = y_0$$

Så, i exemplet ovan är $f(t, y) = t^2 + \sin y$. Begynnelsetiden är t_0 (3 i exemplet) och y vid detta värde är y_0 (4 i exemplet). Både t_0 och y_0 måste vara kända.

Lösningsmetoderna genererar approximationer till lösningen för en uppsättning tidpunkter: (t_0, y_0), (t_1, y_1), $(t_2, y_2), \ldots, (t_n, y_n)$, där t_n är slut-tiden och $y_k \approx y(t_k)$.

> y_k är en approximation av lösningen vid tiden $t = t_k$. Det exakta värdet är $y(t_k)$.

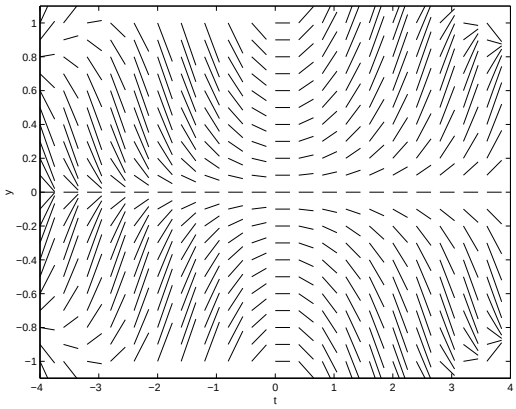
Senare kommer system av ekvationer. Sådana behövs för att vi skall kunna lösa problem som innehåller högre derivator, t.ex.

$$y''' = t + 2y'' + (y')^2 + \sin y, \quad y(0) = 2, \quad y'(0) = -3, \quad y''(0) = 4$$

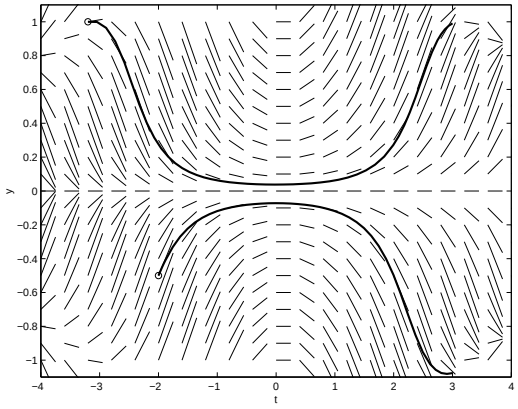
Exempel: låt oss studera problemet: $y' = 1$. Detta är inget begynnelsevärdesproblem (eftersom vi saknar $y(t_0) = y_0$). Ett problem av detta slag har normalt oändligt många lösningar, i detta fall $y(t) = t + c$ där c är ett godtycklig reellt tal. När vi ger ett begynnelsevärde väljer vi ut en av alla dessa oändligt många lösningar. $y(3) = 4$ ger oss lösningen $y(t) = t + 1$.

Med grafiska verktyg kan vi skaffa oss en bild om lösningsmängden även för problemet $y' = f(t, y)$. Låt oss göra detta för problemet $y' = \sin(ty)$.

I bilden nedan har jag skapat ett gitter i (en begränsad del av) (t, y)-planet. I varje gitterpunkt har jag avsatt en pil vars riktning överensstämmer med derivatan av den lösningskurva som går genom punkten. Detta är enkelt eftersom $y' = f(t, y)$, så derivatan är $f(t, y)$.

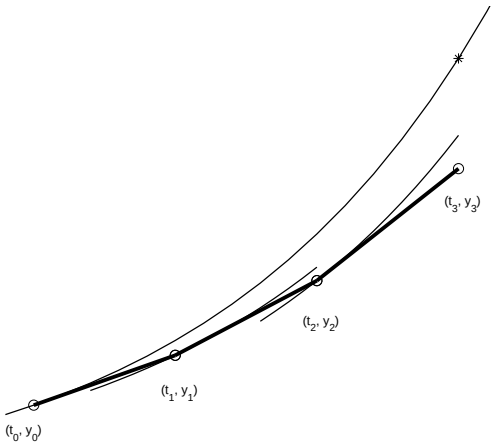


I nästa bild har jag ritat i två lösningskurvor (jag har gett två begynnelsevärden).



Det finns begynnelsevärdesproblem som saknar, eller har flera lösningar. Det kan också vara så att $y(t)$ inte existerar för alla $t > t_0$.

Föregående bild antyder en enkel lösningsmetod. Vi startar i (t_0, y_0) (som vi känner). Vi tar sedan ett litet steg utmed tangenten till lösningen (tangenten kan vi beräkna med hjälp av $f(t, y)$).



Antag att vi stegar med fix steglängd, h, i t så att:

$t_1 = t_0 + h$, $t_2 = t_1 + h$, $t_3 = t_2 + h, \ldots$. Allmänt $t_k = t_0 + kh$.

Vi får Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k), \quad k = 0, 1, 2, \ldots$$

eller utskrivet

$$y_1 = y_0 + hf(t_0, y_0),\ y_2 = y_1 + hf(t_1, y_1),\ y_3 = y_2 + hf(t_2, y_2), \ldots$$

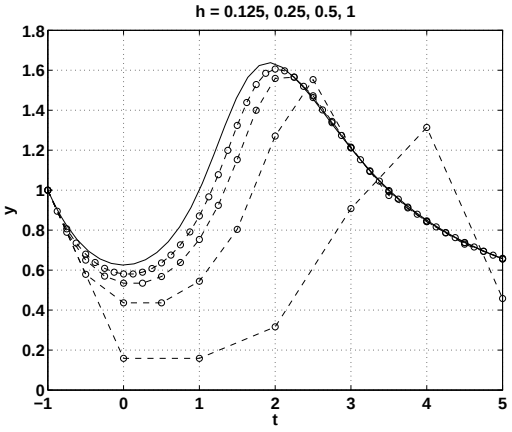
Exempel: $y' = \sin(ty), \quad y(-1) = 1.$
Så $t_0 = -1$, $y_0 = 1$ och $f(t, y) = \sin(ty)$.

Om $h = 0.1$ får vi approximationerna:

$y_1 = y_0 + hf(t_0, y_0) = y_0 + h\sin(t_0 y_0) =$
$1 + 0.1\sin(-1 \cdot 1) \approx 0.9159$

$y_2 = y_1 + hf(t_1, y_1) = y_1 + h\sin(t_1 y_1) \approx$
$0.9159 + 0.1\sin(-0.9 \cdot 0.9159) \approx 0.8425$

$y_3 = y_2 + hf(t_2, y_2) = y_2 + h\sin(t_2 y_2) \approx$
$0.8425 + 0.1\sin(-0.8 \cdot 0.8425) \approx 0.7801$ osv.



Alternativa härledningar av Eulers metod

Taylorutveckling:

$$y(t_k + h) = y(t_k) + h\ y'(t_k) + \frac{h^2}{2}\ y''(t_k) + \cdots$$

Nu är $y'(t_k) = f(t_k, y(t_k))$ och $t_{k+1} = t_k + h$ så att:

$$y(t_{k+1}) \approx y(t_k) + h\ f(t_k, y(t_k))$$

Vi approximerar nu $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ och får:

$$y_{k+1} = y_k + h\ f(t_k, y_k)$$

Nu till en härledning som använder kvadratur (integration).

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} y'(t)\ dt = \int_{t_k}^{t_{k+1}} f(t, y(t))\ dt$$

Vi approximerar nu integralen med arean av en rektangel;

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t))\ dt \approx \underbrace{(t_{k+1} - t_k)}_{h} f(t_k, y(t_k))$$

Så:

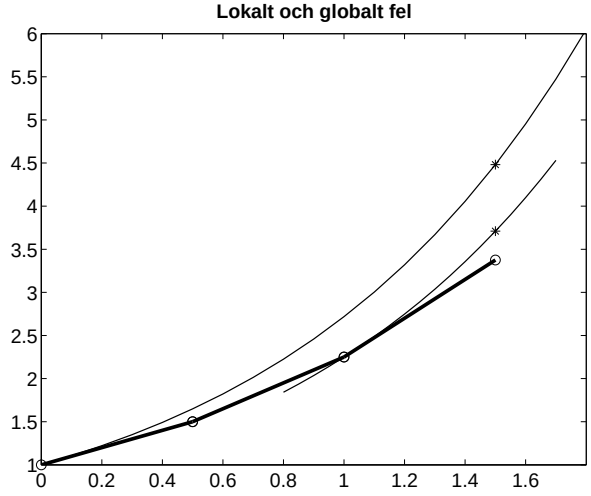
$$y_{k+1} = y_k + h\ f(t_k, y_k)$$

Felkällor

- trunkeringsfel; i Eulers metod trunkerar vi Taylorutvecklingen (approximerar med tangenten)
- avrundningsfel; normalt inte så viktigt

Lokalt fel: felet som uppstår i ett steg när man betraktar startpunkten, (t_{k-1}, y_{k-1}), som exakt. Programvara försöker begränsa detta fel.

Globalt fel: felet mellan approximativ och exakt lösning, $y_k - y(t_k)$



Ordning

Olika metoder har olika ordning: en metod har ordning p om det lokala felet är av storleksordningen h^{p+1} när $h \to 0$. Vi skriver $\mathcal{O}(h^{p+1})$.
Vilken ordning har Eulers metod?

Antag att vi står i punkten (t_{k-1}, y_{k-1}). Vad blir felet i nästa steg förutsatt att (t_{k-1}, y_{k-1}) betraktas som exakt?

Låt oss titta på det speciella problemet $y' = \lambda y, y(0) = y_0$.
Eulers metod ger, som vanligt, approximationerna $y_0, y_1, y_2, \ldots$.

Den exakta lösningen som går genom (t_{k-1}, y_{k-1}) betecknar vi med $z(t)$ och den löser följande problem:

$$z' = \lambda z, \quad z(t_{k-1}) = y_{k-1}$$

Dvs.

$$z(t) = e^{\lambda(t - t_{k-1})} y_{k-1}$$

så när $t = t_k$ är

$$z(t_k) = e^{\lambda(t_k - t_{k-1})} y_{k-1} = e^{\lambda h} y_{k-1}$$

Eulers metod ger:

$$y_k = y_{k-1} + hf(t_{k-1}, y_{k-1}) = (1 + \lambda h) y_{k-1}$$

Det lokala felet blir:

$$y_k - z(t_k) = (1 + \lambda h) y_{k-1} - e^{\lambda h} y_{k-1} =$$

$$\left[1 + \lambda h - \left[1 + \lambda h + \frac{(\lambda h)^2}{2} + \ldots\right]\right] y_{k-1} = -\left[\frac{(\lambda h)^2}{2} + \ldots\right] y_{k-1}$$

som är $\mathcal{O}(h^2)$, så Eulers metod har ordning ett (är en första ordningens metod).

Nu till det globala felet, $y_k - y(t_k)$, där $y(t)$ är den exakta lösningen till $y' = \lambda y, y(0) = y_0$ och y_k är approximationen av $y(t_k)$.

Tydligen är

$$y(t_k) = e^{\lambda t_k} y_0 \quad \text{och} \quad y_k = (1 + \lambda h)^k y_0,$$

Varför?
$y_1 = y_0 + h\lambda y_0 = (1 + h\lambda) y_0$.
$y_2 = y_1 + h\lambda y_1 = (1 + h\lambda) y_1 = (1 + h\lambda)^2 y_0$ etc.

Eftersom $t_k = kh$, får vi följande uttryck för det globala felet:

$$y_k - y(t_k) = (1 + \lambda h)^k y_0 - e^{\lambda t_k} y_0 = (1 + \lambda h)^k y_0 - e^{\lambda k h} y_0 =$$

$$\left[1 + k\lambda h + \frac{k(k-1)}{2}(\lambda h)^2 + \ldots\right] y_0 - \left[1 + k\lambda h + \frac{(k\lambda h)^2}{2} + \ldots\right] y_0 =$$

$$\frac{-k}{2}(\lambda h)^2 y_0 + \ldots = -\frac{1}{2}\lambda^2 (hk) y_0\ h + \ldots = -\frac{1}{2}\lambda^2 t_k y_0\ h + \ldots$$

Så det globala felet uppför sig som h.

Tumregel: det globala felet är $\mathcal{O}(h^p)$.

Vi tappar alltså en potens mellan lokalt och globalt fel.

Vi kan försöka skapa metoder av högre ordning, t.ex. genom att använda tidigare punkter; en så kallad flerstegsmetod.

T.ex.

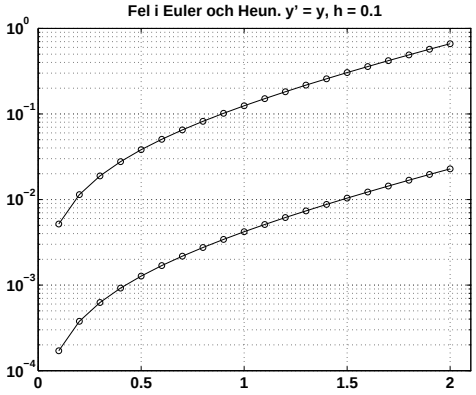
$$y_{k+1} = y_k + \frac{h}{2}\left[3f(t_k, y_k) - f(t_{k-1}, y_{k-1})\right]$$

som har ordning två.
För att starta metoden kan vi ta ett Euler-steg.

En annan metod av andra ordningen är Heuns metod:

$$y_{k+1} = y_k + \frac{h}{2}\left[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))\right]$$

Detta är en enstegsmetod.



System av ekvationer

$$u^{(3)} = u'' - 2tu' + u^2 - t + 1, \quad \begin{cases} u(3) = 2 \\ u'(3) = -1 \\ u''(3) = 0 \end{cases}$$

Inför nya funktioner

$$y_1 = u$$
$$y_2 = u' \Rightarrow y_2 = y_1'$$
$$y_3 = u'' \Rightarrow y_3 = y_2'$$

Vi får

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = y_3 - 2ty_2 + y_1^2 - t + 1 \end{cases}, \quad \begin{cases} y_1(3) = 2 \\ y_2(3) = -1 \\ y_3(3) = 0 \end{cases}$$

Detta problem kan fortfarande skrivas, $y' = f(t, y)$, om vi inför vektorerna y och f, dvs.

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

$$f(t, y) = \begin{bmatrix} y_2 \\ y_3 \\ y_3 - 2ty_2 + y_1^2 - t + 1 \end{bmatrix}, \quad y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

Alla metoder vi har sett kan enkelt generaliseras till systemfallet. Skalära y_k byts mot vektorn $y^{(k)}$. $f(t_k, y_k)$ går över i $f(t_k, y^{(k)})$. Tiden t_k och steglängden h är fortfarande skalärer.

Eulers metod för exemplet ovan blir, med $t_0 = 3$, $h = 0.1$:

$$y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \quad y^{(1)} = y^{(0)} + hf(t_0, y^{(0)})$$

Dvs.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \end{bmatrix} = \begin{bmatrix} y_1^{(0)} \\ y_2^{(0)} \\ y_3^{(0)} \end{bmatrix} + h \begin{bmatrix} y_2^{(0)} \\ y_3^{(0)} \\ y_3^{(0)} - 2t_0 y_2^{(0)} + \left(y_1^{(0)}\right)^2 - t_0 + 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.9 \\ -1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} + 0.1 \begin{bmatrix} -1 \\ 0 \\ 0 - 2 \cdot 3\ (-1) + 2^2 - 3 + 1 \end{bmatrix}$$

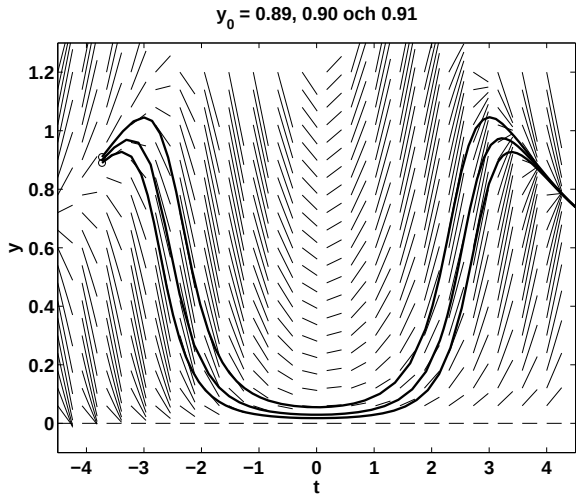
osv.

Problemets stabilitet

Hur ändras lösningen vid små ändringar i problemet?

I följande bild visas hur lösningen (till $y' = \sin(ty)$) varierar med $y(t_0)$. $y(t_0) = 0.89, 0.90$ respektive 0.91.

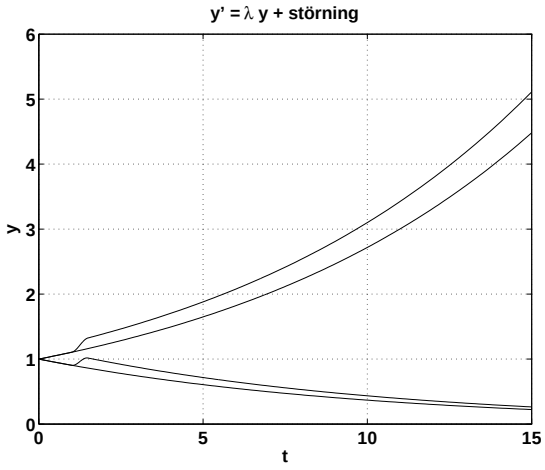
Om lösningskurvorna går ihop eller går isär avgörs av det lokala utseendet på riktningsfältet.



En lösning är stabil om två lösningar kan fås att ligga godtyckligt nära varandra (för $t \geq t_0$) givet att vi stör tillräckligt lite.

I nedanstående bild har jag löst $y' = \lambda y + s(t)$, för ett positivt och ett negativt värde på λ.
$s(t)$ är en liten störning som inträffar omkring $t = 1$.

Den exakta lösningen till $y' = \lambda y$ är $y(t) = e^{\lambda t} y(0)$.



Vi ser att störningen dämpas ut när $\lambda < 0$.

Om λ är komplext med negativ realdel så är differentialekvationen stabil; felen dämpas ut.
Om realdelen är positiv är differentialekvationen instabil.

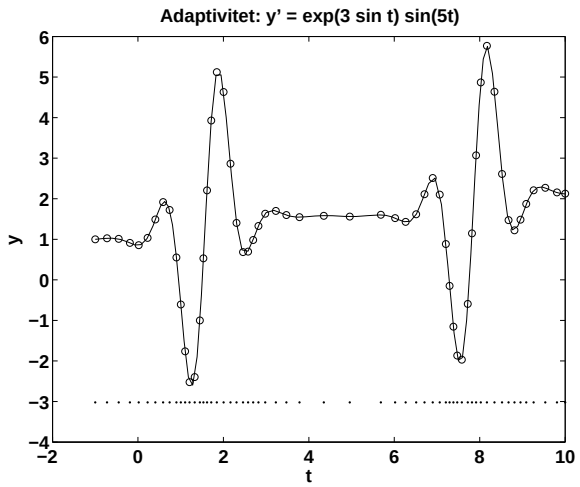
Detta kan generaliseras till ickelinjära problem och system av sådana.

Adaptivitet

De flesta ODE-lösare är adaptiva, dvs. de försöker att anpassa steglängden så att det lokala felet underskrider en tolerans given av programmets användare.

I vissa fall består programmet av en familj av metoder av olika ordning. Programmet kan då även variera ordningen.

I figuren nedan har jag löst ett problem med `ode45` (den heldragna lösningen) och `ode23`, med stor tolerans, (ringarna).



Styva problem och lösarens stabilitet

Det är vanligt med så kallade styva problem (stiff).
Dessa uppkommer t.ex. när man har snabba transienter.

Om vi använder en vanlig ode-lösare på ett styvt problem tvingas lösaren ta mycket korta steg för bibehålla stabiliteten.

Det visar sig att vi kan lära oss mycket om metoders stabilitet genom att studera den skalära testekvationen, $y' = \lambda y, y(0) = 1$.
Normalt har vi dock styva system (och inte skalära ekvationer).

Antag att $\lambda < 0$, den exakta lösningen är då avtagande.

För vilka h ger Eulers metod $y_k \to 0$ då $k \to \infty$?

$$y_{k+1} = y_k + hf(t_k, y_k) = y_k + h\lambda y_k = (1 + h\lambda) y_k$$

så att

$$y_k = (1 + h\lambda)^k$$

När gäller att $y_k \to 0$? Jo då:

$$|1 + h\lambda| < 1$$

dvs, om $\lambda \in \Re$ (och $\lambda < 0$),

$$0 < h|\lambda| < 2$$

Antag nu att λ är ett mycket negativt tal, säg $\lambda = -20000$. För att vi överhuvudtaget skall få en lösning som går mot noll måste $h < 1/10000$.

Vi noterar att $e^{\lambda t} = \epsilon_{mach}$ om $t = (\log \epsilon_{mach})/\lambda \approx 2 \cdot 10^{-3}$ i vårt exempel.

Vad skall vi göra? Lösningen är implicita metoder.

Bakåt-Euler:

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

Stabilitet? Testa på $y' = \lambda y$

$$y_{k+1} = y_k + h\lambda y_{k+1}$$

så att

$$y_{k+1} = (1 - h\lambda)^{-1} y_k$$

och

$$y_k = (1 - h\lambda)^{-k} \quad \text{ty} \quad y_0 = 1$$

När är $|(1 - h\lambda)^{-1}| < 1$? Antag $\lambda < 0$ (reellt)
då är $|(1 - h\lambda)^{-1}| < 1$ för alla $h > 0$!

Detta innebär givetvis inte att vi kan ta godtyckligt långa steg. Tar vi för långa steg blir felet för stort.

Implicita metoder har den nackdelen att vi måste lösa en (normalt ickelinjär) ekvation för att bestämma y_{k+1}.

I en explicit metod, som Eulers metod, är detta inte nödvändigt.

Det finns mer komplicerade implicita metoder, t.ex.

$$y_{k+1} - \frac{4}{3} y_k + \frac{1}{3} y_{k-1} = \frac{2h}{3} f(t_{k+1}, y_{k+1})$$

som är ett exempel på en flerstegsmetod.

Ett exempel:

$$y' = \begin{bmatrix} y_2 \\ -(y_1 + 2y_2)/\epsilon \end{bmatrix}, \quad y(0) = \begin{bmatrix} -1 \\ 10 \end{bmatrix}$$

Med Matlabs `ode23` (en Runge-Kutta-lösare ordning 2 och 3) krävs 11989 steg för att lösa problemet då $\epsilon = 0.0002$. Toleranserna är relativt 10^{-3} och absolut 10^{-6}.

Matlabs ode23s (s för stiff) löser problemet i 192 steg. 140 av dessa steg tas för $t < 0.01$.

Andra problemklasser

Tvåpunkts randvärdesproblem:

$$y'' = f(t, y, y'), \quad \alpha_1 y(a) + \beta_1 y'(a) = \gamma_1, \quad \alpha_2 y(b) + \beta_2 y'(b) = \gamma_2$$

Egenvärdesproblem (vibrerande sträng):

$$(py')' + \lambda \rho y = 0$$

$y(a) = y(b) = 0$, fixerade ändpunkter
$y'(a) = y'(b) = 0$, fria ändpunkter
$y(a) = y(b)$, $y'(a) = y'(b)$, periodiska randvillkor.

Ickelinjärt egenvärdesproblem (bifurkationsproblem).
Knäckning, roterande kedja, Taylor-Couette.

$$y'' + \frac{\lambda y}{\sqrt{y^2 + t^2}} = 0 \quad \text{samt randvillkor}$$

Tidsfördröjningsproblem (delay equations)

$$y'(t) = y(t) - y(t - T) + \ldots$$

Inkubationstid; ändlig utbredningshastighet...

Differentialalgebraiska problem: differentialekvation med algebraiska "bivillkor".
Specialfall, implicita problem: $g(t, y)y' = f(t, y)$.