Föreläsningsanteckiningar för kursen "Numerisk Analys", MMG410

Larisa Beilina, larisa@chalmers.se

March 22, 2021

Kursinformation

- Distansundervisning i Zoom
  - Zoom -länk för föreläsningar och datorlabbar - se kursens sida i CANVAS
  - Jitsi länk för datorlabbar: https://meet.jit.si/mmg410complab
- Kursansvarig och examinator: Larisa Beilina, larisa@chalmers.se
- Handledare för datorlaborationer och övningar med Matlab:
  - Morgan Görtz, morgan.gortz@fcc.chalmers.se,
- Registrering på kursen: kontakta studieadministratör Jeanette Montell, jw@chalmers.se.

Schema

Dag	Tid	Plats	Typ
Mån	13:15-15:00	Zoom	Förel
Ons	13:15-15:00	Zoom	Förel
Fre	13:15-15:00	Zoom	Förel
Mån	15:15-17:00	Zoom	Datorlab
Ons	15:15-17:00	Zoom	Datorlab
Fre	15:15-17:00	Zoom	Datorlab
04.06.2021	14.00-18.00	?	Tentamen
25.08.2021	14.00-18.00	?	Omtentamen
?.01.2022	14.00-18.00	?	Omtentamen

Kurslitteratur

- **Michael T. Heath, Scientific Computing - An introductory survey, McGraw-Hill, 2002.**

  Den äldre upplagan från 1997 duger också. Köp boken via internet.
- **Föreläsningsanteckningar** finns på kursens hemsidan. Flera studenter tycker att boken ej är nödvändig nu när det finns föreläsningsanteckningar (kopior av slides) på kursens hemsidan. Om man skall klara sig med dessa kopior måste man nog gå på föreläsningarna.
- Mina anteckningar/slider.

Former för bedömning

- Kursen består av två poäng-givande moment, laboration och tentamen, 3 Hp för lab och 4.5 Hp för tentamen.
- Vi planerar att ha 3 bonuspoängövningar, som ska utföras i en grupp av cirka 10 personer/grupp. Vi ska testa om vi kan göra de i Canvas nu när vi har distansundervisning. Hela gruppen kan få max 0.5 bp. för varje övningstillfälle, max 1.5 b.p. för hela kursen. Tider för bonuspoängövningar finns på kursens hemsida.
- Skriftlig tentamen samt examination av datorlaborationer i form av skriftliga redovisningar via Canvas.
- Tre obligatoriska laborationer som skall utföras i grupper om precis två personer. Redovisa en lab så fort du är färdig.
- För att erhålla betyg på hela kursen krävs att samtliga obligatoriska moment fullgjorts.
- Vi ska ha 2 extra tentamenstillfällen: i augusti och i januari.

Betyg

- Betygskalan omfattar betygsgraderna Underkänd (U), Godkänd (G) och Väl godkänd (VG).

Skriftlig tentamen	Matlab övningar	Betyg på hela kursen
VG	G	VG
VG	U	U
G	G	G
G	U	U
U	G	U

- Student som enligt avtal har rätt att få betyg satt med ECTS-skalan ska informera kursansvarig om detta senast en vecka efter kursstart. För student utan sådant avtal sätts inga ECTS-betyg. En ECTS-översättning görs schablon-mässigt enligt av rektor fastställd mall.

Kursutvärdering

Kursutvärdering görs med en enkät och samtal med studentrepresentanter.
På kursens aktivitet i GUL (inloggning via Studentportalen) finns en enkät som används vid utvärderingen. Utvärderingen sker genom samtal mellan lärare och studentrepresentanter under kursens gång samt vid ett möte efter kursens slut då enkätresultatet diskuteras och rapport skrivs på speciell blankett.

Kursinnehåll

- Grundläggande egenskaper hos flyttalsräkning.
- Grundläggande begrepp, felanalys och datoraritmetik.
- NLA problem och minstakvadratproblem.
- Några vanliga numeriska metoder för interpolation, derivering, integrering.
- Lösning av icke-linjära ekvationer, system av linjära och icke-linjära ekvationer samt Ordinarie differentialekvationer.

Kursmål

Efter avslutad kurs skall studenten

- vara förtrogen med grundläggande egenskaper hos flyttalsräkning;
- kunna bedöma tillförlitligheten hos beräknade resultat;
- kunna ställa upp några grundläggande numeriska problem på standardform;
- kunna härleda grundläggande metoder för några beräkningsproblem;
- kunna lösa enkla tillämpningsproblem med hjälp av Matlab.

Fyra sista punkterna endast avser de problemområden som står under rubriken “Kursinnehåll”.

Introduktion. 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 (stabiliteten) ?
- Algoritmens egenskaper:
  - Hur snabb är metoden, implementationen?
  - Hur mycket minne går åt?
  - Vilka fel introduceras av algoritmen (avrundningsfel etc)?

Olika typer av fel

Fel som vi som numeriker inte kan göra så mycket åt, är:

- modellfel, bortser från luftmotstånd, friktion.
  Exempel: tidsberoende Maxwell's ekvation (PDE):

$$\begin{aligned} \varepsilon(x)\frac{\partial^2 E(x,t)}{\partial t^2} + \nabla \times \nabla \times E(x,t) = 0, \text{ in } \Omega \times (0,T], \\ \nabla \cdot (\varepsilon E)(x,t) = 0, \end{aligned} \tag{1}$$

  Använder transformation:

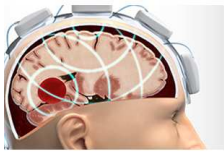
$$\nabla \times \nabla \times E = \nabla(\nabla \cdot E) - \nabla \cdot (\nabla E) \tag{2}$$

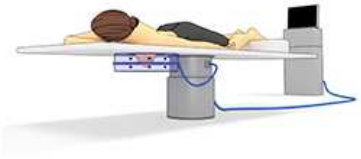
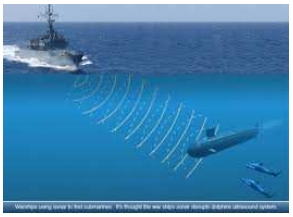
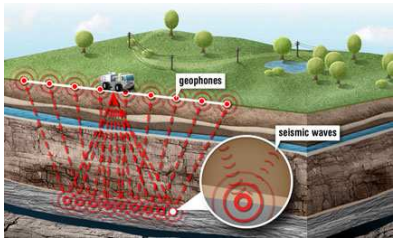
  Ny approximation av Maxwell's ekvation:

$$\varepsilon(x)\frac{\partial^2 E(x,t)}{\partial t^2} - \triangle E(x,t)) = 0, \text{ in } \Omega \times (0,T], \tag{3}$$

- mätfel, vågar etc. är inte exakta mellanavrundningar

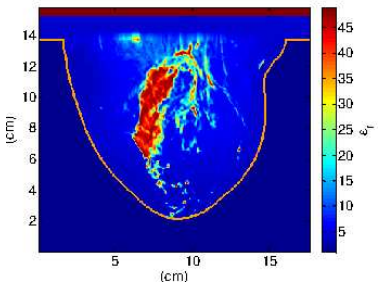
Olika typer av fel: modellfel (exempel)

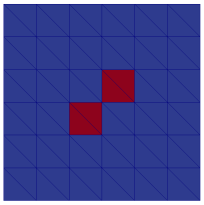


- Breast cancer, land mines, oil prospecting, ability to see through the walls and construction of "invisible materials" can all be modelled and computed using different types of wave equations: acoustic, elastic or electromagnetic.
- Figure shows: Biomedical Imaging at the Department of Electrical Engineering at CTH, Chalmers. Upper Left: setup of Stroke Finder and right: microwave hyperthermia in cancer treatment. Upper Right: breast cancer detection using microwave tomography.

Olika typer av fel: modellfel (exempel)





ε_r som funktion: real data $\varepsilon_r = const.$

Tidsberoende Maxwell's ekvation (PDE):

$$\varepsilon(x)\frac{\partial^2 E(x,t)}{\partial t^2} + \nabla \times \nabla \times E(x,t) = 0, \text{ in } \Omega \times (0,T], \qquad (4)$$
$$\nabla \cdot (\varepsilon E)(x,t) = 0,$$

Ny approximation av Maxwell's ekvation:

$$\varepsilon(x)\frac{\partial^2 E(x,t)}{\partial t^2} - \triangle E(x,t)) = 0, \text{ in } \Omega \times (0,T], \qquad (5)$$

Example

Vi är intresserade av olika typer av beräkningsfel:

- Avrundningsfel i Matlab:
  49 * (1 / 49) - 1
  ans = -1.1102e-16
  Men:
  49/49 - 1
  ans = 0
- Trunkeringsfel: exempel (Taylorsutveckling för e^x)

$$e^x \approx \sum_{k=0}^{N} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + ... + \frac{x^N}{N!}$$

- Diskretiseringsfel: $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
  Viktigt att välja "lagom stort" h.

Fel: absoluta och relativa felet (enkelt fall)

Låt $\hat{x}$ vara en approximation av det exakta värdet x när $\hat{x} \geq x$.
Vi definerar:

- absoluta felet

$$e = \hat{x} - x$$

- relativa felet för $x \neq 0$ är:

$$e_r = \frac{\hat{x} - x}{x}$$

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

Example

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 $1.4 \cdot 10^{-9}$ därför att:
a) Absoluta fel: $1.4 = \hat{x} - 2$, relativa felen: $0.7 = \frac{\hat{x}-2}{2}$.
b) Absoluta fel: $1.4 = \hat{x} - 10^9$, relativa felen: $1.4 \cdot 10^{-9} = \frac{\hat{x}-10^9}{10^9}$.

Fel: absoluta och relativa felet (gemensamt fall)

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

- absoluta felet

$$e = |\hat{x} - x|$$

- relativa felet för $x \neq 0$ är:

$$e_r = \frac{|\hat{x} - x|}{|x|}$$

Kom ihåg:

$$|x| = \begin{cases} x & \text{om } x \geq 0 \\ -x & \text{om } x < 0 \end{cases}$$

Fel: absoluta och relativa felet (gemensamt fall)

Kom ihåg:

$$|x - y| = \begin{cases} x - y & \text{om } x \geq y \\ y - x & \text{om } x < y \end{cases}$$

Example

Är 1.4 ett stort absolut fel? Ja, om det exakta värdet är 2, men inte om det exakta värdet är 10^9.
a) Absoluta fel i gemensamt fall: $1.4 = |\hat{x} - 2|$,

$$1.4 = |\hat{x} - 2| = \begin{cases} \hat{x} - 2 & \text{om } \hat{x} = 3.4 \\ 2 - \hat{x} & \text{om } \hat{x} = 0.6 \end{cases}$$

relativa felen:

$$0.7 = \frac{|\hat{x} - 2|}{|2|} = \begin{cases} \frac{\hat{x}-2}{2} & \text{om } \hat{x} = 3.4 \\ \frac{2-\hat{x}}{2} & \text{if } \hat{x} = 0.6 \end{cases}$$

b) Absoluta fel: $1.4 = |\hat{x} - 10^9|$, relativa felen: $1.4 \cdot 10^{-9} = \frac{|\hat{x}-10^9|}{|10^9|}$.

Fel

På samma sätt kan det absoluta felet 10^{-20} vara stort eller litet. Det är viktigt att känna till problemets skalning.

Example

Absoluta fel för exacta 2 (om $\hat{x} \geq 2$) : $10^{-20} = \hat{x} - 2$, relativa felen: $0.5 \cdot 10^{-20} = \frac{\hat{x}-2}{2}$.

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.

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:

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

Vi ser att alla rötterna $x = 1$. Men i Matlab har vi:
koefficienter :
$r = roots([1 \ -5 \ 10 \ -10 \ 5 \ -1])$
rötterna:
$r = 1.0008 + 0.0006i$
$1.0008 - 0.0006i$
$0.9997 + 0.0009i$
$0.9997 - 0.0009i$
0.9990
Felen:
$disp(abs(r-1)')$
1.1322e-03 1.1322e-03 1.1326e-03 1.1326e-03 1.1328e-03

Nollställen till polynom

Varför? Lös

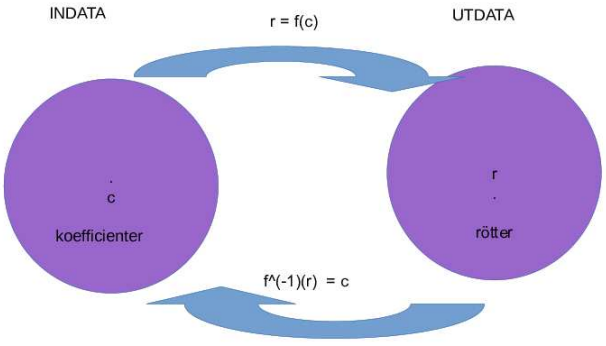
$$(x-1)^5 = \varepsilon,$$

då

$$x = 1 + \varepsilon^{1/5}$$

Om $\varepsilon = 10^{-15}$ så är $\varepsilon^{1/5} = 10^{-15/5} = 10^{-3}$. Nollställena till polynomet $(x-1)^5$ är tydligen känsliga för störningar i koefficienterna.
Är det alltid svårt att beräkna nollställen?
Koefficienter:
c = [1 -15 85 -225 274 -120];
De exakta röttena är olika nu = 1,2,3,4,5:
r = roots(c);
fel = sort(r) - (1:5)'
Felen är mycket mindre nu:
fel = -4.9960e-15 6.6613e-14 -1.5010e-13 9.6811e-14 -8.8818e-16

Konditionstal



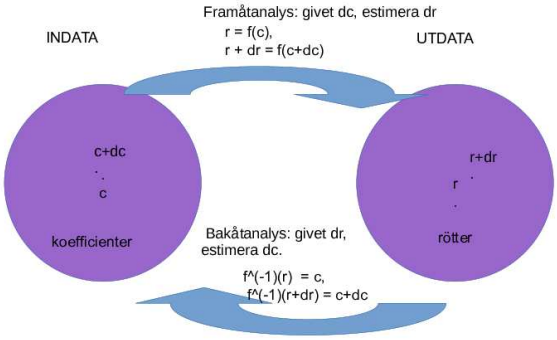
Vi kan betrakta rötterna r som funktioner $f(c)$ av koefficienterna c:

$$r = f(c),$$

var r är lösning för

$$p(x) = \sum_{i=0}^{d} c_i x^i = c_0 + c_1 x + c_2 x^2 + \ldots + c_d x^d = 0.$$

Konditionstal



När vi stör koefficienterna $c + \delta c$, då stör vi också rötterna $r + \delta r$
Om liten relativ ändring av indata $|\delta c|/|c|$ ger en liten relativ ändring av resultatet $|\delta r|/|r|$ säger man att det aktuella problemet är välkonditionerat. Om resultatet ändrar sig mycket är problemet illakonditionerat. Konditionstalet är kvoten mellan de relativa förändringarna, dvs.

$$k = \frac{|\delta r|/|r|}{|\delta c|/|c|}$$

Stabilitet

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 k så att

$$|\delta r|/|r| \leq k|\delta c|/|c|.$$

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

Example

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 $r = (r_1, r_2)$ beteckna en av rötterna och låt $r + \delta r$ beteckna den störda roten när vi ändrar koefficienterna med δa respektive δb.

Example

Vi har sambandet:

$$x^2 + ax + b = (r + \delta r)^2 + (a + \delta a)(r + \delta r) + (b + \delta b) = 0,$$

och vi kan skriva om den:

$$(r^2+ar+b)+(\delta r(2r+a)+\delta ar+\delta b)+((\delta r)^2+\delta a\delta r) = I_1+I_2+I_3 = 0,$$

var

$$\begin{aligned} I_1 &= (r^2 + ar + b) = 0, \\ I_2 &= (\delta r(2r + a) + \delta ar + \delta b) \approx 0, \\ I_3 &= ((\delta r)^2 + \delta a\delta r) \approx 0. \end{aligned} \tag{6}$$

Example

Från andra ekvation i systemet (6) får vi:

$$\delta r \approx -\frac{(\delta a\ r + \delta b)}{2r + a}$$

eller

$$|\delta r| \leq \frac{(|\delta a\ r| + |\delta b|)}{|2r + a|} \tag{7}$$

Eftersom r_1 och r_2 är rötter så gäller att:

$$(x - r_1)(x - r_2) = x^2 - (r_1 + r_2)x + r_1 r_2 = x^2 + ax + b$$

Vi kan jämföra koefficienterna och får

$$-(r_1 + r_2) = a,\ \ b = r_1 r_2.$$

Vi kan skriva om $r_1 - r_2 = 2r_1 + a$, och definera gapet $g := |r_1 - r_2|$.

Example

Vi kan skriva om (7)

$$|\delta r| \leq \frac{|\delta a\ r| + |\delta b|}{|g|} \tag{8}$$

Om g är liten eller $r_1 \approx r_2$, då $|\delta r|$ är stort. Dividera (8) med $|r|$ och förläng med $|a|$ respektive $|b|$.

$$\frac{|\delta r|}{|r|} \leq \frac{1}{|r|}\left(\frac{\frac{|a|}{|a|}|\delta a\ r| + \frac{|b|}{|b|}|\delta b|}{|g|}\right) \leq k \max\left(\frac{|\delta a|}{|a|}, \frac{|\delta b|}{|b|}\right), \tag{9}$$

var konditionstalet är $k \approx \frac{|a|+|b/r|}{g}$.
Observera att detta är en uppskattning av konditionstalet. Det är inte heller beräkningsbart eftersom vi måste känna r_1 och r_2.

Example

Låt

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

Vi vet sedan tidigare att konditionstalet $k \approx \frac{|a|+|b/r|}{g}$ med gapet $g := |r_1 - r_2|$ har storleksordningen $1/(1.0001-1) = 10^4$:

$$k \approx \frac{|-2.0001| + |1.0001/r|}{|1.0001 - 1|} \approx 3 \cdot 10^4.$$

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

$$\frac{|1-1.11|}{|1|} = 0.11, \quad \frac{|1.0001 - 0.895|}{|1|} \approx 0.105.$$

Example

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

$$(x - 1.11)(x - 0.895) = x^2 - 2.005x + 0.99345$$

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.

Framåt- och Bakåtanalys

Vad händer när vi stör koefficienterna c (indata i det allmänna fallet) med δ_c ? Vi har sett s.k. framåtanalys: givet δc vad blir

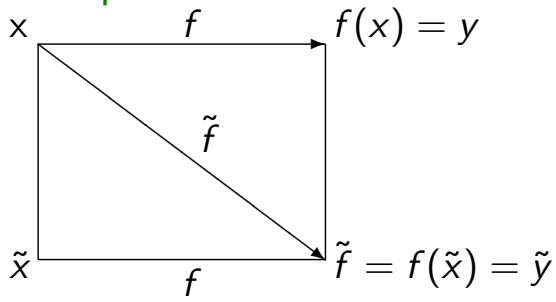
$$\tilde{y} - y = f(c + \delta c) - f(c),$$

var $\tilde{y} = f(c + \delta c), y = f(c)$. 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 δc 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.

Example



Framåtfelet: $|y - \tilde{y}|$; Bakåtfelet: $|x - \tilde{x}|$;
$f(x) = \sqrt{x} = y$; $f(\tilde{x}) \approx \sqrt{2} \approx 1.4 = \tilde{y}$
Låt $y = 1.41421\ldots$
Framåtfelet: $|y - \tilde{y}| = |1.4 - 1.41421| \approx 0.014 \approx 1\%$
Bakåtfelet: $(1.4)^2 = 1.96 = \tilde{x}$,
$\sqrt{1.96} = 1.4$ och $|x - \tilde{x}| = |1.96 - 2| = 0.04 \approx 4\%$

Övning

Vi vill lösa ekvationen $x^2 + ax + b = 0$ då vi vet att a och b båda är positiva och där a är mycket större än b, $a >> b$. Den matematiska formeln inte fungerar tillfredsställande när vi räknar med avrundningsfel. Visa att rötterna är välkonditionerade genom att uppskatta konditionstalen med formeln som vi härledde på föreläsning 1 (det finns en stor rot (mycket negativ) och en liten (nära noll)).
Vi kan uppskatta konditionstalen enligt formeln som vi härledde på föreläsning för $x^2 + ax + b = 0$:

$$k = \frac{|a| + |b/r|}{|g|}$$

Visa att den stora roten går bra att beräkna med standardformeln, men att det blir problem med den lilla. Försök att hitta en bra algoritm för den lilla roten. Taylorutveckling är, som oftast, ett användbart redskap i detta sammanhang.

Övning

Lösning:
Låt oss kalla den stora (negativa) roten R och den lilla, nära noll, r.
Standardformeln och Taylorutveckling ger:

$$R = -\frac{a}{2} - \sqrt{\frac{a^2}{4} - b} = -\frac{a}{2}\left[1 + \sqrt{1 - \frac{4b}{a^2}}\right] = -\frac{a}{2}\left[2 - \frac{2b}{a^2} - \frac{2b^2}{a^4} - ...\right] \approx -a,$$

$$r = -\frac{a}{2} + \sqrt{\frac{a^2}{4} - b} = \frac{a}{2}\left[-1 + \sqrt{1 - \frac{4b}{a^2}}\right] = \frac{a}{2}\left[-\frac{2b}{a^2} - \frac{2b^2}{a^4} - ...\right] \approx -\frac{b}{a}$$

Vi kan uppskatta konditionstalen enligt formeln som vi härledde på föreläsning 1 :

$$k_R = \frac{|a| + |b/R|}{|R - r|} \approx \frac{a + b/a}{a} \approx 1,$$

$$k_r = \frac{|a| + |b/r|}{|R - r|} \approx \frac{a + b/(b/a)}{a} \approx 2.$$

När vi beräknar $r = -\frac{a}{2} + \sqrt{\frac{a^2}{4} - b}$ kommer att få utskiftning av b. I det mest extrema fallet kommer inte b alls med och approximationen blir noll. Hur skall vi beräkna r? Ett sätt är att använda utvecklingen ovan:

$$r = -\frac{b}{a} - \frac{b^2}{a^3} - \frac{2b^3}{a^5}...$$

Ett standardtrick är att förlänga med konjugatet,

$$r = \frac{\left(-\frac{a}{2} + \sqrt{\frac{a^2}{4} - b}\right)\left(-\frac{a}{2} - \sqrt{\frac{a^2}{4} - b}\right)}{-\frac{a}{2} - \sqrt{\frac{a^2}{4} - b}} = \frac{b}{-\frac{a}{2} - \sqrt{(\frac{a}{2})^2 - b}}.$$

Ytterligare ett sätt, är att göra en transformation så att r blir en dominant rot i det transformerade problemet. Sätt $y = 1/x$ (så att $r \to 1/r$). Ekvationen $x^2 + ax + b = 0$ övergår då till $y^2 + (a/b)y + 1/b = 0$. Om vi använder standardformeln får vi för den sökta roten:

$$\frac{1}{r} = -\frac{a}{2b} - \sqrt{\frac{a^2}{4b^2} - \frac{1}{b}}.$$

Diskretiseringsfel (trunkeringsfel) och avrundningsfel

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

- För stora h dominerar diskretiseringsfelet(trunkeringsfel), man kan bortse från avrundningsfelet.
- För små h dominerar avrundningsfelet.
  Se approximation av $f'(x)$: kancellation i täljaren för små h och division med litet tal förstärker felet i täljaren.

Diskretiseringen för $f'(x)$: första ordningen noggrannhet

Approximativt värde (Taylor's theorem):

$$f(x+h) = f(x) + f'(x)h + \frac{f''(Q)h^2}{2!},$$

$Q \in [x, x+h]$

$$f(x+h) - f(x) = f'(x)h + \frac{f''(Q)h^2}{2!}.$$

Dividera med h:

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f''(Q)h}{2!}$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(Q)h}{2!}$$

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

Trunkeringsfel för $f'(x)$

Trunkeringsfel:

$$\frac{f''(Q)h}{2} = \frac{f(x+h) - f(x)}{h} - f'(x).$$

Låt $M \leq |f''(Q)|$, då trunkeringsfel ε är begränsad med

$$\varepsilon < \frac{Mh}{2}.$$

Vi fick för $f'(x)$ första ordning noggrannhet.

Diskretiseringen för $f'(x)$: andra ordning noggrannhet.

Approximativt värde (Taylor's theorem):

$$(*)\ f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(Q)h^3}{3!}$$

$$(**)\ f(x-h) = f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(Q)h^3}{3!}$$

$(*) - (**)$:

$$f(x+h) - f(x-h) = 2f'(x)h + 2\frac{f'''(Q)}{3!}h^3$$

$$2f'(x)h = f(x+h) - f(x-h) - 2\frac{f'''(Q)}{3!}h^3$$

Eller

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{2f'''(Q)h^3}{3! \cdot 2h}$$

Trunkeringsfel för $f'(x)$: andra ordning noggrannhet.

Trunkeringsfel:

$$\frac{2f'''(Q)h^3}{3! \cdot 2h} = \frac{f(x+h) - f(x-h)}{2h} - f'(x).$$

Låt $M \leq |f'''(Q)|$, då trunkeringsfel ε är begränsad med

$$\varepsilon < \frac{Mh^2}{6}.$$

Vi fick för $f'(x)$ andra ordning noggrannhet.

Diskretiseringen för $f''(x)$: andra ordning noggrannhet.

Approximativt värde (Taylor's theorem):

$$(*)\ f(x-h) = f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(x)h^3}{3!} + \dots$$

$$(**)\ f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \dots$$

$(*) + (**)$:

$$f(x+h) + f(x-h) = 2f(x) + \frac{2f''(x)}{2!}h^2 + O(h^4)$$

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x) - O(h^4)}{h^2}$$

$O(h^4) = \frac{2f^{(4)}(x)}{24}h^4$; $\frac{f^{(4)}(x)}{12}\frac{h^4}{h^2} = \frac{f^{(4)}(x)}{12}h^2 \longrightarrow$ Låt $M \leq |f^{(4)}(Q)|$, då trunkeringsfel ε är begränsad med $\varepsilon < \frac{Mh^2}{12}$, och $f''(x)$ har andra ordningen noggrannhet.

Representation av tal i dator

- Alla tal lagras i ett begränsat antal bitar i minnet, vanligen i binär form.
- För heltal: lagring i dator är utan problem. Heltal upp till en viss storlek lagras exakt.
- Reella tal lagras exakt utan måste avrundas. Representation av reella tal kallas flyttalsrepresentation och talen kallas flyttal.

IEEE flyttalsrepresentation

IEEE 754 (Institute of Electrical and Electronics Engineers, Inc.) definierar enkel och dubbel precision (bland annat).

- Under 60- och 70-talen hade varje datortillverkare sitt eget flyttalsystem.
- En flyttalstandard utvecklades under tidigt 80-tal och följdes av tillverkare som Intel och Motorola.
- IEEE standarden har 3 viktiga krav: konsistent flyttalsrepresentation, korrekt avrundningsaritmetik, konsistent hantering av exceptionella situationer.

Flyttalsaritmetik

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

var

$$0 \leq d_k \leq \beta - 1, L \leq e \leq U,$$

Exempel:

$$-3.25: \ -[1.625] \cdot 2^1 = -\underbrace{[1 + 0.625]}_{\textit{mantissa}} \cdot 2^1$$

Här:

- β bas (s vi kommer att ha $\beta = 2$)
- e exponent (heltal)
- t precision
- $[L, U]$ exponentomfång
- $d_k, k = 0, ..., t - 1$ mantissa (heltal)

Flyttalsaritmetik

- β bas (s vi kommer att ha $\beta = 2$)
- e exponent (heltal)
- t precision
- $[L, U]$ exponentomfång
- $d_k, k = 0, ..., t-1$ mantissa (heltal)

Vi antar att $= 2$ från och med nu, då:

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

Flyttalsaritmetik

- IEEE enkel precision: tecknet ("+" 0, "-" 1) 1 bit, exponent 8 bitar, mantissa 23 bitar $= 32$ bitar:

$$\pm e_1 e_2 ... e_8 d_0 d_1 ... d_{22}$$

- IEEE dubbel precision: tecknet ("+" 0, "-" 1) 1 bit, exponent 11 bitar, mantissa 52 bitar $= 64$ bitar:

$$\pm e_1 e_2 ... e_{11} d_0 d_1 ... d_{51}$$

Ett tal $\neq 0$ är normaliserat om $d_0 \neq 0$.
Om $\beta = 2$ så är $d_0 = 1$ varför man inte lagrar d_0.
Det finns speciella bitformat för diverse specialfall, t.ex:
[0, -0]
ans = 0000000000000000 8000000000000000

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

Antalet olika tal räknas som:

$$2(\beta-1)\beta^{t-1}(U-L+1)+1$$

och är

- $4.2614 \cdot 10^9$ i enkel precision: $\beta = 2, L = -126, U = 127, t = 24$ i formula $2(\beta-1)\beta^{t-1}(U-L+1)+1$.
- $1.8429 \cdot 10^{19}$ i dubbel precision: $\beta = 2, L = -1022, U = 1023, t = 53$ i formula $2(\beta-1)\beta^{t-1}(U-L+1)+1$.
- Att testa en funktion för alla flyttal i dubbel precision är nästan omöjligt. 109 tester per sekund ger 584.4 år.

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

- Minsta positiva representerbara normaliserade talet är $2^L \approx 1.17 \cdot 10^{-38}$ i enkel (se tabell, $L = -126$) och $\approx 2.2 \cdot 10^{-308}$ i dubbel precision (se tabell, $L = -1022$).
- Det största representerbara talet har största exponenten och ettor i hela mantissan. I enkel precision $\approx 3.4 \cdot 10^{38}$, i dubbel $\approx 1.8 \cdot 10^{308}$.
- Tal större än största representerbara talet ger overflow och mindre än minsta positiva representerbara normaliserade talet ger underflow.
- Underflow Level: $UFL = \beta^L$, Overflow Level: $OFL = \beta^{U+1}(1-\beta^{-t})$. I enkel precision OFL räknas som: $OFL = (2^{128}) \cdot (1-(2^{-24})) \approx 3.4 \cdot 10^{38}$, i dubbel OFL räknas som: $OFL = (2^{1024}) \cdot (1-(2^{-53})) \approx 1.8 \cdot 10^{308}$.
- Exempel: Matlab programet floatgui.m. Floating-point system: $\beta = 2, p = 3, L = -1, U = 1$. Antalet flytttal: $2(\beta-1)\beta^{t-1}(U-L+1)+1 = 25$.

Example

När vi skriver i Matlab:

- $1e-200^2 = 10^{-200^2}$ ger underflow, svar 0.
- $1e200^2$ ger overflow, svar infinity.
- $log(0)$, svar -infinity
- sin(1/0), svar NaN (not a number)
- exp(log(10000)) − log(exp(10000)), *svar* :exp(log(10000)) = 10000, exp(10000) = Inf och log(exp(10000)) = Inf och exp(log(10000)) − log(exp(10000)) = −Inf
- exp(log(10)) − log(exp(10)), svar: exp(log(10)) = 10, log(exp(10)) = 10 och exp(log(10000)) − log(exp(10000)) = 0.

Övning: talet i binär form som flyttal i dator.

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

Example

$$-3.25: \ -[1.625] \cdot 2^1 = -\underbrace{[1 + 0.625]}_{mantissa} \cdot 2^1$$

Exponenten $e = 1$ lagras som: $1 + 1023 = 1024 = 2^{10}$. Mantissa: 1 kodas inte,

$$0.625 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \ldots = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \ldots$$

$\frac{1}{2} = 0.5 < 0.625; \quad \frac{1}{4} = 0.25 : 0.625 - 0.5 = 0.125; 0.25 > 0.125$, darför $\frac{1}{4} \cdot 0$, $\frac{1}{8} = 0.125 = 0.125$ och darför $\frac{1}{8} \cdot 1$ och stop (resten i mantissa ska vara 0).

Vi får följande binär representation för -3.25 :

1	10000000000	1010 0
tecken	exponent 11 bitar	mantissa 52 bitar

Example

-3.25 i binär form lagras som:

1	10000000000	1010 0
tecken	exponent 11 bitar	mantissa 52 bitar

-3.25 i hexadecimalt (bas 16) form lagras som:
c00a000000000000

Bas 16:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										a	b	c	d	e	f

Nu grupperar vi om binär form för -3.25 i 4 bitar:

1100	0000	0000	1010	0000	....	0000

och kodar första fyra bitar: 1100 = c

$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 12 = c$

0000 koderas som 0, och sedan

1010 = a

$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 10 = a$

Example

$-9.28 := -[1.16] \cdot 2^3 = -[1 + 0.16] \cdot 2^3$

$$0.16 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \frac{1}{32}x_5 + \ldots =$$
$$= \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 0 + \ldots$$

Exponenten 3 lagras som: $3 + 1023 = 1026 = 1024 + 2 =$
$1 \cdot 2^{10} + 1 \cdot 2^1 + 0 \cdot 2^0$

1	10000000010	00101 ....
tecken	exponent 11 bitar	mantissa 52 bitar

$\frac{1}{2} = 0.5 > 0.16$, darför $\frac{1}{2} \cdot 0$, $\frac{1}{4} = 0.25 : 0.25 > 0.16$, darför $\frac{1}{4} \cdot 0$, $\frac{1}{8} = 0.125 < 0.16$ och darför $\frac{1}{8} \cdot 1$, $\frac{1}{16} = 0.0625 : 0.16 - 0.125 = 0.035, 0.0625 > 0.035$, och darför $\frac{1}{16} \cdot 0$, $\frac{1}{32} = 0.0312 : 0.0312 < 0.035$, och darför $\frac{1}{32} \cdot 1$, och så vidare ...

Example

$-9.28 := -[1.16] \cdot 2^3 = -[1 + 0.16] \cdot 2^3$

Kollar i Matlab:

```
q = quantizer('double');
y = num2bin(q,-9.28)
y =
1100000000100010100011110101110000101000111101011100001010001111
```

Obs: uppe finns inte plats for y på sliden, vi ska ha:

```
y =
1100000000100010100011110101110000101000111101011100001010001111
```


Example

$6.28 = +[1.57] \cdot 2^2 = +[1 + 0.57] \cdot 2^2$

$$0.57 = \frac{1}{2} + 0.07 =$$
$$= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} + 0 \cdot \frac{1}{32} + \ldots + \frac{1}{256} + \ldots$$

Exponenten 2 lagras som: $2 + 1023 = 1025 = 1024 + 1 =$
$1 \cdot 2^{10} + 1 \cdot 2^0$

0	10000000001	10010 ....10...
tecken	exponent 11 bitar	mantissa 52 bitar

I Matlab:

```
y = num2bin(q,6.28)
y =
0100000000011001000111101011100001010001111010111000010100011111
```

Syns inte sista siffror:

```
y =
0100000000011001000111101011100001010001111010111000010100011111
```

Flyttal-räkning

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.

Example

Exempel: Låt oss anta att vi räknar decimalt med fyra siffror.

$$\begin{aligned} fl(\pi) &= fl(3.141592653589...) = 3.142, \\ fl(31415926.53589...) &= 3.142 \cdot 10^7. \end{aligned} \tag{10}$$

Hur stort kan det absoluta felet bli vid avrundning till närmaste flyttal? Maximalt en halv enhet i fjärde siffran. Så om vårt tal är

$$\pm s_1.s_2s_3s_4...10^e$$

(där $s_1, s_2, ...$ betecknar decimala siffror) är absolutbeloppet av absoluta felet maximalt $0.0005 \cdot 10^e$. Relativa felet är maximalt (för ett normaliserat tal)

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

Maskinnoggrannheten ε_{mach}

ε_{mach} beror på method, som vi använder i avrundning. Definitioner för ε_{mach} för precision t:

- I rounding by choping:

$$\varepsilon_{mach} = \beta^{1-p} = \beta^{-t}$$

- I rounding to nearest:

$$\varepsilon_{mach} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}\beta^{-t}$$

Maskinnoggrannheten ε_{mach}

I dubbel precision (när $t = 53$, 64 bits dator) gäller att $\varepsilon_{mach} = 2^{-t} \approx 1.11 \cdot 10^{-16}$ och i enkel precision (när $t = 24$, 32 bits dator) $\varepsilon_{mach} = 2^{-t} \approx 6 \cdot 10^{-8}$.

Example

	chop rounding to 2 digits	to nearest to 2 digits
1.849	1.8	1.8
1.850	1.8	1.9
1.851	1.8	1.9
1.899	1.8	1.9

Example

$1 := [1] \cdot 2^0 = [1 + 0.0] \cdot 2^0$
Mantissa är 0 här.
Exponenten 0 lagras som: $0 + 1023 = 1024 - 1 = 1 \cdot 2^{10} - 1 \cdot 2^0 =$
$\underbrace{10000000000}_{11\ bitar} - \underbrace{00000000001}_{11\ bitar} = \underbrace{01111111111}_{11\ bitar}$.

0	01111111111	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Example

$0.5 := [1] \cdot 2^{-1} = [1 + 0.0] \cdot 2^{-1}$
Mantissa är 0 här.
Exponenten -1 lagras som: $-1 + 1023 = 1024 - 2 =$
$1 \cdot 2^{10} - 1 \cdot 2^{1} = \underbrace{10000000000}_{11\ bitar} - \underbrace{00000000010}_{11\ bitar} = \underbrace{01111111110}_{11\ bitar}.$

0	01111111110	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Example

$0.1 := [1.6] \cdot 2^{-4} = [1 + 0.6] \cdot 2^{-4}$
Exponenten -4 lagras som: $-4 + 1023 = 1019 = 1024 - 5 =$
$1 \cdot 2^{10} - (1 \cdot 2^{2} + 1 \cdot 2^{0}) = \underbrace{10000000000}_{11\ bitar} - \underbrace{00000000101}_{11\ bitar} = \underbrace{01111111011}_{11\ bitar}.$

0	01111111011	100...
tecken	exponent 11 bitar	mantissa 52 bitar

Kollar i Matlab:

```
q = quantizer('double');
y = num2bin(q,0.1)
y =
0011111110111001100110011001100110011001100110011001100110011010
```

Obs: uppe finns inte plats for y på sliden, vi ska ha:
y =
0011111110111001100110011001100110011001100110011001100110011010

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.

$$\begin{aligned} fl(\pi) &= fl(3.141592653589...) = 3.142, \\ fl(31415926.53589...) &= 3.142 \cdot 10^7. \end{aligned} \tag{11}$$

Hur stort kan det absoluta felet bli vid avrundning till närmaste flyttal? Maximalt en halv enhet i fjärde siffran. Så om vårt tal är

$$\pm s_1.s_2s_3s_4...10^e$$

(där $s_1, s_2, ...$ betecknar decimala siffror) är absolutbeloppet av absoluta felet maximalt $0.0005 \cdot 10^e$. Relativa felet är maximalt (för ett normaliserat tal)

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

Denna begränsning kallas relativa maskinnoggrannheten

$\max \varepsilon_{mach} := 0.0005$

Maskinnoggrannheten ε_{mach}

ε_{mach} beror på method, som vi använder i avrundning. Definitioner för ε_{mach} för precision t:

- I rounding by choping:

$$\varepsilon_{mach} = \beta^{1-p} = \beta^{-t}$$

- I rounding to nearest:

$$\varepsilon_{mach} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}\beta^{-t}$$

Maskinnoggrannheten ε_{mach}

I dubbel precision (när $t = 53$, 64 bits dator) gäller att $\varepsilon_{mach} = 2^{-t} \approx 1.11 \cdot 10^{-16}$ och i enkel precision (när $t = 24$, 32 bits dator) $\varepsilon_{mach} = 2^{-t} \approx 6 \cdot 10^{-8}$.

Example

	chop rounding to 2 digits	to nearest to 2 digits
1.849	1.8	1.8
1.850	1.8	1.9
1.851	1.8	1.9
1.899	1.8	1.9

Maskinnoggrannheten ε_{mach}

Now the following values of machine epsilon apply to standard floating point formats:

Table 1. *Values of machine epsilon in standard floating point formats. Notation $*$ means that one bit is implicit in precision p. Machine epsilon ε_{mach} is computed accordingly to Demmel, Applied Numerical Linear Algebra, SIAM.*

EEE 754 - 2008	description	Base, b	Precision, p	Machine eps. $\varepsilon_{mach} = 0.5 \cdot \beta^{-(p-1)}$
binary16	half precision	2	11^*	$2^{-11} = 4.88e-04$
binary32	single precision	2	24^*	$2^{-24} = 5.96e-08$
binary64	double precision	2	53^*	$2^{-53} = 1.11e-16$
binary80	extended precision	2	64	$2^{-64} = 5.42e-20$
binary128	quad. precision	2	113^*	$2^{-113} = 9.63e-35$
decimal32	single prec. decimal	10	7	5×10^{-7}
decimal64	double prec. decimal	10	16	5×10^{-16}
decimal128	quad. prec. decimal	10	34	5×10^{-34}

Flyttal-räkning

Normaliserat tal:

$$\pm s_1.s_2s_3s_4...10^e$$

,
där $s_1, s_2, .., s_1 \neq 0$. betecknar decimala siffror.
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.
Denormaliserat tal i enkel precision (f=fraktion):

$$\pm 0.f \cdot 2^{-126}.$$

Denormaliserat tal i dubbel precision:

$$\pm 0.f \cdot 2^{-1022},$$

Alla icke-zero reella tal kan normaliseras.

Example

$x = 918.082$ i normalizerat form: $9.18082 \cdot 10^2$.
-0.00574012 i normalizerat form: $-5.74012 \cdot 10^{-3}$.

Flyttal-räkning: IEEE Standard 754

Floating Point Range

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

	Denormaliserat	Normaliserat
Enkel prec.	$\pm 2^{-149}$ till $(1-2^{-23}) \cdot 2^{-126}$	$\pm 2^{-126}$ till $(2-2^{-23}) \cdot 2^{127}$
Dubbel prec.	$\pm 2^{-1074}$ till $(1-2^{-52}) \cdot 2^{-1022}$	$\pm 2^{-1022}$ till $(2-2^{-52}) \cdot 2^{1023}$

	Decimal
Enkel prec.	$\pm \approx 10^{-44.85}$ till $\approx 10^{38.53}$
Dubbel prec.	$\pm \approx 10^{-323.3}$ till $\approx 10^{308.3}$

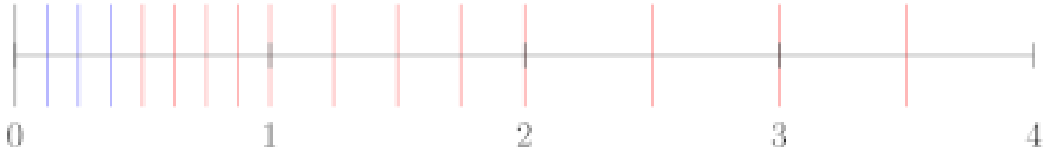
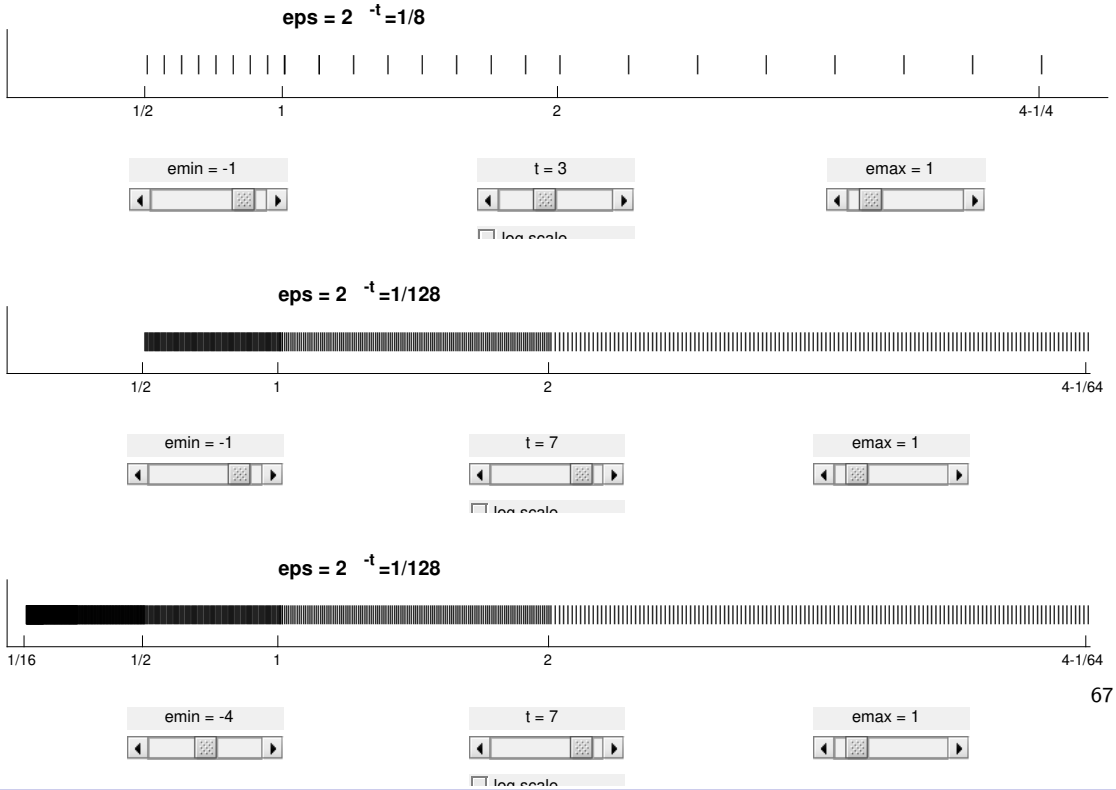


Figure: *Exempel: Normaliserade tal (röd), denormaliserade tal (blå) (Wikipedia)*

Flyttal-räkning: program floatgui.m



Maskinnoggrannheten ε_{mach}

Vi kan skriva

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

på ett annat sätt. Det gäller med $|\varepsilon| \leq \varepsilon_{mach}$ att

$$fl(x) = (1+\varepsilon)x = x + \varepsilon x$$

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

$$fl(x) = (1+\varepsilon)x;$$
$$fl(x) - x = \varepsilon x;$$
$$\underbrace{\left|\frac{fl(x)-x}{x}\right|}_{\leq \varepsilon_{mach}} = |\varepsilon|$$

Flyttal-räkning

Låt $\otimes$ beteckna någon av operationer $+, -, *, /$. Låt x och y vara två flyttal. Då gäller att med $|\varepsilon| \leq \varepsilon_{mach}$:

$$fl(x \otimes y) = (1 + \varepsilon)(x \otimes y) = (x \otimes y) + \varepsilon(x \otimes y)$$

Beloppet av absoluta felet är:

$$|fl(x \otimes y) - (x \otimes y)| = |\varepsilon(x \otimes y)| \leq \varepsilon_{mach}|x \otimes y|$$

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

$$\frac{|fl(x \otimes y) - (x \otimes y)|}{|x \otimes y|} = |\varepsilon| \leq \varepsilon_{mach}.$$

Flyttal-räkning: problem

Några vanliga problem med fluttalsräkning:
Antag att vi räknar i dubbel precision:

1e16 + 1 - 1e16
ans = 0

men om vi skriver

1e16 - 1e16 + 1
ans = 1

Flyttal-räkning: problem

1 + 1e-16 - 1
ans = 0

men

1 -1 + 1e-16
ans = 1.0000e-16

Flyttal-räkning: problem

1e16 + 1.00000000001 - 1e16
ans = 0

men

1e16 - 1e16 + 1.00000000001
ans = 1

Flyttal-räkning: kancellation

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.
Kancellation - subtraktion av två nästan lika stora tal.
Antag att vi subtraherar två olika tal:

1.03678947f
1.03678935g
==========
0.00000012t

Här, f och g betecknar fel, och fi får nytt fel t. I de två första talen kommer felet i tionde siffran: $1.03678947 \underbrace{f}_{10\ siffra}$, och i skillnaden finns felet redan i tredje sifrran: $0.00000012t = 1.2 \underbrace{t}_{3\ siffra} e - 7$. Vi har förlorat information.

Flyttal-räkning: exempel

Antag att a, b, c är redan avrundade flyttal och att vi vill beräkna $a + bc$. Vi får införa en ε-term för varje räkneoperation med $|\varepsilon_k| \leq \varepsilon_{mach}, k = 1, 2$:

$$fl(a + bc) = fl(a + fl(bc)) = (a + bc(1 + \varepsilon_1))(1 + \varepsilon_2).$$

Multiplicera ihop faktorerna och tar beloppet av absoluta felet:

$$fl(a + bc) = a + bc + bc\varepsilon_1 + (a + bc)\varepsilon_2 + bc\varepsilon_1\varepsilon_2,$$
$$|fl(a + bc) - (a + bc)| = |bc\varepsilon_1 + (a + bc)\varepsilon_2 + bc\varepsilon_1\varepsilon_2|.$$

Övre begränsning av det absoluta felet är:

$$|fl(a + bc) - (a + bc)| \leq |bc|\varepsilon_{mach} + |a + bc|\varepsilon_{mach} + |bc|\varepsilon_{mach}^2|$$
$$= ((1 + \varepsilon_{mach})|bc| + |a + bc|)\varepsilon_{mach}$$

Flyttal-räkning: exempel

För relativa felet har vi (om $a + bc \neq 0$):

$$\frac{|fl(a+bc)-(a+bc)|}{|a+bc|} \leq \frac{(1+\varepsilon_{mach})|bc|+|a+bc|}{|a+bc|}\varepsilon_{mach}$$
$$= \left[\frac{(1+\varepsilon_{mach})|bc|}{|a+bc|}+1\right]\varepsilon_{mach}.$$

Relativa felet är litet om $\frac{|bc|}{|a+bc|}$ inte är för stort. Om, till exempel, $|bc| \approx 1, |a+bc| \approx 0$, vi få ett stort relativt fel.
Om man inte kräver en strikt gräns utan endast en uppskattning kan man tillåta sig att slänga t.ex. $\varepsilon_1\varepsilon_2$- termer (produkter av termer) ty $\varepsilon_{mach}^2 << \varepsilon_{mach}$.

Flyttal-räkning: exempel

För att se att analysen stämmer rätt bra kommer här ett numeriskt exempel i fyrsiffrig decimal aritmetik:

$$a = 10.70,\ b = -4.567,\ c = 2.344,\ a+bc = -0.005048\ (exakt).$$

$$fl(a+bc) = fl(a+fl(bc)),$$
$$fl(bc) = -10.71\ (\text{eftersom}\ bc = -10.705048),$$
$$fl(a+fl(bc)) = fl(10.70+(-10.71)) = -0.010.$$

Beloppet av absoluta felet är:

$$a+bc = -0.005048\ \text{exakt},$$
$$|fl(a+bc)-(a+bc)| = |-0.010-(-0.005048)| = 0.004952.$$

Notera att detta fel är ungefär lika stort som det exakta värdet. Det relativa felet är:

$$\frac{|fl(a+bc)-(a+bc)|}{|a+bc|} = \frac{|-0.010-(-0.005048)|}{|-0.005048|} \approx 0.98.$$

Flyttal-räkning: exempel

Det relativa felet är:

$$\frac{|fl(a+bc)-(a+bc)|}{|a+bc|} = \frac{|-0.010-(-0.005048)|}{|-0.005048|} \approx 0.98$$

$$\leq \underbrace{\left[\frac{(1+\varepsilon_{mach})|bc|}{|a+bc|}+1\right]\varepsilon_{mach}}_{\text{uppskattning för felet}}$$

$$= \left[\frac{(1+0.0005)|\cdot|-10.705048|}{|-0.005048|}+1\right]0.0005 \approx 1.061$$

Här, relativa maskinnoggrannheten $\max \varepsilon_{mach} := 0.0005$.
Resultat stämmer väl med uppskattning, vi fick: $0.98 \leq 1.061$.

Matrisfaktoriseringar: LU-faktorisering

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

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

- L för "Lower triangular", undertriangulär och
- U för "Upper triangular", övertriangulär.
- Kallas LU-faktorisering. Används för att lösa Ax = b-problem.
- Matlab-kommando **lu**.

Matrisfaktoriseringar: QR-faktorisering

För att approximativt lösa överbestämda ekvationssystem (minstakvadratproblem) använder vi QR-faktorisering:

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

- $\dim A = m \times n$, $\dim Q = m \times n$, $\dim R = n \times n$.
- Q för ortogonal matris, dvs. $Q^T Q = I_n$.
- R för "Upper triangular", övertriangulär.
- Kallas QR-faktorisering. Används för att lösa minstakvadratproblem, hitta egenvärden i symmetrisk matris.
- Matlab-kommando **qr**.

Matrisfaktoriseringar: diagonalisering

Om A är en s.k. diagonaliserbar matris kan vi använda Matlabs **eig**-kommando för att beräkna:

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

$\lambda_1, \lambda_2, \lambda_3$ 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 $X^{-1} = X^T$.

Singulära värdena

Låt A vara en $m \times n$ matris och låt $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ vara egenvärdena till A^*A ordnade i storleksordning. Talen $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ definierade genom $\sigma_j = \sqrt{\lambda_j(A^*A)}$ kallas de singulära värdena till A.
Conjugate transpose matrix:

$$A^*_{ij} = \overline{A}_{ji},$$

$\overline{A}_{ji}$ - scalar complex conjugate elements.

Example

$$\mathbf{A} = \begin{bmatrix} 3+i & 5 & -2i \\ 2-2i & i & -7-13i \end{bmatrix}$$

då

$$\mathbf{A}^* = \begin{bmatrix} 3-i & 2+2i \\ 5 & -i \\ 2i & -7+13i \end{bmatrix}$$

Singuläravärdesfaktoriseringen (SVD)

Singuläravärdesfaktoriseringen (i Matlab commando **svd**) är en slags generalisering av egenvärdesuppdelningen av $A = U\Sigma V^T$ till ickekvadratiska matriser.

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

där $U^TU = I, V^TV = I$ är ortogonala matriser och $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Faktoriseringen existerar även för liggande matriser. Används för att lösa minstakvadratproblem, hitta egenvärden i symmetrisk matris, komprimera bilder.

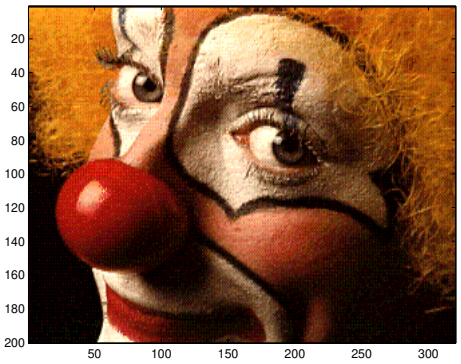
Example of application of linear systems: image compression using SVD

Definition SVD *Let A be an arbitrary m-by-n matrix with $m \geq n$. Then we can write $A = U\Sigma V^T$, where U is m-by-n and satisfies $U^T U = I$, V is n-by-n and satisfies $V^T V = I$, and $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The columns $u_1, \ldots, u_n$ of U are called $left\ singular\ vectors$. The columns $v_1, \ldots, v_n$ of V are called $right\ singular\ vectors$. The σ_i are called $singular\ values$. (If $m < n$, the SVD is defined by considering A^T.)*

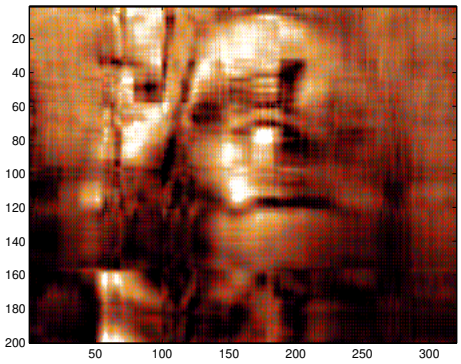
Theorem

Write $V = [v_1, v_2, \ldots, v_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $||\cdot||_2$ is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and $||A - A_k||_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

Application SVD: Image compression using SVD



a) Original image



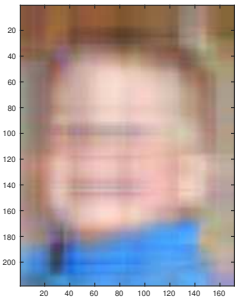
b) Rank k=20 approximation

Image compression using SVD in Matlab

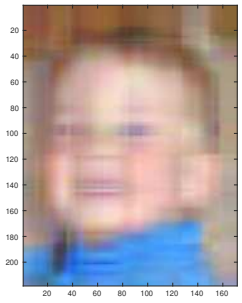
See path for other pictures:
/matlab-2012b/toolbox/matlab/demos
load clown.mat;
Size(X) $= m \times n = 320 \times 200$ pixels.
[U,S,V] = svd(X);
colormap(map);
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)$= m \times k$, size(V)$= n \times k$.

Image compression using SVD in Matlab

a) Original image



b) Rank k=4 approximation



b) Rank k=5 approximation

c) Rank k=6 approximation

d) Rank k=10 approximation

d) Rank k=15 approximation

Image compression using SVD for arbitrary picture

To get image on the previous slide, I took picture in jpg-format and loaded it in matlab. You can also try to use following matlab code for your own pictures:

```
A = imread('Child.jpg');
Real size of A: size(A) ans= 218 171 3
figure(1); image(DDA);
DDA=im2double(A);
[U1,S1,V1] = svd(DDA(:,:,1)); [U2,S2,V2] = svd(DDA(:,:,2));
[U3,S3,V3] = svd(DDA(:,:,3));
k=15;
svd1 = U1(:,1:k)*S1(1:k,1:k)*V1(:,1:k)';
svd2 = U2(:,1:k)*S2(1:k,1:k)*V2(:,1:k)';
svd3 = U3(:,1:k)*S3(1:k,1:k)*V3(:,1:k)';
DDAnew = zeros(size(DDA));
DDAnew(:,:,1) = svd1; DDAnew(:,:,2) = svd2; DDAnew(:,:,3) = svd3;
figure(2); image(DDAnew);
```


Matrisfaktoriseringar: LU-faktorisering

$Ax = b$ löses i de tre stegen:

1. Beräkna L (undertriangular matris) och U (övertriangular matris) 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 $\cdot$.
- $Lz = b$ kostar $n^2/2$ vardera av $+$ och $\cdot$.
- $Ux = z$ kostar lika mycket ($n^2/2$).

LU-faktorisering för linjära ekvationssystem

Vad är det för fördel med detta jämfört med vanlig Gausselimination ?
Svar: enklare att hantera vid teoretiskt arbete.
Det gör det också möjligt att effektivt 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 Gausseliminationutnyttjas.
Man löser ett sådant problem så här:

- Beräkna L och U så att $A = LU$.
- Lös $LUx_k = b_k, k = 1, 2, ...$

Gaussian Elimination

The Algorithm — uniqueness of factorization

Definition
The leading j-by-j principal submatrix of A is $A(1 : j, 1 : j)$.

Theorem 2.4.
The following two statements are equivalent:
1. There exists a unique unit lower triangular L and non-singular upper triangular U such that $A = LU$.
2. All leading principal submatrices of A are non-singular.

Gaussian Elimination

The Algorithm — uniqueness of factorization

Bevis.

We first show that (1) implies (2). $A = LU$ may also be written

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \times \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix}$$

where A_{11} is a j-by-j leading principal submatrix, as well as L_{11} and U_{11}. Therefore
$detA_{11} = det(L_{11}U_{11}) = detL_{11}detU_{11} = 1 \cdot \prod_{k=1}^{j}(U_{11})_{kk} \neq 0$,
since L is unit triangular and U is triangular.

Gaussian Elimination

The Algorithm — uniqueness of factorization

Bevis.

(2) implies (1) is proved by induction on n. It is easy for 1-by-1 matrices: $a = 1 \cdot a$. To prove it for n-by-n matrices $\tilde{A}$, we need to find unique (n-1)-by-(n-1) triangular matrices L and U, unique (n-1)-by-1 vectors l and u, and unique nonzero scalar η such that

$$\tilde{A} = \begin{bmatrix} A & b \\ c^T & \delta \end{bmatrix} = \begin{bmatrix} L & 0 \\ l^T & 1 \end{bmatrix} \times \begin{bmatrix} U & u \\ 0 & \eta \end{bmatrix} = \begin{bmatrix} LU & Lu \\ l^T U & l^T u + \eta \end{bmatrix}$$

By induction unique L and U exist such that $A = LU$. Now let $u = L^{-1}b$, $l^T = c^T U^{-1}$, and $\eta = \delta - l^T u$, all of which are unique. The diagonal entries of U are nonzero by induction, and $\eta \neq 0$ since $0 \neq det\tilde{A} = det(U) \cdot \eta$.

□

Övning: räkna LU-faktorisering

Example

$$A = \begin{bmatrix} 2 & 6 \\ 4 & 15 \end{bmatrix}; \quad L-? \quad U-? \quad A = LU$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11}\ell_{11} & \ell_{11}u_{12} \\ \ell_{21}u_{11} & \ell_{21} \cdot u_{12} + \ell_{22} \cdot u_{22} \end{bmatrix}$$

$$\ell_{11} \cdot u_{11} = a_{11} \Rightarrow L = \begin{bmatrix} 1 & 0 \\ \ell_{21} & 1 \end{bmatrix}$$

Example

$$\Rightarrow u_{11} = \frac{a_{11}}{\ell_{11}} = a_{11}$$

$$\ell_{11} \cdot u_{12} = a_{12} \Rightarrow u_{12} = a_{12}$$

$$\ell_{21} \cdot u_{11} = a_{21} \Rightarrow \ell_{21} = \frac{a_{21}}{u_{11}} = \frac{a_{21}}{a_{11}} = \frac{4}{2} = 2$$

$$\ell_{21} \cdot u_{12} + \ell_{22} \cdot u_{22} = a_{22} \Rightarrow 2 \cdot a_{12} + 1 \cdot u_{22} = a_{22} \Rightarrow$$

$$u_{22} = a_{22} - 2 \cdot a_{12} = 15 - 2 \cdot 6 = 3$$

$$\underbrace{\begin{bmatrix} 2 & 6 \\ 4 & 15 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix}}_{U}$$

Är LU-faktorisering 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:

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

a_1/ε blir ett stort tal, vilket ger utskiftning i beräkningen av $u_{22} = a_3 - a_2(a_1/\varepsilon)$. Låt oss anta att hela a_3 skiftas ut och att allt annat räknas ut exakt. Hur stort blir bakåtfelet?

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

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

Är LU-faktorisering en stabil algoritm?

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

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

LU-faktorisering blir nu:

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

Notera att ε/a_1 är ett litet tal. Vi får alltså inte farlig utskiftning i $u_{2,2}$. Låt oss anta att $a_3(\varepsilon/a_1)$ skiftas ut:

$$\begin{bmatrix} 1 & 0 \\ \varepsilon/a_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_3 \\ 0 & a_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_3 \\ \varepsilon & a_2 + a_3\varepsilon/a_1 \end{bmatrix} = B + \begin{bmatrix} 0 & 0 \\ 0 & a_3\varepsilon/a_1 \end{bmatrix}$$

Gaussian elimination

The basic algorithm for solving $Ax = b$.

1. Permutation matrices.
2. The algorithm - overview.
3. The algorithm - factorization with pivoting.

Permutation matrices

Definition

Permutation matrix := identity matrix with permuted rows.

Example

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \rightarrow \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

Permutation matrices

Properties

Properties of permutation matrices (P, P_1, P_2):

- $P \cdot X =$ same matrix X with rows permuted
- $P_1 \cdot P_2$ is also a permutation
- $P^{-1} = P^T$ (reverse permutation)
- $\det(P) = \pm 1$ (+1 for even permutations, -1 for odd)

Gaussian Elimination

The Algorithm — Overview

Solving $Ax = b$ using Gaussian elimination.

1. Factorize A into $A = PLU$
   Permutation Unit lower triangular Non-singular upper triangular
2. Solve $PLUx = b$ (for LUx) :
$$LUx = P^{-1}b$$
3. Solve $LUx = P^{-1}b$ (for Ux) by forward substitution:
$$Ux = L^{-1}(P^{-1}b).$$
4. Solve $Ux = L^{-1}(P^{-1}b)$ by backward substitution:
$$x = U^{-1}(L^{-1}P^{-1}b).$$

Gaussian Elimination

LU factorization with pivoting: calculating the permutation matrix P, the unit lower triangular matrix L, and the nonsingular upper triangular matrix U such that $LU = PA$ for a given nonsingular A.

let $P = I$, $L = I$, $U = A$
for $i = 1$ to $n - 1$
find m such that $|U(m, i)|$ is the largest entry in $|U(i : n, i)|$
if $m \neq i$
swap rows m and i in P
swap rows m and i in U
if $i \geq 2$ swap elements $L(m, 1 : i - 1)$ and $L(i, 1 : i - 1)$
end if
$L(i + 1 : n, i) = U(i + 1 : n, i)/U(i, i)$
$U(i+1 : n, i+1 : n) = U(i+1 : n, i+1 : n) - L(i+1 : n, i)\ U(i, i+1 : n)$
$U(i + 1 : n, i) = 0$
end for

Forward substitution

The next algorithm is *forward substitution*. We use it to easily solve a given system $Lx = b$ with a unit lower triangular matrix L.

Forward substitution: solving $Lx = b$ with a unit lower triangular matrix L.

$x(1) = b(1)$
for $i = 2$ to n
    $x(i) = b(i) - L(i, 1 : (i - 1))\ x(1 : (i - 1))$
end for

Backward substitution

Using Backward substitution, we easily solve a given system $Ux = b$ with an upper triangular matrix U.

Backward substitution: solving $Ux = b$ with a nonsingular upper triangular matrix U.

$x(n) = b(n)/U(n,n)$
for $i = n-1$ to 1
  $x(i) = (b(i) - U(i,(i+1):n)\ x((i+1):n))/U(i,i)$
end for

LDU-faktorisering

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

Definition

LDU-faktoriseringen

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

Example

Här ett exempel där vi struntar i pivotering för att slippa bråk.

$$\underbrace{\begin{bmatrix} 2 & 6 \\ 4 & 15 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix}}_{U_0} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}}_{U}$$

Så allmänt A = LDU. Vi kan utnyttja detta för att titta på två viktiga fall:

LDU-faktorisering

1. A är symmetrisk matris: $A = A^T$, då $U = DL^T$ så att $A = LDL^T$. Innebär halva antalet operationer för faktoriseringen (förutsatt att vi utnyttjar symmetrin i vår algoritm). Halverat minnesbehov.

$$\underbrace{\begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 4 \\ 0 & -3 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}}_{L^T}$$

   Problem med pivotering och symmetri ty partiell pivotering förstör symmetrin (finns andra pivoteringsalgoritmer).
2. Det andra viktiga fallet inträffar när D i $A = LDL^T$ har positiva diagonalelement.

Matrisfaktoriseringar: DU-faktorisering

$$A = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} \Rightarrow A = DU, \quad D - \text{diagonal matrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} = \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix}$$

$$d_{11} \cdot 1 = a_{11} \Rightarrow d_{11} = a_{11}$$
$$d_{11} \cdot u_{12} = a_{12} \Rightarrow a_{11} \cdot u_{12} = a_{12} \Rightarrow u_{12} = \frac{a_{12}}{a_{11}}$$
$$d_{22} \cdot 1 = a_{22}$$

Övning: DU-faktorisering

Example

Räkna DU-faktorisering för

$$A = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix}$$

$$d_{11} = 2;\ d_{22} = a_{22} = 3;\ u_{12} = \frac{a_{12}}{a_{11}} = \frac{6}{2} = 3$$

$$A = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D} \cdot \underbrace{\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}}_{U}$$

Example: Choleskyfaktorisering

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

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

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

Choleskyfaktorisering

$A = CC^T$ kallas Choleskyfaktorisering och den existerar när A är symmetrisk $A = A^T$ och positivt definit:

$$x \neq 0,\ x^T Ax > 0.$$

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.

Example

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

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

$= V^T MV/2$.
$E_{kin} > 0$ om någon massa rör sig, dvs. om $V \neq 0$ och $V^T MV > 0$ så att M är positivt definit.

Theorem

En symmetrisk, positivt definit (s.p.d) matris har positiva egenvärden. Omvändningen gäller också: en reell, symmetrisk matris A är positivt definit om den har positiva egenvärden.

Bevis.

En reell och symmetrisk matris A har reella egenvärden och egenvektorer.

$$Ax = \lambda x$$

då

$$x^T Ax = \lambda x^T x$$

och

$$\lambda = \frac{x^T Ax}{x^T x} > 0$$

ty $x^T x = \sum_{k=1}^{n} x_k^2 > 0$ $k > 0$ eftersom x inte är nollvektorn.

□

Matrisfaktoriseringar: Choleskyfaktorisering

Choleskyfaktorisering för s.p.d. A är $A = L \cdot L^T$, var L är undertriangular matris.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{bmatrix} \cdot \begin{bmatrix} \ell_{11} & \ell_{21} \\ 0 & \ell_{22} \end{bmatrix} = \begin{bmatrix} \ell_{11}^2; & \ell_{11} \cdot \ell_{21} \\ \ell_{21} \cdot \ell_{11}; & \ell_{21}^2 + \ell_{22}^2 \end{bmatrix}.$$

$\ell_{11} = 1, \quad \ell_{21} = a_{21};$

$$\ell_{21}^2 + \ell_{22}^2 = a_{22}$$

$$a_{21}^2 + \ell_{22}^2 = a_{22}$$

$$\ell_{22} = \sqrt{a_{22} - a_{21}^2} > 0$$

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ a_{21} & \ell_{22} \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} 1 & a_{21} \\ 0 & \ell_{22} \end{bmatrix}}_{L^T}$$

Matrisfaktoriseringar: Choleskyfaktorisering $A = L \cdot L^T$

$$A = \underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} \ell_{11} & \ell_{21} \\ 0 & \ell_{22} \end{bmatrix}}_{L^T} = \begin{bmatrix} \ell_{11}^2; & \ell_{11} \cdot \ell_{21} \\ \ell_{21} \cdot \ell_{11}; & \ell_{21}^2 + \ell_{22}^2 \end{bmatrix}.$$

$$\ell_{11}^2 = a_{11} \Rightarrow \ell_{11} = \sqrt{a_{11}}$$
$$\ell_{11} \cdot \ell_{21} = a_{12} \Rightarrow \ell_{21} = \frac{a_{12}}{\ell_{11}} = \frac{a_{12}}{\sqrt{a_{11}}}$$
$$\ell_{21} \cdot \ell_{21} + \ell_{22} \cdot \ell_{22} = a_{22}$$
$$\ell_{21}^2 + \ell_{22}^2 = a_{22} \Rightarrow \ell_{22} = \sqrt{a_{22} - \ell_{21}^2} = \sqrt{a_{22} - \frac{a_{12}^2}{a_{11}}};$$

Example

Räkna Choleskyfaktorisering $A = L \cdot L^T$ för

$$A = \begin{bmatrix} 4 & 8 \\ 8 & 25 \end{bmatrix};$$

Är A s.p.d. ? Räknar egenvärden:

$$\begin{matrix} \lambda_1 \approx 1.3 > 0 \\ \lambda_2 \approx 27.7 > 0 \end{matrix} \Rightarrow$$

$$\ell_{11} = \sqrt{a_{11}} = \sqrt{4} = 2;\ \ell_{21} = \frac{a_{12}}{\sqrt{a_{11}}} = \frac{8}{2} = 4;\ \ell_{22} = \sqrt{25 - \frac{8^2}{4}} = 3.$$

$$A = L \cdot L^T = \begin{bmatrix} 2 & 0 \\ 4 & 3 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 \\ 0 & 3 \end{bmatrix}$$

Cholesky algorithm:

for $j = 1$ *to* n
  $l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{1/2}$
  for $i = j + 1$ *to* n
    $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}$
  end for
end for

Here, $dim A = n$. If A is not positive definite, then (in exact arithmetic) this algorithm will fail by attempting to compute the square root of a negative number or by dividing by zero; this is the cheapest way to test if a symmetric matrix is positive definite.

I Matlab använder vi $chol(A)$.

Tillämpningar av LU, LDL^T, DU, Cholesky faktoriseringar

1) Lösa system av linjära ekvationer $Ax = b$;
2) För att räkna inversa matriser.

$$A = LU \Rightarrow A^{-1} = (LU)^{-1} \Rightarrow A^{-1} = U^{-1}L^{-1}$$
$$A = DU \Rightarrow A^{-1} = (DU)^{-1} \Rightarrow A^{-1} = U^{-1}D^{-1}$$
$$A = LL^T \Rightarrow A^{-1} = (LL^T)^{-1} \Rightarrow A^{-1} = L^{-1}(L^T)^{-1} = L^{-1}(L^{-1})^T$$

3) För att beräkna determinant av $dim(A) = n \times n$:

$$A = LU \Rightarrow det(A) = det(LU) = det(L) \cdot det(U) = \prod_{i=1}^{n} l_{ii} \prod_{i=1}^{n} u_{ii},$$
$$A = PLU \Rightarrow det(A) = det(P)det(L) \cdot det(U) = (-1)^s \prod_{i=1}^{n} l_{ii} \prod_{i=1}^{n} u_{ii},$$
$$A = LL^T \Rightarrow det(A) = det(LL^T) = det(L) \cdot det(L^T) = (det(L))^2.$$

s = hur många gånger var gjort permutation.

Beräkning av inversa matris med hjälp av *DU* faktorisering

Example

$$A = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix}$$

$$d_{11} = 2;\ d_{22} = a_{22} = 3;\ u_{12} = \frac{a_{12}}{a_{11}} = \frac{6}{2} = 3$$

$$A = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D} \cdot \underbrace{\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}}_{U}$$

$$\mathrm{A}^{-1} = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}^{-1} =$$

$$\begin{bmatrix} 1/2 & 0 \\ 0 & 1/3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}^{-1}$$

Hur ska vi räkna inversen på U ?

Example

Observera, att

$$\mathrm{U} = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 3 \\ 0 & 0 \end{bmatrix}}_{N}$$

Vi ska använda formula : $(I+N)^{-1} = I + \sum_{k=1}^{n-1}(-1)^k N^k$.
Härledning: N är triangular med 0 på diagonalen, då $N^n = 0$. Från polynomial faktorizering: $1 - x^n = (1-x)(1 + x + x^2 + ... + x^{n-1})$. För $x = -N$ vi har:
$(I+N)(I - N + N^2 - N^3 + ... + (-1)^{n-1}N^{n-1}) = (I - N^n) = I$ och
$(I+N)^{-1} = I + \sum_{k=1}^{n-1}(-1)^k N^k$.

$$U^{-1} = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \sum_{k=1}^{2-1}(-1)^k N^k$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + (-N) = \begin{bmatrix} 1 & -3 \\ 0 & 1 \end{bmatrix}$$

Example

Vi kan beräkna nu A^{-1}

$$A^{-1} = \begin{bmatrix} 2 & 6 \\ 0 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/3 \end{bmatrix} \cdot \begin{bmatrix} 1 & -3 \\ 0 & 1 \end{bmatrix}$$

Positivt definit matris

Positivt definit matris: $x^T Ax > 0 \quad \forall x \neq 0$.
Exempel: en symmetrisk, positivt definit matris har positiva egenvärden.
Bevis: En reell och symmetrisk matris har reella egenvärden och egenvektorer

$$Ax = \lambda x \Rightarrow x^T Ax = \lambda x^T x \Rightarrow \lambda = \frac{x^T Ax}{x^T x} > 0,$$

och $x^T x > 0, x \neq 0$. Omvändningen gäller också: en reell, symmetrisk matris är positivt definit om den har positiva egenvärden.

Positivt definit matris

Example

En positivt definit matris har positiva diagonalelement. Tag $x = e_j$, kolonn j i I, enhetsmatrisen. Då är (med Matlabnotation)

$$Ae_j = A(:, j), e_j^T A = A(j, :), e_j^T Ae_k = A(j, k) = a_{j,k}$$

Så

$$e_j^T Ae_j = a_{j,j} > 0, j = 1, ..., n$$

Observera implikationen. Positiva diagonalelement är nödvändigt för att vi skall ha en positivt definit matris. Det är inte ett tillräckligt villkor.

Example

$$[1, -1] \cdot \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -2 < 0.$$

Matrisen är indefinit med egenvärden -1 och 3.

Diagonalelementen måste också vara tillräckligt stora jämfört med de utomdiagonala, för att matrisen skall vara positivt definit.

Positivt definit matris

Example

Låt A vara symmetrisk och positiv definit matris, dim $A = n \times n$. Använd definitionen på positivt definit för att bevisa att A är ickesingulär.
Svar:
Definition av s.p.d. matris är: $\forall x \neq 0 \;\; x^T Ax > 0$. Om A vore singulär ($detA = 0$)då skulle det existera $x \neq 0$ så att $Ax = 0$, men det strider mot antagande att A är positivt definit. Exempel:

$$\underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \begin{bmatrix} x_1 + x_2 \\ x_1 + x_2 \end{bmatrix}$$

Vi kan välja $x_1 = -x_2$, då $Ax = 0$.

Positivt definit matris

Example

Ett exempel från flervariabelkursen. Låt $z = f(x, y)$ vara en reellvärd funktion av två variabler. Vi vill undersöka om f har ett strängt lokalt minimum i punkten (a, b). Om f är tillräckligt snäll (har tillräckligt många kontinuerliga derivator) gäller att:

$$f(a+h, b+k) = f(a,b) + f'_x(a,b)h + f'_y(a,b)k + \frac{f''_{xx}(a,b)h^2 + 2f''_{xy}(a,b)hk + f''_{yy}(a,b)k^2}{2} + \ldots \tag{12}$$

Ett nödvändigt villkor för minimum är att gradienten är nollvektorn, ty annars kan vi göra f mindre genom att gå i negativa gradientens riktning. Alltså gäller:

$$f(a+h, b+k) = f(a,b) + \frac{f''_{xx}(a,b)h^2 + 2f''_{xy}(a,b)hk + f''_{yy}(a,b)k^2}{2} + \ldots \tag{13}$$

Positivt definit matris

Nu är (där vi inte skriver ut (a, b))

$$f''_{xx}h^2 + 2f''_{xy}hk + f''_{yy}k^2 = f''_{xx}h^2 + f''_{xy}hk + f''_{yx}hk + f''_{yy}k^2 = [h,\ k] \cdot \begin{bmatrix} f''_{xx} & f''_{xy} \\ f''_{yx} & f''_{yy} \end{bmatrix} \cdot \begin{bmatrix} h \\ k \end{bmatrix} = v^T H v \tag{14}$$

med

$$v = \begin{bmatrix} h \\ k \end{bmatrix}, H = \begin{bmatrix} f''_{xx} & f''_{xy} \\ f''_{yx} & f''_{yy} \end{bmatrix}.$$

H är den s.k. Hessianen. Om H är positivt definit så har f ett strängt lokalt minimum i (a, b).
Detta gäller allmänt. Om $w = f(x, y, z)$ där $\nabla f(a, b, c)$ är nollvektorn, så har f ett strängt lokalt minimum i (a, b, c) om

$$H = \begin{bmatrix} f''_{xx} & f''_{xy} & f''_{xz} \\ f''_{yx} & f''_{yy} & f''_{yz} \\ f''_{zx} & f''_{zy} & f''_{zz} \end{bmatrix}.$$

är positivt definit (alla derivator är beräknade i (a, b, c)).

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.

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

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

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

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

$$B = \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix} \Rightarrow y - x = B^{-1}f = \begin{bmatrix} 0.5f_1 \\ 10f_2 \end{bmatrix}$$

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

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:

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

- 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.

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

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.

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1+\delta \end{bmatrix} \Rightarrow C^{-1} = \frac{1}{\delta}\begin{bmatrix} 1+\delta & -1 \\ -1 & 1 \end{bmatrix}$$

- Vi ser att C^{-1} är proportionell mot $1/\delta$, så inversen är stor. C ligger också nära en singulär matris. Om vi subtraherar δ från $c_{2,2}$ så blir matrisen singulär ($detC \approx 0$ för $\delta \approx 0$). Detta gäller allmänt.
- Om C har element av storleksordningen ett $\Rightarrow$ 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 $\|\cdot\|_p$:

$$\|x\|_p = (\sum_{k=1}^{n} |x_k|^p)^{1/p}, \quad p \geq 1.$$

- $p = 1, \|x\|_1 = \sum_{k=1}^{n} |x_k|$, ettnormen
- $p = 2, \|x\|_2 = (\sum_{k=1}^{n} |x_k|^2)^{1/2}$, tvånormen
- $p = \infty, \|x\|_\infty = \max_{1 \leq k \leq n} |x_k|$, maxnormen

Vektornormer

Dessa tre normer (liksom alla vektornormer) uppfyller:

- $x \neq 0 \rightarrow \|x\| > 0$ (positivitet), $\|0\| = 0$.
- $\|\alpha x\| = |\alpha|\|x\|$ for alla $\alpha \in \mathbb{R}$ (homogenitet)
- $\|x + y\| \leq \|x\| + \|y\|$ (triangelolikheten)

Normer är olika stora, men de är ekvivalenta ("jämförbara"). Det existerar positiva tal α och β som inte beror på x utan bara på n så att t.ex.

$$\alpha\|x\|_1 \leq \|x\|_2 \leq \beta\|x\|_1$$

I detta fall är $\alpha = 1/\sqrt{n}, \beta = 1$. Olikheterna är skarpa, det existerar $x \neq 0$ där likhet antas. 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.

Vektornormer

Normer är olika stora, men de är ekvivalenta ("jämförbara").

Example

- $$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\,\|x\|_2$$
- $$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\,\|x\|_\infty$$
- $$\|x\|_\infty \leq \|x\|_1 \leq n\,\|x\|_\infty\,,$$
- $$\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\,\|x\|_2 \leq n\,\|x\|_\infty$$

Innerprodukt

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

$x \cdot y = (x, y) = \sum_{k=1}^{n} x_k y_k = x^T y, \quad \|x\|_2 = \sqrt{x^T x}.$
Notera att $x^T y$ är en skalär men xy^T är en matris.

Example

$$x^T y = [-1,\ 2,\ 3] \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = (-1) \cdot (3) + 2 \cdot 2 + 3 \cdot 1 = 4.$$

$$xy^T = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} \cdot [3,\ 2,\ 1] = \begin{bmatrix} -3 & -2 & -1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Vektornormer

Example

$$x = \begin{bmatrix} -1 \\ 2 \\ 3 \\ -5 \end{bmatrix}; \qquad \begin{aligned} ||x||_1 &= |-1| + |2| + |3| + |-5| = 11 \\ ||x||_2 &= \sqrt{(-1)^2 + 2^2 + 3^2 + (-5)^2} = \sqrt{39} \end{aligned}$$

$||x||_\infty = max(|-1|, |2|, |3|, |-5|) = 5.$
En vektor x är normerad om $||x|| = 1$. Om $x \neq 0$ så är $\dfrac{x}{||x||}$ normerad.

$$x^T x = [-1,\ 2,\ 3,\ -5] \cdot \begin{bmatrix} -1 \\ 2 \\ 3 \\ -5 \end{bmatrix} = (-1) \cdot (-1) + 2 \cdot 2 + 3 \cdot 3 + (-5)^2 = 39$$

$$V = \frac{x}{||x||} = \left[\frac{-1}{\sqrt{39}}, \frac{2}{\sqrt{39}}, \frac{3}{\sqrt{39}}, \frac{-5}{\sqrt{39}} \right]^T \implies ||V||_2 = 1$$

Matrisnormer

Matrisnormer är funktioner från $\mathbb{R}^{m\times n}$ till $\mathbb{R}$, eller $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ och uppfyller de tre vektornormsvillkoren ovan, eller:

- $\|A\| \geq 0$ (positivitet).
- $\|\alpha A\| = |\alpha|\|A\|$ for alla $\alpha \in \mathbb{R}$ (homogenitet)
- $\|A + B\| \leq \|A\| + \|B\|$ (subadditivitet, triangelolikheten)

Om $A, B \in \mathbb{R}^{n\times n}$ användbara matrisnormer är submultiplikativa:

- $\|AB\| \leq \|A\| \cdot \|B\|$ (submultiplikativitet)

Normer är olika stora, men de är ekvivalenta ("jämförbara"). Det existerar positiva tal α och β som inte beror på A att t.ex.

$$\alpha\|A\|_1 \leq \|A\|_2 \leq \beta\|A\|_1$$

Matrisnormer

Vi kan bilda matrisnormer utgående från vektornormer. En operatornorm 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:

- $p = 1$, ettnormen, största kolonnsumman

$$||A||_1 = \max_k \sum_{r=1}^{m} |a_{r,k}|$$

- $p = 2$, tvånormen,

$$||A||_2 = \max \sqrt{\lambda(A^T A)}$$

- $p = \infty$, maxnormen, största radsumman

$$||A||_\infty = \max_r \sum_{k=1}^{n} |a_{r,k}|$$

Matrisnormer

Example

För $A \in \mathbb{R}^{m \times n}$ gäller:

- $\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty$
- $\frac{1}{\sqrt{m}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$.
- $\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}$.

Matrisnormer

Example

$$A = \begin{bmatrix} 1 & -2 & -3 \\ 6 & 4 & 2 \\ 9 & -6 & 3 \end{bmatrix}, \quad \begin{aligned} ||A||_1 &= \max(|1| + |6| + |9|, |-2| + |4| + \\ &+ |-6|, |-3| + |2| + |3|) = \\ &= \max(16, 12, 8) = 16 \end{aligned}$$

$$\begin{aligned} ||A||_\infty &= \max(|1| + |-2| + |-3|, |6| + |4| + |2|, \\ &|9| + |-6| + |3|) = \max(6, 12, 18) = 18 \end{aligned}$$

Matrisnormer

Example

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

$||A||_2 = \max \sqrt{\lambda(A^T A)}$

$$A^T = \begin{bmatrix} 1 & 6 & 9 \\ -2 & 4 & -6 \\ -3 & 2 & 3 \end{bmatrix}, \; A^T A = \begin{bmatrix} 118 & -32 & 36 \\ -32 & 56 & -4 \\ 36 & -4 & 22 \end{bmatrix}$$

$$\lambda(A^T A) = \begin{bmatrix} 8.9683 \\ 45.3229 \\ 141.7089 \end{bmatrix}; \quad \max \sqrt{\lambda(A^T A)} = \max(2.9947, 6.7322, 11.9042) = 11.9042$$

Matrisnormer

Example

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \; A^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \; A^T A - \lambda I = \begin{bmatrix} 1-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} = 0;$$

$\lambda_1 = 1, \; \lambda_2 = 1; \; ||A||_2 = \max \sqrt{\lambda(A^T A)} = \max(1, 1) = 1$

Matrisnormer

Example

Beräkna $\|A\|_2$:

$$A = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}$$

Svar: $\|A\|_2 := \max \sqrt{\lambda(A^T A)}$.

$$A^T = \begin{bmatrix} 3 & 1 \\ 0 & 5 \end{bmatrix}, \quad A^T A = \begin{bmatrix} 10 & 5 \\ 5 & 25 \end{bmatrix},$$

och för att hitta $\lambda(A^T A)$ vi ska lösa ekvation $\lambda^2 - 35\lambda + 225 = 0$.
Egenvärden: $\lambda_1 = 8.4861, \lambda_2 = 26.5139$. $\sqrt{\lambda_1} = 2.9131, \sqrt{\lambda_2} = 5.1492$,
and $\|A\|_2 = \max(\sigma_1, \sigma_2) = (2.9131, 5.1492) = 5.1492$.

Matrisnormer

Example

Beräkna $||A||_\infty, ||A||_1, \|A\|_2$ för A

$$A = \begin{bmatrix} 7 & -10 & 0 \\ -10 & 5 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

Svar:

$$A^T = \begin{bmatrix} 7 & -10 & 0 \\ -10 & 5 & 1 \\ 0 & 1 & 3 \end{bmatrix}.$$

Sedan

$$A^T A = \begin{bmatrix} 149 & -120 & -10 \\ -120 & 126 & 8 \\ -10 & 8 & 10 \end{bmatrix}$$

Matrisnormer

Example

$$A = \begin{bmatrix} 7 & -10 & 0 \\ -10 & 5 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

and $\lambda(A^TA) = (9.2592, 17.0342, 258.7066)$, $\sqrt{\lambda(A^TA)} = (3.0429, 4.1273, 16.0844)$,och
$\|A\|_2 = \max(3.0429, 4.1273, 16.0844) = 16.0844.$
$||A||_1 = \max(|7|+|-10|+0, |-10|+5+1, 0+1+3) = \max(17, 16, 4) = 17$
(ettnormen, största kolonnsumman),
$||A||_\infty = 17$ (maxnormen, största radsumman).

Störningsteori för $Ax = b$

Vi använder normer för att studera konditionstalet för problemet $Ax = b$. Vi vill veta vad som händer med x när A och/eller b ändras. Vi kommer endast att ändra b.

Sats

Låt A vara ickesingulär och $Ax = b \neq 0$. Om $Ay = b + f$ så gäller

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

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 $Ax = b$, varför $\|x\| \leq \|A^{-1}\|\|b\|$ eller $1/||x|| \leq ||A||/||b||$. □

Man kan visa liknande satser för fallen när A eller A *och* b störs.

Konditionstal för $Ax = b$

Antag att $||\cdot||$ är en operatornorm. Då gäller:

- $\kappa(A) \geq 1$, 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 *nästan* är singulär. Om $\kappa(A)$ är stort så finns en matris E med liten norm, så att $A + E$ blir singulär. Vi säger att A "ligger nära" mängden av singulära matriser. Om däremot $\kappa(A) \approx 1$, måste $||E||$ vara stor för att $A + E$ ska bli singulär.
Man kan visa att de E som gör $A + E$ singulär och har minst norm uppfyller $||E|| = ||A||/\kappa(A)$.

Konditionstal för $Ax = b$

Determinanten för A är *inte* ett bra mått på att A nästan är singulär vilket följande exempel illustrerar

Exempel

$$\det(\alpha I) = \alpha^n \text{ medan } \kappa(\alpha I) = 1.$$

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

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

Konditionstal för $Ax = b$

Exempel (Hur väl stämmer satsen?)

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

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

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

dvs. likhet i gränsen. Tar vi istället

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

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

Tolkning av satsen

Antag att elementen i x respektive y är ungefär lika stora. Då gäller $x \approx x_k e$ och $y \approx y_k e$ där $e = (1, 1, ..., 1)$. Vi får

$$\frac{||x-y||}{||x||} \approx \frac{||(x_k - y_k)e||}{||x_k e||} = \frac{|x_k - y_k|}{|x_k|}$$

dvs. normen uppskattar det elementvisa felet. För detta specialfall gäller

$$\frac{|x_k - y_k|}{|x_k|} \lessapprox \kappa(A)\frac{||f||}{||b||}$$

Ovanstående säger att det relativa felet i varje komponent begränsas av det relativa felet i indata multiplicerat med konditionstalet för A.

Om t.ex. $||f||/||b|| = 0.5 \cdot 10^{-k}$ (k decimaler) och $\kappa(A) \approx 10^p$ så

$$\frac{|x_k - y_k|}{|x_k|} \lessapprox 10^p \cdot 0.5 \cdot 10^{-k} = 0.5 \cdot 10^{p-k}$$

Som tumregel får vi således följande:

Tolkning av satsen

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 $||\cdot||_\infty$. Om $p - k = -3$ så

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

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.

Tolkning av satsen

Hur stora fel har vi i indata? Låt oss studera två fall.
Exakt indata: Vi får eventuellt avrundningsfel när $a_{j,k}$ och b_k lagras i minnet. Relativa felet (per komponent) är cirka ϵ_{mach}. Vi får även avrundningsfel när vi löser $Ax = b$.
Vi kan troligen tillåta ganska stora $\kappa(A)$, men det beror på hur många siffror vi behöver. Om vi har stora krav, eller för väldigt stort $\kappa(A)$ kan vi minska ϵ_{mach} genom att använda t.ex. Maple eller Mathematica. Att räkna med många fler siffror går dock mycket långsammare (mjukvara, inte hårdvara).
Indata med osäkerhet (mätdata): Ger normalt större begränsningar på hur stort $\kappa(A)$ vi kan tillåta, eftersom vi oftast inte mäter väldigt noga.
Att räkna med mindre ϵ_{mach} ger oftast inte en mer exakt lösning, ty om $\kappa(A)$ är måttligt stort så dominerar mätfelen över avrundningsfelen.

Uppskattning av $\kappa(A)$

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 det linjära ekvationssystem (samt andra listigheter). Även LAPACK kan ge en sådan uppskattning när man löser $Ax = b$. Uppskattningen kostar nästan ingenting, eftersom man uppnyttjar den LU-faktorisering som redan beräknats.

Tolkning av residualen

Vad säger residualen $r = b - A\hat{x}$? ($\hat{x}$ är den beräknade lösningen).

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

Kan visa $(A + E)\hat{x} = b,\ ||E||_2 \leq ||r||_2/||\hat{x}||_2$. Bevis:
$A\hat{x} + E\hat{x} = b, E\hat{x} = b - A\hat{x} = r \rightarrow\ ||E||_2 \leq ||r||_2/||\hat{x}||_2$.
En liten residual betyder att vi har löst nästan rätt problem. I framåtriktningen kommer $\kappa(A)$ in:

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

Vi får $||x - \hat{x}|| \leq ||A^{-1}||\ ||r||$. Vidare om $b \neq 0$ får vi

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

Kombinerar vi dessa olikheter får vi

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

Felet i lösningen kan vara godtyckligt stort även om $||r||$ är liten.

En mer generell störningssats

Sats

Lemma
Låt $Ax = b \neq 0$ och $(A + \delta A)\hat{x} = b + \delta b$, $\hat{x} = x + \delta x$, då

$$\frac{||\hat{x} - x||}{||\hat{x}||} \leq k(A) \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\hat{x}\|} \right).$$

En mer generell störningssats

Bevis:

$$\begin{aligned} (1) \quad & Ax = b, \\ (2) \quad & (A + \delta A)\hat{x} = b + \delta b. \end{aligned}$$

(2)-(1) och använd $\hat{x} = x + \delta x$:

$$\begin{aligned} Ax + A\delta x + \delta A x + \delta A \delta x - Ax &= \delta b, \\ A\delta x + \delta A(\delta x + x) &= \delta b, \\ A\delta x &= \delta b - \delta A \hat{x}, \\ \delta x &= A^{-1}(\delta b - \delta A \hat{x}). \end{aligned}$$

En mer generell störningssats

- Ta normer och använd triangelolikheten:

$$\|\delta x\| \leq \|A^{-1}\|(\|\delta A\| \cdot \|\hat{x}\| + \|\delta b\|)$$

- Rearranging:

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \underbrace{\|A^{-1}\| \cdot \|A\|}_{k(A)} \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\hat{x}\|} \right)$$

var $k(A) = \|A^{-1}\| \cdot \|A\|$ är konditionstalet för A.
□

En till störningssats: använd residualen

Lemma

Låt $Ax = b \neq 0$. Låt den beräknade lösningen $\hat{x} = \delta x + x$. Då

$$||\delta x|| \leq ||A^{-1}|| \cdot ||r||.$$

Bevis

Låt oss definiera residualen som $r = A\hat{x} - b$. Då $\hat{x} = A^{-1}(r + b)$ och $\delta x = \hat{x} - x = \hat{x} - A^{-1}b = A^{-1}(r + b) - A^{-1}b = A^{-1}r$. Ta normer:

$$||\delta x|| \leq ||A^{-1}r|| \leq ||A^{-1}|| \cdot ||r||.$$

Detta är det enklaste sättet att uppskatta δx.

En mer generell störningssats

Lemma

Låt $Ax = b \neq 0$ och $(A + \delta A)\hat{x} = b + \delta b$, $\hat{x} = x + \delta x$, då

$$\frac{||\hat{x} - x||}{||x||} \leq \frac{k(A)}{1 - k(A)\frac{||\delta A||}{||A||}} \left(\frac{||\delta A||}{||A||} + \frac{||\delta b||}{||b||} \right).$$

En mer generell störningssats

Bevis

$$\begin{aligned} (1) \quad & Ax = b, \\ (2) \quad & (A + \delta A)\hat{x} = b + \delta b. \end{aligned}$$

(2)-(1) och använd $\hat{x} = x + \delta x$:

$$\begin{aligned} Ax + A\delta x + \delta A x + \delta A \delta x - Ax &= \delta b, \\ A\delta x + \delta A(\delta x + x) &= \delta b, \\ A\delta x &= \delta b - \delta A \hat{x}, \\ \delta x &= A^{-1}(\delta b - \delta A \hat{x}). \end{aligned}$$

En mer generell störningssats

- Vi har:

$$\delta x = A^{-1}(-\delta A\hat{x} + \delta b)$$

Skriver om:

$$\begin{aligned} A\delta x &= -\delta A\hat{x} + \delta b, \\ A\delta x + \delta A\hat{x} &= \delta b, \\ A\delta x + \delta A(x + \delta x) &= \delta b, \\ \delta Ax + (A + \delta A)\delta x &= \delta b. \end{aligned}$$

Löser $\delta Ax + (A + \delta A)\delta x = \delta b$ för δx och får:

-

$$\begin{aligned} \delta x &= \big((A + \delta A)^{-1}(-\delta Ax + \delta b) \\ &= [A(I + A^{-1}\delta A)]^{-1}(-\delta Ax + \delta b) \\ &= (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta Ax + \delta b)\big) \end{aligned}$$

En mer generell störningssats

Lemma

Let $\|\cdot\|$ satisfy $\|AB\| \leq \|A\| \cdot \|B\|$. Then $\|X\| < 1$ implies that $I - X$ is invertible. $(I - X)^{-1} = \sum_{i=0}^{\infty} X^i$, and

$$\|(I - X)^{-1}\| \leq \frac{1}{1 - \|X\|}. \tag{15}$$

En mer generell störningssats

- Ta normer och dividera med $\|x\|$:

$$\begin{aligned}\frac{\|\delta x\|}{\|x\|} &\leq \|(I + A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \left(\|\delta A\| + \frac{\|\delta b\|}{\|x\|}\right) \\ &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|} \left(\|\delta A\| + \frac{\|\delta b\|}{\|x\|}\right) \ (\textit{Lemma}\ ) \\ &= \frac{\|A^{-1}\| \cdot \|A\|}{1 - \|A^{-1}\| \cdot \|A\| \frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|x\|}\right) \\ &\leq \frac{k(A)}{1 - k(A)\frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right)\end{aligned} \tag{16}$$

- relativa felet $\frac{\|\delta x\|}{\|x\|}$ är berående på relativa felet $\frac{\|\delta A\|}{\|A\|}$ och $\frac{\|\delta b\|}{\|b\|}$ i input data.

Exempel: varför behövs pivoting

Vi vill beräkna L, U i LU-faktorisering för

$$A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix}.$$

Först, A är välkonditionerat eftersom
$k(A) = ||A||_\infty \cdot ||A^{-1}||_\infty \approx 4$. Då kan vi förvänta oss att lösa $Ax = b$ exakt. Vi skriver LU-faktorisering utan pivoting:

$$L = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix}, U = \begin{bmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{bmatrix}$$

Vi observerar att LU är inte samma, som själva A:

$$LU = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 0 \end{bmatrix}$$

Exempel: varför behövs pivoting

Jämför konditionstalet för A med konditionstal för L och U:

- $k(A) = ||A||_\infty \cdot ||A^{-1}||_\infty \approx 4$
- $k(L) = ||L||_\infty \cdot ||L^{-1}||_\infty \approx 10^8$
- $k(U) = ||U||_\infty \cdot ||U^{-1}||_\infty \approx 10^8$

Vi ser att $k(A) << k(L) \cdot k(U)$ - varning: vi ska ha förlust av noggrannhet i LU. Det betyder, att vi måste använda pivoting.

Exempel: varför behövs pivoting

Nu ska vi göra LU factorisering med pivoting för A med omvänt ordning av ekvationer:

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-4} & 1 \end{bmatrix}$$

LU factorisering är nu:

$$L = \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Med sådana matriser L och U matrix A är korrekt:

$$A \approx LU = \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Både matriser L och U är välkonditionerade här och den beräknade lösningen är också ganska exakt:

- $k(A) = ||A||_\infty \cdot ||A^{-1}||_\infty \approx 4$
- $k(L) = ||L||_\infty \cdot ||L^{-1}||_\infty \approx 1$
- $k(U) = ||U||_\infty \cdot ||U^{-1}||_\infty = 4$

Exempel: varför behövs pivoting

$$L = \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

och

$$L^{-1} = \begin{bmatrix} 1 & 0 \\ -10^{-4} & 1 \end{bmatrix}, U^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

$$k(L) = ||L||_\infty \cdot ||L^{-1}||_\infty = (|1| + |10^{-4}|) \cdot (|1| + |-10^{-4}|) \approx 1,$$
$$k(U) = ||U||_\infty \cdot ||U^{-1}||_\infty = (|1| + |1|)(|1| + |-1|) = 2 \cdot 2 = 4.$$

- $k(A) = ||A||_\infty \cdot ||A^{-1}||_\infty \approx 4$
- $k(L) = ||L||_\infty \cdot ||L^{-1}||_\infty \approx 1$
- $k(U) = ||U||_\infty \cdot ||U^{-1}||_\infty = 4$

Vi ser att $k(A) \approx k(L) \cdot k(U)$ - korrekt LU-faktorisering.

Några inledande exempel

Ett vanligt problem är att vi har en matematisk modell och uppmätta värden, och vill bestämma värden på parametrar i modellen. En vanlig modell är $b = ce^{\lambda t}$ (halveringstid, befolkningstillväxt, etc.) b skulle kunna vara befolkningen vid tiden t och c är befolkningsmängden vid tiden $t = 0$.
Vi vill bestämma parametern λ genom att utföra m mätningar av b vid olika tidpunkter t_k. Vi får således m stycken par (t_k, b_k), $k = 1, 2, ..., m$. Vi antar att c är känd. Hur ska vi beräkna λ? Vi har m *olika* ekvationer

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

Det är inte sannolikt att samma λ satisfierar alla ekvationerna. Vi är heller inte intresserade av att få m olika värden på λ.

Några inledande exempel

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

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

Detta kan formuleras på följande sätt:
Försök att göra residualerna

$$r_1 = b_1 - ce^{\lambda t_1}, r_2 = b_2 - ce^{\lambda t_2}, ..., r_m = b_m - ce^{\lambda t_m}.$$

så små som möjligt. Vi kan definiera "små" på många olika sätt (normer), t.ex.

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

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

Några inledande exempel

Dessa förslag är inte slumpvis utvalda. Vi inför residualvektorn $r = [r_1, ..., r_m]^T$. Då kan de tre måtten på föregående sida skrivas som

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

Olika normer kommer i regel att ge olika värden på λ. Varje λ är dock det bästa valet för den givna normen.
Det finns oändligt många frågor, med olika svar. Varje svar är dock korrekt svar på den givna frågan. Det finns normalt inte ett bästa värde på λ.

Några inledande exempel

Man kan ha modeller med fler än en parameter. Ett vanligt exempel är att man vill anpassa mätpunkter till en rät linje. Modellen kan skrivas $b = x_1 + x_2 t$, där x_1 och x_2 är parametrar och (t_k, b_k) uppmätta värden. Residualvektorn blir

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

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

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

i någon lämplig norm.

Linjära problem

Observera att detta normalt inte är ett linjärt ekvationssystem. Det går normalt inte att i lösa $Ax = b$ pga. avvikelser i ekvationerna. Slarvigt kan vi skriva $Ax \approx b$.
Om vi hade kunnat lösa $Ax = b$ så hade residualvektorn varit $r = Ax - b = 0$ och mätpunkterna hade följt modellen exakt.
Notera även att matrisen A har fler rader än kolonner.
När residualvektorn kan skrivas $r = Ax - b$ säger vi att problemet är linjärt. Modellen har då utseendet:

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

där uttryck_k beror av mätvärdena och inte på 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.

Linjära problem

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. Vi får

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

λ ingår nu linjärt i modellen.
Om vi antar att c inte är känd (vi mätte aldrig b för $t = 0$) så är c en parameter som ingår ickelinjärt i modellen. Sätt $x_1 = \log c$ och vi får en linjär modell som är identisk med modellen för vår räta linje:
$\log b = x_1 + \lambda t$. För att göra analogin tydligare sätter vi också $x_2 = \lambda$.
Då får vi

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

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

Linjära problem

När vi gör transformationer på detta sätt ändrar vi (ibland) på normen. Logaritmering, t.ex. har en utjämnande effekt och minskar de stora residualernas inflytande.

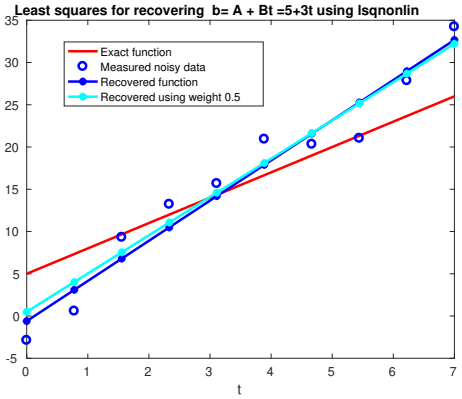
Detta kan jämföras med att minimera i en annan norm. Vi ställer en annan fråga, men den kan vara lika relevant.

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.

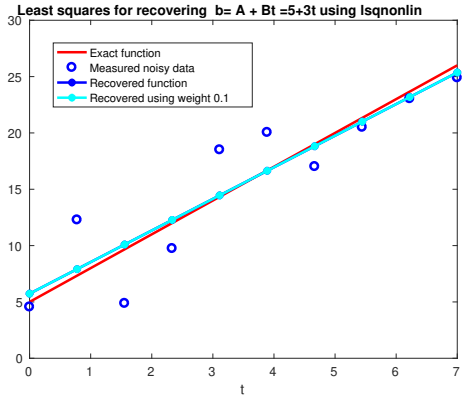
$$\min_x \|V(Ax - b)\|, \;\; V = \text{diag}(v_1, v_2, ..., v_m)$$

Residual r_k multipliceras alltså med vikten v_k.

Lösning av normalekvationerna med lsqnonlin



a) vikt=0.5



b) vikt=0.1

Anpassning till räta linjen $b = 5 + 3x$.
m=antalet datapunkter: vi beräknade med $m = 10$:
x = linspace(0,7.0,m);
brus = brus*60.0*randn(size(x)); b = b + brus; brus=0.5 eller brus=0.1 i exemplet
Linjära minstakvadratproblemet (i vårt fall anpassning till rätt linje $c_1 + c_2 \cdot x = b$) är :

$$\min_c \|(Ac - b)\|_2^2$$

och med vikt $V = diag(v_1, \ldots, v_m)$:

$$\min_c \|V(Ac - b)\|_2^2$$

Optimalitet

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 och a_2; kan enkelt generaliseras till ett godtyckligt fall. För en godtycklig $x \in \mathbb{R}^2$ 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, $\mathcal{R}(A)$.

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

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \; b = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Optimalitet

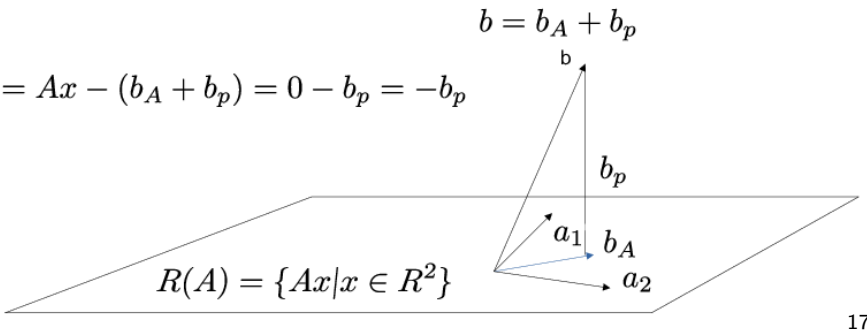
Normalt bildar dock b en vinkel mot planet, tag t.ex. $b = [2, 1, 2]^T$. Vektorn b kan då inte skrivas som en linjärkombination av As kolonner, men vi vill minimera avvikelsen, längden av residualvektorn $Ax - b$.

Deta upp b i två komponenter, b_A som ligger i planet, och $b_\perp$ som är ortogonalt mot planet. Oavsätt 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 därmed är en linjärkombination av As kolonner, dvs. det finns (minst) ett x så att $b_A = Ax$. Det är detta x vi söker.
Residualvektorn blir $r = Ax - b = Ax - b_A - b_\perp = -b_\perp$.

Optimalitet

Igen: dela upp b i två komponenter, b_A som ligger i planet, och $b_p = b_\perp$ som är ortogonalt mot planet. Oavsätt hur vi väljer x så kan vi inte nollställa någon del av b_p, eftersom b_p är ortogonal mot alla linjärkombinationer, Ax. Däremot kan vi nollställa b_A, eftersom b_A ligger i planet och därmed är en linjärkombination av As kolonner, dvs. det finns (minst) ett x så att $b_A = Ax$. Det är detta x vi söker.
Residualvektorn blir $r = Ax - b = \underbrace{Ax - b_A}_{=0} - b_p = -b_p = -b_\perp$.

$$r = Ax - b = Ax - (b_A + b_p) = 0 - b_p = -b_p$$



Optimalitet

Här följer samma resonemang med normer:

Sats (Pythagoras)

Om y och z är ortogonala vektorer gäller:

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

ty

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

Optimalitet

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. Testa:

$$\|A(x+z) - b\|_2^2 = \|A(x+z) - b_A - b_\perp\|_2^2 =$$
$$\|\underbrace{Ax - b_A}_{=0} + Az - b_\perp\|_2^2 = \|Az\|_2^2 + \|b_\perp\|_2^2 \geq \|b_\perp\|_2^2$$

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 $a_1^T r = a_2^T r = 0$. Vi kan skriva dessa likheter på följande form:

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

vilket ger oss normalekvationerna:

Normal Equations

Our goal is to minimize the residual $\|r(x)\|_2^2 = ||Ax - b||_2^2$. To find minimum of this functional and derive the *normal equations*, we look for the x where the gradient of $||Ax - b||_2^2 = (Ax - b)^T(Ax - b)$ vanishes, or where $(r^T(x)r(x))' = 0$. So we want

$$\begin{aligned} 0 &= \lim_{e\to 0} \frac{r^T(x+e)r(x+e) - r^T(x)r(x)}{||e||_2} \\ &= \lim_{e\to 0} \frac{(A(x+e)-b)^T(A(x+e)-b) - (Ax-b)^T(Ax-b)}{||e||_2} \\ &= \lim_{e\to 0} \frac{2e^T(A^TAx - A^Tb) + e^TA^TAe}{||e||_2} \end{aligned}$$

The second term $\frac{|e^TA^TAe|}{||e||_2} \leq \frac{||A||_2^2||e||_2^2}{||e||_2} = ||A||_2^2||e||_2$ approaches 0 as e goes to 0, so the factor $A^TAx - A^Tb$ in the first term must also be zero, or $A^TAx = A^Tb$. This is a system of n linear equations in n unknowns, the normal equations.

Entydighet

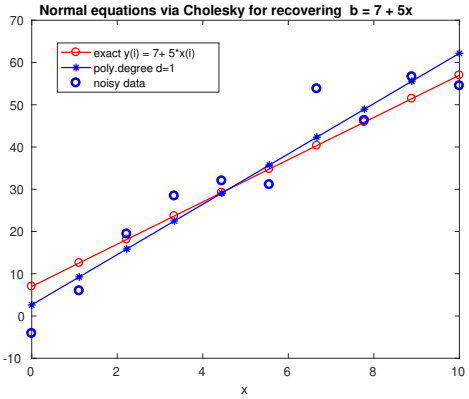
rang$(A) = n \Rightarrow A^TA$ symmetrisk och positivt definit. Kan lösa normalekvationerna med Choleskyfaktorisering.

Entydighet?

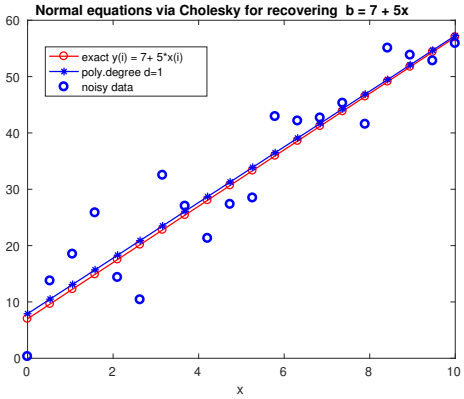
- Om A har linjärt obereoende kolonner så har minstakvadratproblemet en entydlig lösning. Matrisen har full rang.
- Om A har linjärt beroende kolonner (är rangdefekt) så finns det oändligt många lösningar som ger samma residualvektor, ty tag $z \in \mathcal{N}(A)$. Då gäller $A(x+z) = Ax$.

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 $\kappa(A^TA) = \kappa(A)^2$.

Lösning av normalekvationerna med Choleskyfaktorisering



a) $m = 10$



b) $m = 20$

Anpassning till räta linjen $b = 7 + 5x$. Data är genererade med brus 10%:
m=antalet datapunkter: vi beräknade med $m = 10$ och $m = 20$
x = linspace(0,10.0,m);
brus = 0.1; brus = 0.1*60.0*randn(size(x)); b = b + brus;
Linjära minstakvadratproblemet (i vårt fall anpassning till rätt linje $c_1 + c_2 \cdot x = b$) är :

$$\min \|Ac - b\|_2^2$$

Lösning av normalekvationerna med Choleskyfaktorisering

Linjära minstakvadratproblemet (i vårt fall anpassning till rätt linje $c_1 + c_2 \cdot x = b$) är :

$$\min_c \|Ac - b\|_2^2$$

A är konstruerat som Vandermonde matris i problemmet polynomial fitting $f(x, c) = \sum_{i=1}^{d} c_i x^{i-1} = \sum_{i=1}^{d} c_i \phi_i$ till data $(x_i, b_i), i = 1, ..., m$:

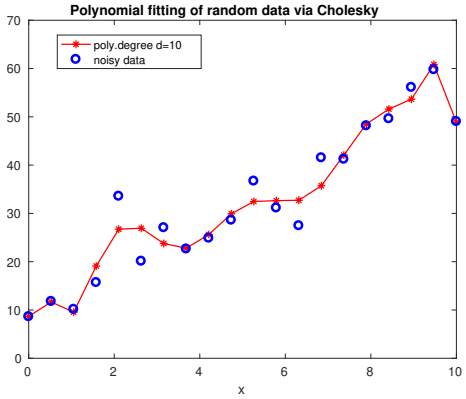
$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^{d-1} \\ 1 & x_2 & x_2^2 & ... & x_2^{d-1} \\ 1 & x_3 & x_3^2 & ... & x_3^{d-1} \\ ... & ... & ... & ... & ... \\ 1 & x_m & x_m^2 & ... & x_m^{d-1} \end{bmatrix}$$

I exemplet anpassning till rätt linje $c_1 + c_2 \cdot x = b$ har vi:

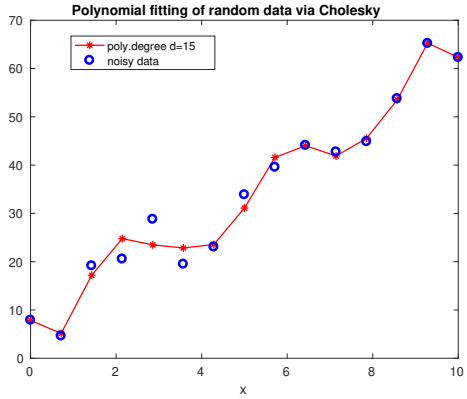
$$A = \begin{bmatrix} 1 & x_1 \\ .. & .. \\ 1 & x_m \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ ... \\ b_m \end{bmatrix} \Rightarrow x = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Exempel på beräknade $\tilde{c}$: $c_1 = 7.9046, c_2 = 4.9372$.
Relative fel: $e = \frac{\|c - \tilde{c}\|_2}{\|c\|_2} = 0.1054$

Polynomial fitting med Choleskyfaktorisering



a) $m = 20, d = 10$



b) $m = 15, d = 15$

Anpassning till polynomial $f(x, c) = \sum_{i=1}^{d} c_i x^{i-1} = \sum_{i=1}^{d} c_i \phi_i$ till data $(x_i, b_i), i = 1, ..., m$.

$$\min_c \|Ac - b\|_2^2$$

Entydighet

Det finns bättre metoder baserade på så kallad QR-faktorisering.
"x = A \ b" i Matlab använder QR-faktorisering.
Observera att operatorn \ är överlagrad. Om A är kvadratisk används LU-faktorisering, annars QR-faktorisering. Matlabkoderna för de olika fallen har ingen gemensam del.

Kort om konditionstal för minstakvadratproblem

Antag att x och y löser problemen

$$\min_x \|Ax - b\|_2^2 \text{ resp. } \min_y \|(A+F)y - (b+f)\|_2^2$$

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 för att $A+F$ ska ha samma rang som A. Härledningen blir då avsevärt enklare, men ändå lite besvärlig. Vi antar därför även att $F = 0$, precis som 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 ekvationssystem $Cx = b$ kan skrivas $x = C^{-1}b$ så det verkar rimligt att betrakta $(A^TA)^{-1}A^T$ som en generaliserad invers.

Kort om konditionstal för minstakvadratproblem

Detta gör man och denna invers kan pseudoinversen, betecknat A^+, och kan beräknas med Matlabkommandot `pinv`.

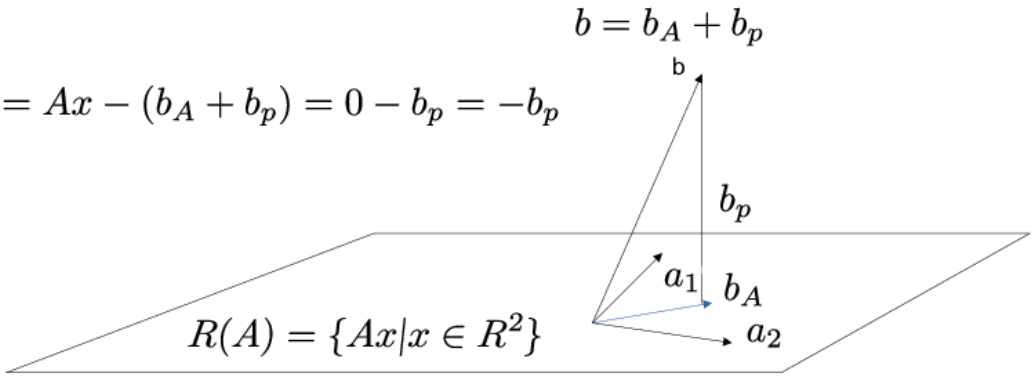
A^+ är ett matematiskt hjälpmedel och den brukar inte användas för att lösa minstakvadratproblem i praktiken. Vi ser att A^+ är en vänsterinvers, $A^+A = (A^TA)^{-1}A^TA = I$. Däremot är inte A^+ en högerinvers, så $AA^+ \neq I$. Man kan definiera A^+ även om A är rangdefekt (men då gäller inte att $A^+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

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

Vi behöver även en undre begränsning av $\|x\|_2$ och använder då 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 behöver även en undre begränsning av $\|x\|_2$ och använder då 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 $x \neq 0$.
Residualvektorn: $r = Ax - b = \underbrace{Ax - b_A}_{=0} - b_p = -b_p = -b_\perp$.

$$r = Ax - b = Ax - (b_A + b_p) = 0 - b_p = -b_p$$



Kort om konditionstal för minstakvadratproblem

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:

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

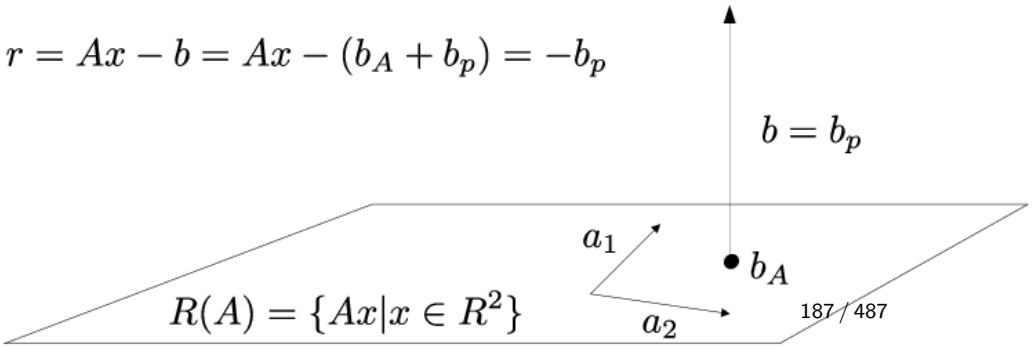
Om modell och mätdata stämmer väl överens så kommer $\|b\|_2/\|b_A\|_2$ att vara nära ett (kvoten är alltid ≥ 1), men om modell och mätdata inte passar ihop så kan kvoten bli stor.

Kort om konditionstal för minstakvadratproblem

Extremfallet är att b är ortogonal mot A:s bildrum i vilket fall $b_A = 0$ och kvoten $\frac{\|b\|_2}{\|b_A\|_2}$ i

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

blir oändlig.



Kort om konditionstal för minstakvadratproblem

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

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

Alternativ till normalekvationerna

Ett exempel som visar nackdelen med normalekvationerna. Låt

$$A = \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix}, \epsilon > 0, A^T A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \epsilon & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1+\epsilon^2 \end{bmatrix}$$

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

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

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

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

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

En komplex matris, Q, sägs vara unitär då $Q^H Q = I$.

Alternativ till normalekvationerna

Så unitär är motsvarigheten till ortogonal för reella matriser. Bevis av $\|Qv\|_2 = \|v\|_2$: Uttnyttja att $\|\cdot\|_2 \geq 0$ och att

$$\|Qv\|_2^2 = (Qv)^T Qv = v^T Q^T Qv = v^T Iv = v^T v = \|v\|_2^2$$

Sats (QR-faktorisering)

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

Lösningen x till minstakvadratproblemet ges då av

$$Rx = Q^T b$$

som är ett triangulärt system. Detta följer av normalekvationerna (som vi bara använder för att visa ovantsående): $A^T Ax = A^T b$ med $A = QR$, där R är ickesingulär ger

$$(QR)^T (QR)x = (QR)^T b \Leftrightarrow R^T Q^T QRx = R^T Q^T b \Leftrightarrow Rx = Q^T b$$

Ett fysikproblem

Svante Arrhenius (1859-1927) är en av grundarna av den fysikaliska kemin. Han undersökte (bland annat) hur hastigheten hos kemiska reaktioner beror av temperaturen. Om t.ex. ämnena α och β reagerar och producerar ämnet γ, så gäller (ofta) att:

$$\frac{d[\gamma]}{dt} = k(T)\,[\alpha]^m\,[\beta]^n$$

där $[\cdot]$ betecknar koncentrationen, t är tiden, och T är absoluta temperaturen (i Kelvin). m och n kallas ordningar (båda kan vara ett t.ex.).
Arrhenius ekvation (1889) är en modell för utseendet på k(T):

$$k(T) = Ae^{-E/RT}$$

A kallas den pre-exponentiella faktorn, E (ofta skriven E_a) är aktiveringsenergin och R är den allmänna gaskonstanten.

Ett fysikproblem

Arrhenius resonerade så här: För att en kemisk reaktion, mellan två molekyler skall inträffa, måste rörelseenergin hos molekylerna uppnå en viss nivå, aktiveringsenergin E.

Enligt Ludwig Boltzmanns (1844-1906) arbeten (statisktisk mekanik och termodynamik) följer att antalet kollisioner med energi $\geq E$ är $e^{E/RT}$ så $k(T)$ bör vara proportionell mot denna faktor. Om temperaturen öker så blir sannolikheten större att molekyler uppnår E varför k(T) ökar.

Arrhenius formel passar till flera andra situationer: frekvensen av syrsors spelande (som funktion av T), myrors krypande, åldrandets hastighet, eldflugors lysande, och hur snabbt man glömmer. Anledningen att Arrhenius formel passar in är att ovanstående processer är kemiska.

Ett fysikproblem

Nu till tillverkningen av glas. Det är intressant att ha en modell för beroendet mellan viskositet (av en glas-smälta) och temperatur. Arrhenius modell stämmer inte så bra. Man noterade att $\log b$ inte var linjär i $1/T$:

$$b = Ae^{-E/RT} \Leftrightarrow \log b = \log A - \frac{E}{R} \cdot \frac{1}{T}$$

Här, $\log = \log_e = \ln$.

Gordon Fulcher (Corning Glass Works, NY) listade, i en artikel från 1925 följande modeller

$$\log b = A - B/T + C/T^2$$
$$\log b = -A + B/T + C/T^2$$
$$\log b = -A + B \log T + C/T^2$$
$$\log b = -A + B/(T - T_0)^2$$
$$\log b = -A + B/(T - 273)^2.33$$
$$\log b = -A + 10^3 \cdot B/(T - T_0)$$

T ges i $^\circ C$ och $\log = \log_{10}$.

Ett fysikproblem

Den sista ekvationen fungerade rätt väl. Vogel (1925) och Tammann (1926) publicerade samma formel, som nu kallas: Vogel-Fulcher-Tammanns modell (VFT), här skriven på en vanlig form:

$$b = Ae^{E/(T-T_0)} \quad \texttt{VFT}$$

$T_0 = 0$ ger Arrhenius modell. Vi har mätt b vid olika temperaturer, T och vill bestämma parametrarna A, E, samt T_0. Vi har tydligen en ickelinjär modell i parametrarna.

Fulcher använde en grafisk teknik. Först bestämde han T_0 från tre mätvärden. Han plottade sedan $\log b$ som funtion av $1/(T - T_0)$ och anpassade en rät linje till mätpunkterna. Låt oss nu attackera problemet med moderna hjälpmedel. Första idén: formulera problemet som ett ickelinjärt minstakvadratproblem (jag har tagit bort kvadratroten):

Ett fysikproblem

$$\min_{A,\,E,\,T_0} \sum_{k=1}^{n} \left[b_k - Ae^{E/(T_k-T_0)} \right]^2$$

De lösare som används är iterativa och kräver en startapproximation och produerar (förhoppningsvis) en serie approximationer som konvergerar mot ett lokalt minimum.

Lösaren stannar när en avbrottskriterium är uppfyllt. Detta kriterium baseras normalt på förändrigen av approximationerna, förändrigen av funktionen som skall minimeras (objektfunktion, målfunktion) och på normen av gradienten.

Det är viktigt med bra startapproximationer. En dålig approximation kan ge divergens eller konvergens mot ett lokalt minimum med större minimivärde.

Ett fysikproblem

I Matlab kan man använda kommandot `lsqnonlin` för att lösa det ickelinjära minstakvadratproblemet. Använder vi en slumpvektor som startgissning kan man få dåliga anpassningar som inte beror på toleranser i avbrottskriteriet (dessa kan skärpas).
Residualvektorn kan få element av samma storleksordning, men där de relativa avvikelserna blir enormt stora få de små mätvärdena.

Om vi tror (vet) att alla b-värden är givna med samma relativa fel kan man använda vikter, så att alla mätvärden får samma inflytande. Om vi viktar med $1/b_k$ får vi problemet

$$\min_{A,\,E,\,T_0} \sum_{k=1}^{n} \left[\frac{b_k - Ae^{E/(T_k - T_0)}}{b_k} \right]^2$$

Detta visade sig inte fungera så bra, men de små värden kom i alla fall med. Felet var startgissningen. Vi behöver bättre värden.

Ett fysikproblem

För att bestämma startapproximationer på parametrarna skriver vi om det ickelinjära problemet som ett linjärt problem. Detta går givetvis inte alltid. Logaritmera VFT:

$$\log b = \frac{E}{T - T_0} + \log A$$

Multiplicera upp $T - T_0$ och samla ihop termerna:

$$T \log b = \underbrace{T_0}_{x_1} \log b + T \underbrace{\log A}_{x_2} + \underbrace{E - T_0 \log A}_{x_3}$$

Låt $x_1 = T_0, x_2 = \log A$ och $x_3 = E - T_0 \log A$. Det linjära problemet kan då skrivas för $x = [x_1, x_2, x_3]^T$:

$$\min_x \left\| \begin{bmatrix} \log b_1 & T_1 & 1 \\ \log b_2 & T_2 & 1 \\ \vdots & \vdots & \vdots \\ \log b_n & T_n & 1 \end{bmatrix} x - \begin{bmatrix} T_1 \log b_1 \\ T_2 \log b_2 \\ \vdots \\ T_n \log b_n \end{bmatrix} \right\|_2^2$$

Ett fysikproblem

Här är Matlabkoden:

```
n = length(T);
x = [log(b), T, ones(n, 1)] \ (T .* log(b));

T0 = x(1);
A= exp(x(2));
E = x(3) + T0 * x(2);
```

Använder vi dessa startapproximationer till `lsqnonlin` blir plotten av anpassningen nästan perfekt.

Ett fysikproblem

$$\min_{A,\,E,\,T_0} \sum_{k=1}^{n} \left[\log b_k - \log A - \frac{E}{T_k - T_0} \right]^2$$

Värden skiljer sig dock inte ifrån vad det viktade problemet ger. Detta kan bero på att parametrarna är dåligt bestämda av målfunktionen. Två enkla exempel:

Exempel

Minimera $f(x) = x^2$ och $g(y) = y^4$. Antag att vi accepterar x och y som minimum om $f(x) \leq 10^{-8}, g(y) \leq 10^{-8}$. Vi får intervallen $-10^{-4} \leq x \leq 10^{-4}$ respektive $-10^{-2} \leq y \leq 10^{-2}$.

Ett fysikproblem

Exempel

$$\min_{x_1,\, x_2} (x_1 + x_2)^2$$

Minimivärden är noll, som antas för alla x_1 och x_2 där $x_1 + x_2 = 0$. Vi har inte ett entydigt minimum. Måfunktionen ser ut som ett dike (ränna). Om vi rör oss utmed dikets botten ändras inte funktionens värde. Hessianen, H, är en 2×2-matris av tvåor, så H är positivt semidefinit (singulär).

En annan orsak kan vara att vi har få mätpunkter, nio, i förhållande till antalet, tre, parametrar. Det hade varit trevligt med, säg 30, mätpunkter. Tyvärr tar redan nio mätpunkter ett dygn att producera. Mätfel påverkar också resultatet.

Ett fysikproblem

Betrakta följande funktion för fixt T_0 (det optimala värdet), som funktion av A och E

$$f(A, E) = \left\{ \sum_{k=1}^{n} \left[\log b_k - \log A - \frac{E}{T_k - T_0} \right]^2 \right\}^{1/2}$$

Grafen till denna funktion ser ut som ett dike. Övriga kombinationer, fixt A respektive fixt E ger liknande plottar. Det går även att förstå problemet genom att studera residualfunktionen:

$$f(\log A, E, T_0) = \sum_{k=1}^{n} \left[\log b_k - \log A - \frac{E}{T_k - T_0} \right]^2$$

Man kan visa att residualen inte ändrar sig så mycket utmed ett tredimensionellt dike, dvs. minimum är illa bestämt.

Matlabs funktion lsqnonlin: example 1

Idé: formulera problem som ett ickelinjärt minstakvadratproblem:

$$\min_{A,E,T_0} \|b - A \cdot exp^{E/(t-T_0)}\|_2^2$$

Problemet kan skrivas som

$$\min_{A,E,T_0} \sum_{k=1}^{n} (b_k - A \cdot exp^{E/(t_k-T_0)})^2$$

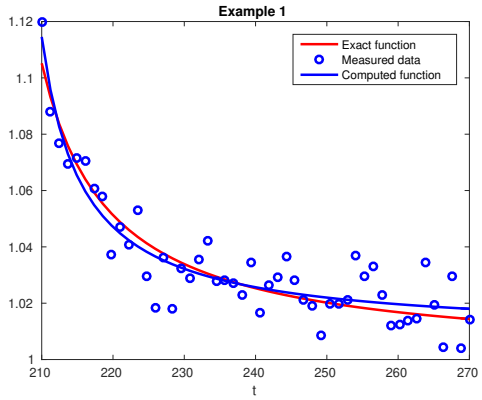
Matlabs funktion lsqnonlin: example 1, version 1

```
% genererar data
t = linspace(210,270,50);
% genererar exakt funktion i data punkter
b = A*exp(E*(1./(t-T0)));
% genererar observationer med random brus
brus = 0.01;
rhs = b + brus*randn(size(t));
%init gissning: fint gissning är: x0 = [A,E,T0];
%init gissning
x0 = [1,1,1];
%definition av funktion som vi vill anpassa till mätningar i
rhs fun = @(x)x(1)*exp(x(2)*(1./(t-x(3)))) - rhs;
x = lsqnonlin(fun,x0)
figure
plot(t,b,'r-', t,rhs, 'b o',t,fun(x)+rhs,'b -','LineWidth',2)

xlabel('t')
legend('Exact function','Measured data','Computed function')
```

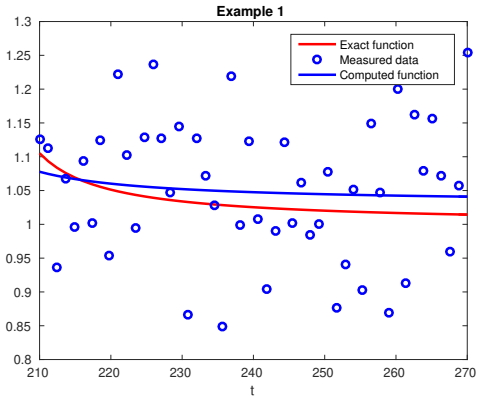
Matlabs funktion lsqnonlin: example 1, version 2

```
% exakta koefficienter:
A=1; E=1; T0 =200;
% genererar data
t = linspace(210,270,50);
% genererar exakt funktion i data punkter
b = A*exp(E*(1./(t-T0)));
% genererar observationer med random brus
brus = 0.01; rhs = b + brus*randn(size(t));
%init gissning
x0 = [1,1,1];
%definition av funktion som vi vill anpassa
fun = @(x)x(1)*exp(x(2)*(1./(t-x(3)))) - rhs;
% lower band (lb) and upper bound (ub) för A,E,T0
lb = [1/2,1/2,0.0]; ub = [2.0,3.0,400.0];
x = lsqnonlin(fun,x0,lb,ub)
figure
plot(t,b,'r-', t,rhs, 'b o',t,fun(x)+rhs,'b -','LineWidth',2)
xlabel('t') legend('Exact function','Measured data','Computed
function') title('Example 1')
```


Example 1



a) brus 1%



b) brus 10 %

Beräknad x med brus=1% (vi har fått i matlab-program
$A = 0.9932, E = 1.4977, T_0 = 196.5996$):
x = 0.9932 1.4977 196.5996
Beräknad x med brus= 10%:
x = 0.9654 2.7695 193.6234

Matlabs funktion lsqnonlin: example 2

Idé: formulera problem som ett linjärt minstakvadratproblem:

$$\min_{A,E,T_0} \left\| log(b) - log(A) - \frac{E}{t - T_0} \right\|_2^2$$

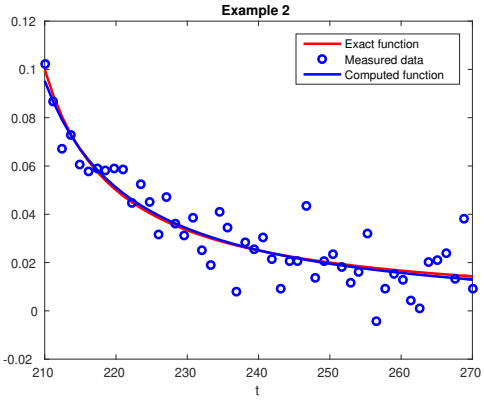
Problemet kan skrivas som

$$\min_{A,E,T_0} \sum_{k=1}^{n} \left(log(b_k) - log(A) - \frac{E}{t_k - T_0} \right)^2$$

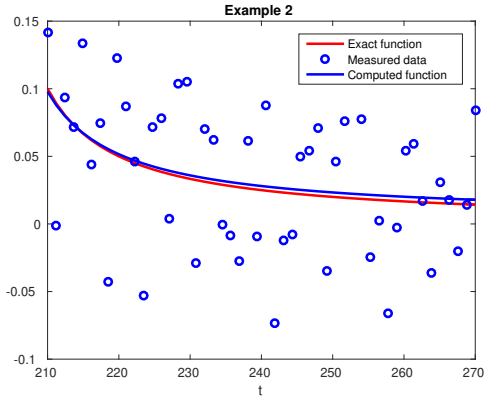
Matlab program: example 2

```
% exakta parametrar A=1; E=1; T0 =200;
% genererar data punkter
t = linspace(210,270,50);
% genererar exakt funktion y=log(b) i data punkter
y = log(A) + E*(1./(t-T0));
% genererar observationer med random brus
brus = 0.01; rhs = y + brus*randn(size(t));
%init gissning
x0 = [1,1,1];
%definition av funktion som vi vill anpassa
fun = @(x)log(x(1)) + x(2)*(1./(t-x(3))) - rhs;
x = lsqnonlin(fun,x0)
figure
plot(t,y,'r-', t,rhs, 'b o',t,fun(x)+rhs,'b -','LineWidth',2)

xlabel('t')
legend('Exact function','Measured data','Computed function')
```

Example 2



a) brus 1%



b) brus 5 %

Beräknad x med brus=1% (vi har fått $A = 0.9958$, $E = 1.2448$, $T_0 = 197.4796$):
x =
0.9958 1.2448 197.4796
Beräknad x med brus= 5%:
x =

1.0042 0.9581 199.7549

Matlabs funktion lsqnonlin: example 3

Idé: formulera problem som ett viktat ickelinjärt minstakvadratproblem:

$$\min_{A,E,T_0} \left\| \frac{b - A \cdot exp^{E/(t-T_0)}}{vikt} \right\|_2^2$$

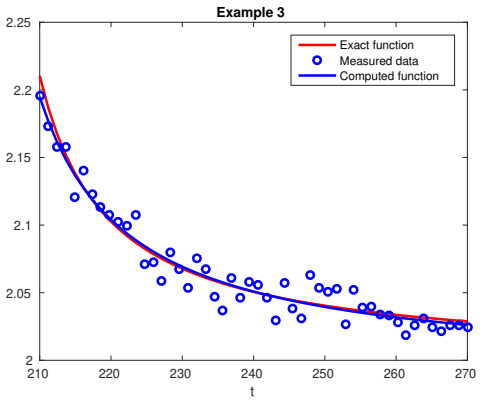
Problemet kan skrivas som

$$\min_{A,E,T_0} \sum_{k=1}^{n} ((b_k - A \cdot exp^{E/(t_k-T_0)})/vikt)^2$$

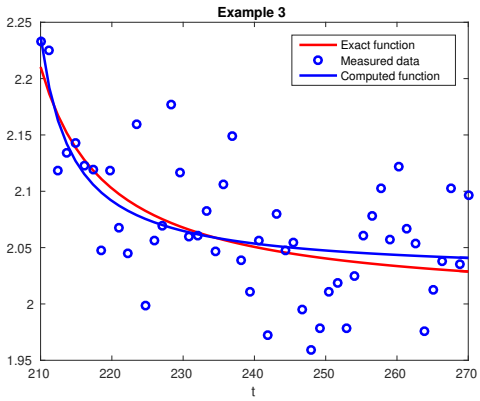
Matlab program: example 3

```
A=1; E=1; T0 =200;
% genererar data punkter t = linspace(210,270,50);
% genererar exakt funktion i data punkter med vikt
vikt = 0.5;
b = (A*exp(E*(1./(t-T0))));
% genererar observationer med random brus
brus = 0.01;
rhs = b + brus*randn(size(t));
% init gissning
x0 = [1,1,1];
%definition av funktion som vi vill anpassa
fun = @(x)(x(1)*exp(x(2)*(1./(t-x(3)))) - rhs)*vikt;
x = lsqnonlin(fun,x0)
funcomp = x(1)*exp(x(2)*(1./(t-x(3))));
figure
plot(t,b,'r-', t,rhs, 'b o',t,funcomp,'b -','LineWidth',2)
xlabel('t')
legend('Exact function','Measured data','Computed function')
```


Example 3



a) brus 1%



b) brus 5 %

Beräknad x med brus=1% (vi har fått
$A = 0.9957, E = 1.2655, T_0 = 196.9944$):
x = 0.9957 1.2655 196.9944
Beräknad x med brus= 5%:
x = 1.0131 0.4666 205.2873

Övningar

Gordon Fulcher (Corning Glass Works, NY) listade, i en artikel från 1925 följande modeller

1. $\log b = A - B/T + C/T^2$
2. $\log b = -A + B \log T + C/T^2$
3. $\log b = -A + B/(T - T_0)^2$

T ges i $^\circ C$ och $\log = \log_{10}$.
Vi vill bestämma parametrarna $x = (A, B, C)$ eller $x = (A, B, T_0)$ givet mätvärden $(T_1, b_1), ..., (T_m, b_m)$. Gör en lämplig transformation och ställ upp ett minstakvadratproblem i formen $\min_x \|Ax - b\|_2^2$. Matrisen A samt vektorerna b och x skall redovisas.

Ickelinjära ekvationer

Betrakta ekvationen

$$f(x) = 0, \ f : \mathbb{R} \to \mathbb{R}$$

Vi kan också betrakta 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 $0, 1, 2, 3, ..., \infty$ lösningar. Ett linjärt problem $Ax = b$ kan ha $0, 1$ eller ∞ många lösningar.

Ickelinjära ekvationer

Det kan tänkas att f är definierad via en procedur, t.ex.

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

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

Ickelinjära ekvationer (Bisektionsmetoden)

Givet en kontinuerlig funktion f och $p, n \in \mathbb{R}$ med $f(n) < 0$, $f(p) > 0$.

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

Om begynnelseintervallet har längden τ har intervallet längden

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

efter k iterationer.

Ickelinjära ekvationer (Bisektionsmetoden)

Bisektionsmetodens fördelar

- räcker att f är kontinuerlig
- konvergerar alltid för f kontinuerlig
- får ett intervall där roten ligger
- deterministiskt 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.

Ickelinjära ekvationer (Sekantmetoden)

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 den nya approximationen c ges utav

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

Iterera: givet två startvärden x_0, x_1

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

Om f är linjär ger denna algoritm nollstället i ett steg.

Matlabs program (halveringsmetoden och sekantmetoden) för $f(x) = x^3 - 2x - 5 = 0$

```
close all; tolerance = 10e-15;
fun = @(x)x^3 - 3 * x - 5;
% definition av intervalet [n,p]
n= 0; p=3; it=0;
% Halveringsmetoden
while abs(n - p) > tolerance
m = (n + p)/2;
if fun(m)< 0
n = m;
else
p = m;
end
it = it+1;
func_val(it) =fun(p); iteration(it) = it;
end
```

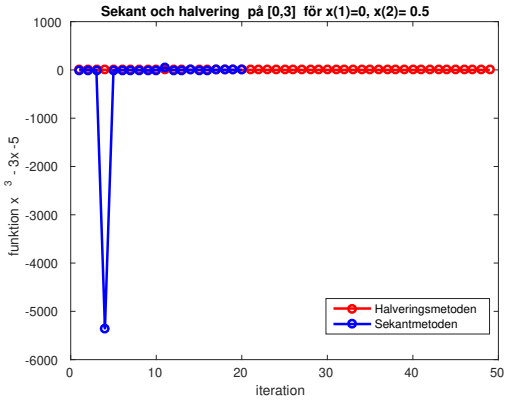

Matlabs program (fortsättningen)

```
%Sekantmetoden
x=0; iteration_sekant = 0;
func_sekant = 0;
k=2;
%Initialiseringen
x(1) = 2.0; x(2) = 1.0;
iteration_sekant(1) = 1; iteration_sekant(2) = 2;
func_sekant(1) =fun(x(1)); func_sekant(2) =fun(x(2));
while abs( x(k) - x(k-1) ) > tolerance
numerator = fun(x(k))*(x(k-1) - x(k));
denominator = fun(x(k-1)) - fun(x(k));
x(k+1) = x(k) - numerator/denominator;
k = k+1
iteration_sekant(k) = k;
func_sekant(k) =fun(x(k));
end
```

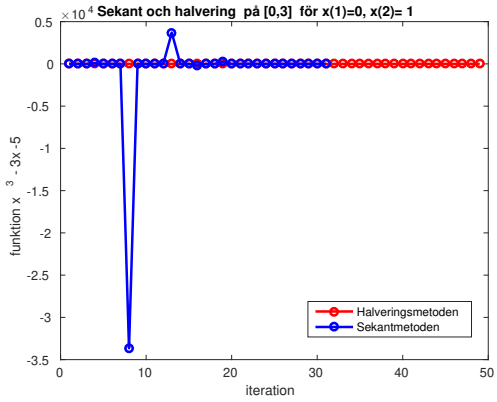
Matlabs program (fortsättningen)

figure
plot(iteration, func_val,'o r-', 'LineWidth',2)
hold on plot(iteration_sekant,func_sekant, 'o b-', 'LineWidth',2)
xlabel('iteration')
ylabel('funktion $x^3 - 3x - 5$')
legend('Halveringsmetoden','Sekantmetoden')
title(['Sekant och halvering på [0,3] för x(1)=',num2str(x(1)),', x(2)= ',num2str(x(2))])

Exempel: halveringsmetoden versus sekantmetoden



a) startvärde $x(1) = 0, x(2) = 0.5$

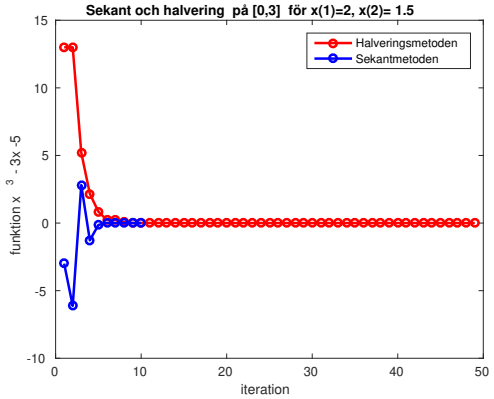


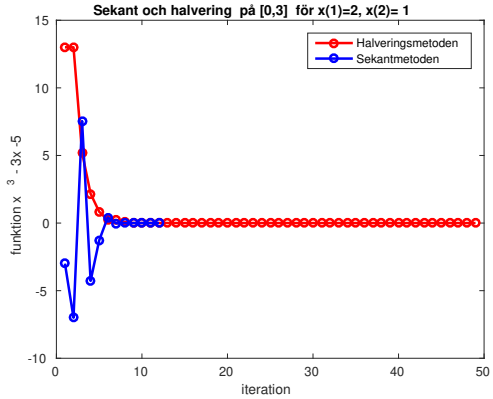
b) startvärde $x(1) = 0, x(2) = 1$

tol = 10e-15; Beräknad x i halveringsmetoden:
x = 2.2790
Beräknad x i sekantmetoden:
x = 2.2790

nr.iterations i halveringsmetoden: a), b) 49, nr.iterations i sekantmetoden: a) 20, b) 31;

Exempel: halveringsmetoden och sekantmetoden

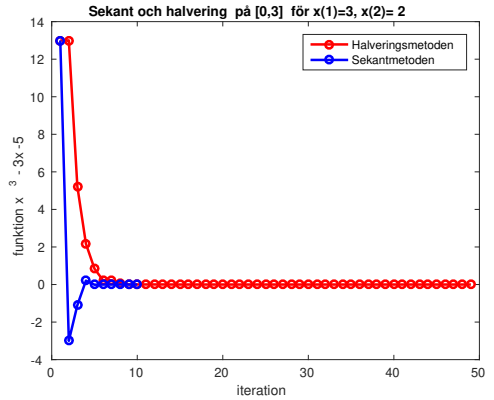


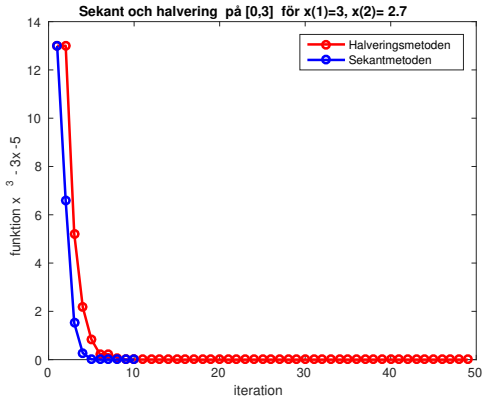


a) startvärde $x(1) = 2, x(2) = 1.5$ b) startvärde $x(1) = 2, x(2) = 1$

tolerance = 10e-15; Beräknad x i halveringsmetoden:
x = 2.2790
Beräknad x i sekantmetoden:
x = 2.2790
nr.iter. i halveringsmet.: a), b) 49, nr.iter. i sekantmet.: a) 10, b) 12;

Exempel: halveringsmetoden och sekantmetoden





a) startvärde $x(1) = 3, x(2) = 2$ b) startvärde $x(1) = 3, x(2) = 2.7$

tolerance = 10e-15; Beräknad x i halveringsmetoden:
x = 2.2790
Beräknad x i sekantmetoden:
x = 2.2790

nr.iterations i halveringsmetoden: a), b) 49, nr.iterations i sekantmetoden: a), b) 10.

Ickelinjära ekvationer (Newtons metod)

Vi vill lösa ekvation $f(x) = 0$.
Taylor's theorem:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(Q)h^2}{2!},$$

för $Q \in [x, x+h]$, eller vi kan skriva Taylor's theorem som:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(Q)(x - x_0)^2}{2!},$$

för $Q \in [x_0, x]$, $h = x - x_0$.

$$0 = f(x) \approx f(x_0) + f'(x_0)(x - x_0),$$

som vi skriver om:

$$f(x_0) + f'(x_0)(x - x_0) \approx 0$$

Vi kan beräkna x från ovanstpående ekvation som:

$$x - x_0 \approx -\frac{f(x_0)}{f'(x_0)}, \; x \approx x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Ickelinjära ekvationer (Newtons metod)

Kan approximera med tangenten istället för sekanten (Newton-Raphson, 1690). Tangenten har ekvationen:

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

När $x = c$ (en rot) är $y = 0$. Alltså

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

Iterera:

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

Kräver endast ett startvärde, men måste ha derivatan. Newtons eget exempel: $x^3 - 2x - 5 = 0$. Iterationen blir

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

Algorithm: Newtons's metod för 1-D ickelinjär ekvation

- x_0 = initial guess;
- for $k = 0, 1, 2, ...$
  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
  end

Matlabs program (halveringsmetoden, sekantmetoden och Newton's metod) för $f(x) = x^3 - 2x - 5 = 0$

```
close all; tolerance = 10e-10;
fun = @(x)x^3 - 3 * x - 5;
% definition av intervalet [n,p]
n= 0; p=3; it=0;
% Halveringsmetoden
while abs(n - p) > tolerance
m = (n + p)/2;
if fun(m)< 0
n = m;
else
p = m;
end
it = it+1;
func_val(it) =fun(p); iteration(it) = it;
end
```

Matlabs program (fortsättningen)

```
%Sekantmetoden
x=0; iteration_sekant = 0;
func_sekant = 0;
k=2;
%Initialiseringen
x(1) = 2.0; x(2) = 1.0;
iteration_sekant(1) = 1; iteration_sekant(2) = 2;
func_sekant(1) =fun(x(1)); func_sekant(2) =fun(x(2));
while abs( x(k) - x(k-1) ) > tolerance
numerator = fun(x(k))*(x(k-1) - x(k));
denominator = fun(x(k-1)) - fun(x(k));
x(k+1) = x(k) - numerator/denominator;
k = k+1
iteration_sekant(k) = k;
func_sekant(k) =fun(x(k));
end
```


Matlabs program (fortsättningen)

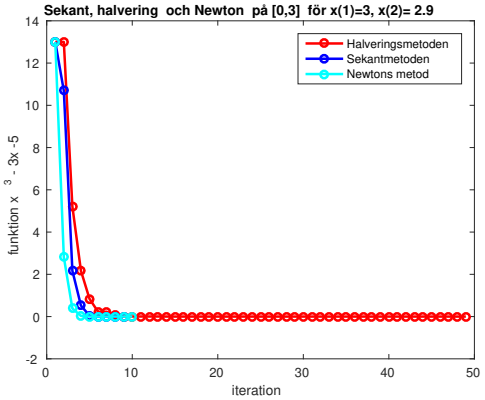
```
figure
plot(iteration, func_val,'o  r-', 'LineWidth',2)
hold on
plot(iteration_sekant,func_sekant, 'o  b-', 'LineWidth',2)
% Newtons metod
y=0; iteration_newton = 0; func_newton = 0;
%Initialiseringen
y(1) = 3.0; k=2;
iteration_newton(1) = 1; func_newton(1) =fun(y(1));
numerator = fun(y(1)); denominator = 3.0*y(1)^2 - 2.0;
y(2) = y(1) - numerator/denominator;
% Main Newton's iterations
while abs( y(k) - y(k-1) )> tolerance
numerator = fun(y(k)); denominator = 3.0*y(k)^2 - 2.0;
y(k+1) = y(k) - numerator/denominator;
iteration_newton(k) = k; func_newton(k) =fun(y(k));
k = k+1;
end
```

Matlabs program (fortsättningen)

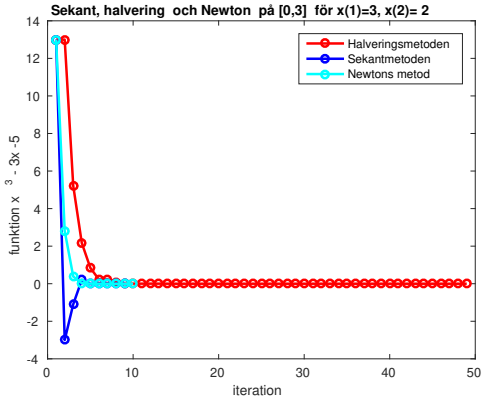
```
plot(iteration_newton,func_newton, 'o  c-','LineWidth',2)
xlabel('iteration')
ylabel('funktion x^3 − 3x − 5')
legend('Halveringsmetoden','Sekantmetoden','Newtons metod')
title(['Sekant, halvering och Newton på [0,3] för x(1)=',num2str(x(1)),',
x(2)= ',num2str(x(2))])
```


Exempel: halveringsmetoden, sekantmetoden och Newton's metod för $f(x) = x^3 - 2x - 5 = 0$

Startvärde i Newton's metod: $x(1) = 3.0$, tolerance = 10e-10.



a) startvärde i sekantmetod: $x(1) = 3.0, x(2) = 2.9$



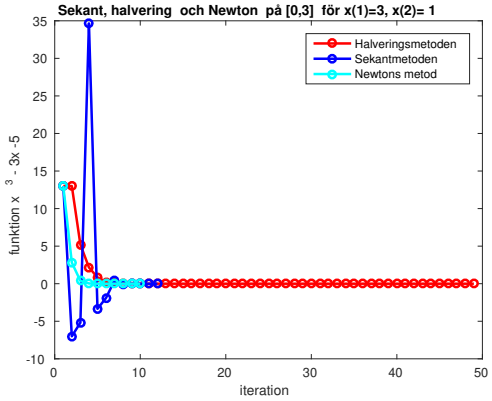
b) startvärde i sekantmetod: $x(1) = 3, x(2) = 2.0$

Beräknad x i halveringsmetoden, sekantmetoden och Newton's metoden:
x = 2.2790

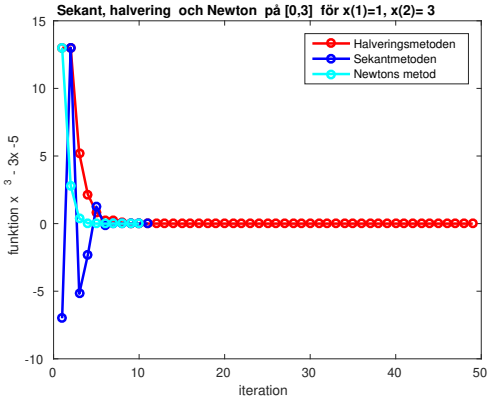
nr.iterations i halveringsmetoden: a), b) 49, nr.iterations i sekantmetoden: a), b) 9; i Newton's metod : a), b) 10.

Exempel: halveringsmetoden, sekantmetoden och Newton's metod för $f(x) = x^3 - 2x - 5 = 0$

Startvärde i Newton's metod: $x(1) = 3.0$, tolerance = 10e-10.



a) startvärde i sekantmetod: $x(1) = 3.0, x(2) = 1.0$



b) startvärde i sekantmetod: $x(1) = 1, x(2) = 3.0$

Beräknad x i halveringsmetoden, sekantmetoden och Newton's metoden:
x = 2.2790

nr.iterations i halveringsmetoden: a), b) 49, nr.iterations i sekantmetoden: a), 12 b) 11; i Newton's metod : a), b) 10.

Ickelinjära ekvationer (Newton för system)

Repetition av Taylors formel.

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

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

Matrisen av partiella derivator, $J(a,b)$, 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 $g(x_j + h, y_j + k) = 0$.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} f(x_j + h, y_j + k) \\ g(x_j + h, y_j + k) \end{bmatrix} \approx \begin{bmatrix} f(x_j, y_j) \\ g(x_j, y_j) \end{bmatrix} + J(x_j, y_j) \begin{bmatrix} h \\ k \end{bmatrix}$$

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

Ickelinjära ekvationer (Newton för system)

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

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:

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

Ickelinjära ekvationer (Newton för system)

Betrakta som exempel ekvationen

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

Vi skriver problemet på normalform (nollor i ena ledet), så att våra funktioner blir

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

Newtons metod blir då

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

Om vi startar i $x_0 = -3$ och $y_0 = 10$ 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
```

Ickelinjära ekvationer (Newton för system)

```
 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   3.760e+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 för $x, y, z, w, \ldots$ utan får använda vektorer, analogt för funktionerna.

Ickelinjära ekvationer (Newton för system)

Exemplet kan skrivas på följande vis. x och y får vara elementen x_1 resp. x_2 i vektorn x.

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

Vår funktion f med två komponenter blir

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

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:

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

Övningar

Sätt upp Newtons metod för följande problem:

1. $x^3 - 2x - 5 = 0$
2. $e^{-x} = x$
3. $x \sin(x) = 1$

Svar

Newtons metod:

1. $x_{k+1} = x_k - \frac{x_k^3 - 2x_k - 5}{3x_k^2 - 2}$
2. $x_{k+1} = x_k - \frac{e^{-x_k} - x_k}{-e^{-x_k} - 1}$
3. $x_{k+1} = x_k - \frac{x_k \sin(x_k) - 1}{x_k \sin(x_k) + x_k \cos(x_k)}$

Ickelinjära ekvationer (Konvergensordning)

Hur skall vi karakterisera de olika konvergenshastigheterna för halvering, sekant och Newton?

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

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

Ickelinjära ekvationer (Konvergensordning)

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

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

  Dvs. $|x_{k+1} - x^*| \approx C^k|x_0 - x^*|$.
- Vad innebär $r = 2$? Om x_0 ligger tillräckligt nära x^* så gäller att:

  $$|x_1 - x^*| \approx C|x_0 - x^*|^2,\ |x_2 - x^*| \approx C|x_1 - x^*|^2 \approx C^3|x_0 - x^*|^4$$

  Dvs. $|x_{k+1} - x^*| \approx C^{2^k-1}|x_0 - x^*|^{2^k}, k = 1, 2, 3, \ldots$.
- Vad innebär $r = p$? Om x_0 ligger tillräckligt nära x^* så gäller att:

  $$|x_1 - x^*| \approx C|x_0 - x^*|^p,\ |x_2 - x^*| \approx C|x_1 - x^*|^p \approx C(C|x_0 - x^*|^p)^p$$

  Dvs. $|x_{k+1} - x^*| \approx C^{p^k-1}|x_0 - x^*|^{p^k}, k = 1, 2, 3, \ldots$.

Ickelinjära ekvationer (Konvergensordning)

Exempel

$|x_{k+1} - x^*| \approx C|x_k - x^*|^r$. Antag att $|x_0 - x^*| = 0.1$ och $C = 0.1$, då för $r = 1$ är $|x_{k+1} - x^*| \approx C \cdot |x_0 - x^*|^k = 0.1^{k+1}$, $k = 0, 1, 2, ...$ och för $r > 1$ är $|x_{k+1} - x^*| \approx C^{r^k-1}|x_0 - x^*|^{r^k} \approx 0.1^{r^k-1} \cdot 0.1^{r^k} = 0.1^{2r^k-1}$, $k = 0, 1, 2, 3, ...$:

	linjar		kvadratisk	
k	r = 1	q	r = 2	q
0	1e-1		1e-1	
1	1e-2	1	1e-3	2
2	1e-3	1	1e-7	4
3	1e-4	1	1e-15	8
4	1e-5	1	1e-31	16

Ickelinjära ekvationer (Konvergensordning)

Här, q är konvergenshastighet och den är beräknat som

$$q = \log 10(r_k/r_{k+1}), \;\; k = 0, 1, 2, 3, 4$$

Vi observerar att:

- För linjär konvergens när $r = 1$ konvergenshastigheten är $(q_k)^1 = 1, k = 0, 1, 2, 3, 4$
- För kvadratisk konvergens när $r = 2$ konvergenshastigheten är $(q_k)^2, k = 0, 1, 2, 3, 4$

Ickelinjära ekvationer (Konvergensordning)

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 kvadratisk konvergent (om enkelrot)

Ickelinjära ekvationer (Konvergensordning)

Exempel: lös med Newtons metod och halveringsmetoden

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

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

$$x_{k+1} = x_k - \frac{x_k^{10} - a}{10x_k^9} = \frac{9}{10}x_k + \frac{a}{10x_k^9} \approx \frac{9}{10}x_k$$

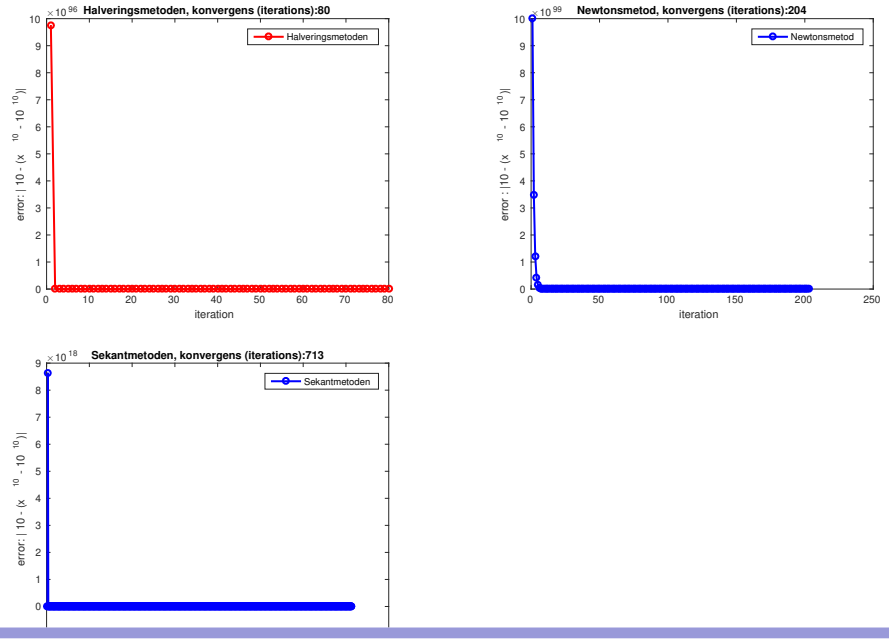
för stora x_k, så att

$$|x_{k+1} - x^*| \approx |x_{k+1}| \approx |\frac{9}{10}x_k| \approx \frac{9}{10}|x_k - x^*|$$

dvs. vi får linjär konvergens med $C = \frac{9}{10}$.

Example: halveringsmetoden, sekantmetoden och Newton's metod för $f(x) = x^{10} - 10^{10} = 0$

Startvärde i Newton's metod: $x(1) = 10^{10}$, tolerance = 10e-15. För halveringsmetoden: $[n, p] = [0, 10^{10}]$. Startvärde i Sekantmetod: $x(1) = 5.0, x(2) = 7$.



Ickelinjära ekvationer (Konvergensordning)

Exempel: lös med Newtons metod och halveringsmetoden

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

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

$$x_{k+1} = x_k - \frac{x_k^{10} - a}{10x_k^9} = \frac{9}{10}x_k + \frac{a}{10x_k^9} \approx \frac{9}{10}x_k$$

för stora x_k, så att

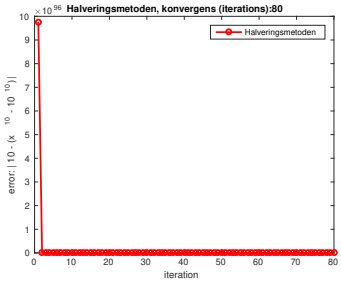
$$|x_{k+1} - x^*| \approx |x_{k+1}| \approx |\frac{9}{10}x_k| \approx \frac{9}{10}|x_k - x^*|$$

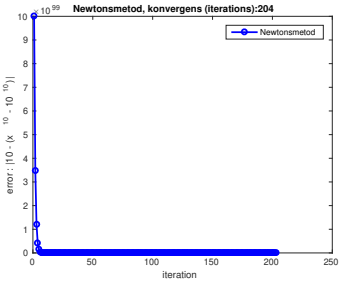
dvs. vi får linjär konvergens med $C = \frac{9}{10}$.

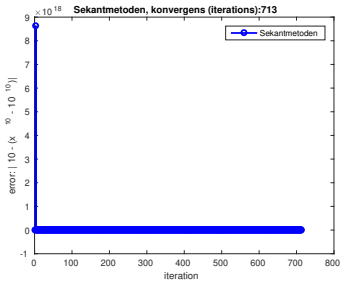
Exempel: halveringsmetoden, sekantmetoden och Newton's metod för $f(x) = x^{10} - 10^{10} = 0$

Startvärde i Newton's metod: $x(1) = 10^{10}$, tolerance = 10e-15. För halveringsmetoden: $[n, p] = [0, 10^{10}]$.

Startvärde i Sekantmetod: $x(1) = 5.0, x(2) = 7$.







Ickelinjära ekvationer (Konvergensordning)

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. "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 den totala körtiden och minnesbehovet.

- få komplexa iterationer
- många enkla iterationer

Ickelinjära ekvationer (Metodoberoende feluppskattningen)

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

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

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 $f'(0) = 0$. Nu ett exempel:

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

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

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

Ickelinjära ekvationer (Metodoberoende feluppskattningen)

I praktiken är det svårt att beräkna M, så en vettig approximation är $f(\hat{x})/f'(\hat{x})$. I Newtons metod får vi denna approximation på köpet:

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)} \Rightarrow x_j - x_{j+1} = \frac{f(x_j)}{f'(x_j)}$$

Om $x^* \approx x_{j+1}$ så är $f(\hat{x})/f'(\hat{x})$ en uppskattning av felet i x_j. Nu ett exempel med föregående problem, $f(x) = 1/x - 1/10$. Newtons metod lyder

$$x_{j+1} = x_j - \frac{1/x_j - 1/10}{-1/x_j^2}$$

Ickelinjära ekvationer (Metodoberoende feluppskattningen)

```
x = 1;                          % bad
for j = 1:10
   f  = 1 / x(j) - 0.1;         % f
   fp = -1 / x(j)^2;            % f'
   q  = f / fp;

   x(j + 1) = x(j) - q;         % update
   e(j)     = x(j) - 10;        % error
   a(j)     = q;                % approx. of error
end
fprintf( ...
  '        x          error       approx err\n')
disp([x(1:end-1)', e', a'])     % print a table
```


Ickelinjära ekvationer (Metodoberoende feluppskattningen)

x	error	approx err
1.0000e+00	-9.0000e+00	-9.0000e-01
1.9000e+00	-8.1000e+00	-1.5390e+00
3.4390e+00	-6.5610e+00	-2.2563e+00
5.6953e+00	-4.3047e+00	-2.4517e+00
8.1470e+00	-1.8530e+00	-1.5097e+00
9.6566e+00	-3.4337e-01	-3.3158e-01
9.9882e+00	-1.1790e-02	-1.1776e-02
1.0000e+01	-1.3901e-05	-1.3901e-05
1.0000e+01	-1.9323e-11	-1.9322e-11
1.0000e+01	-1.7764e-15	-1.3878e-15

Ickelinjära ekvationer (Avbrottskriterium)

I sekantmetoden får vi inte en sekvens av intervall som innerhåller roten. Metoden kan ju till och med divergera. så, hur vet vi när vi ska 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 man betyda att det går långsamt
- $|f(x_k)|$, noll i lösningen (tänk också på $|f(x_k)|/M$)

Första försöket: avsluta om ($|$ = eller):

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

maxit (max antal iterationer), tol_x och tol_f ges av användaren. Man kan givetvis kräva att $|x_k - x_{k-1}| \leq tol_x \;\&\; |f(x_k)| \leq tol_f$.

Ickelinjära ekvationer (Avbrottskriterium)

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:

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

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 till funktionen (och kanske derivatan) i ett ändligt antal punkter. Det finns oändligt många funktioner som går genom dessa punkter.

Ickelinjära ekvationer (Modifierad Newton)

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

Differensapproximation av J. Vi vill normalt inte 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 h (se övning):

$$f(x + he_i) \approx f(x) + hJe_i$$

eller

$$Je_i \approx \frac{f(x + he_i) - f(x)}{h}$$

där e_i är kolonn i i enhetsmatrisen.

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	5.0808e+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

Fixpunkter och lite teori

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

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

Om vi konvergerar 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 på 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.

Fixpunkter och lite teori

Tre saker kan inträffa:

1. Om $0 \leq x_0 \leq 1$ så konvergerar värdena mot 0. T.ex. $0.1, 0.01, 0.0001, ...$
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 mening 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") där g är kontinuerligt deriverbar. Om x_k konvergerar $x_k \to x^*$ gäller att $x^* = g(x^*)$. Vi kallar g en fixpunktsiteration och x^* en fixpunkt. Startar vi i en fixpunkt får vi tillbaka den.

Övningar

Sätt upp Newtons metod för följande problem:

1.

$$\sin(x) + \cos(y) = 2,$$
$$\cos(x^2)y + \sin(y^2)x = 3.$$

2.

$$(\cos(x))^2 \sin(y) = \sin(x^2),$$
$$(\sin(y))^2 \cos(x) = \cos(y^2).$$

Fixpunkter och lite teori

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

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

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

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

dvs. $f(x^*) = 0$ (antag enkelrot så att $f'(x^*) \neq 0$). Så fixpunkten är en lösning till vårt problem.

- Om vi ska lösa $f(x) = 0$ med fixpunktsiteration: $f(x) - x + x = 0$ och $f(x) + x = x$ och fixpunktsiteration är: $x_{k+1} = x_k + f(x_k)$.

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.

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

$$g(x^*) + (x_k - x^*)g'(\theta_k) - x^* = g'(\theta_k)(x_k - x^*),\ \theta_k \in (x_k, x^*)$$

Så

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

Ett steg till:

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

Alltså:

$$|x_k - x^*| = \underbrace{|g'(\theta_{k-1})|\,...\,|g'(\theta_1)|\,|g'(\theta_0)|}_{|g'(\theta)|}\,|x_0 - x^*| = |g'(\theta)||x_0 - x^*|.$$

Fixpunkter och lite teori

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

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

Följande villkor garanterar konvergens:

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

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

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 $|g'(\theta_0)| \le \lambda$ vilket medför att $x_1 \in I$. Induktion!
Normalt linjär konvergens; ju mindre $|g'(x^*)|$ desto snabbare konvergens

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

Fixpunkter och lite teori

Vad gäller för Newtons metod?

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

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

Innebär (minst) kvadratisk konvergens (inte att det konvergerar i ett steg). Låt oss troliggöra detta. Inför $\delta_k = x_k - x^*$. Vi får

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

eller

$$\delta_{k+1} = \delta_k - \frac{f(x^* + \delta_k)}{f'(x^* + \delta_k)} = \delta_k - \frac{f(x^*) + \delta_k f'(x^*) + \delta_k^2 f''(x^*)/2 + \ldots}{f'(x^*) + \delta_k f''(x^*) + \ldots}$$

så att

$$\delta_{k+1} = \frac{\delta_k^2 f''(x^*)/2 + \ldots}{f'(x^*) + \delta_k f''(x^*) + \ldots} \approx \frac{f''(x^*)}{2f'(x^*)}\delta_k^2$$

Konvergens för sekantmetod

Vad gäller för sekantmetod? Iterera: givet två startvärden x_0, x_1 sekantmetoden:

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

$$\begin{aligned} x_{k+1} &= x_k - f(x_k)\frac{x_{k-1} - x_k}{f(x_{k-1}) - f(x_k)} \\ &= \frac{x_k f(x_{k-1}) - x_k f(x_k) - f(x_k)x_{k-1} + f(x_k)x_k}{f(x_{k-1}) - f(x_k)} \\ &= \frac{x_k f(x_{k-1}) - f(x_k)x_{k-1}}{f(x_{k-1}) - f(x_k)} \end{aligned} \tag{17}$$

$$\begin{aligned} x_{k+1} - x^* &= \frac{x_k f(x_{k-1}) - f(x_k)x_{k-1}}{f(x_{k-1}) - f(x_k)} - x^* \\ &= \frac{x_k f(x_{k-1}) - f(x_k)x_{k-1} - x^*(f(x_{k-1}) - f(x_k))}{f(x_{k-1}) - f(x_k)} \end{aligned} \tag{18}$$

och kan skrivas som

$$
\begin{aligned}
x_{k+1} - x^* &= \frac{x_k f(x_{k-1}) - f(x_k)x_{k-1} - x^* f(x_{k-1}) + x^* f(x_k)}{f(x_{k-1}) - f(x_k)} \\
&= \frac{(x_k - x^*)f(x_{k-1}) - f(x_k)(x_{k-1} - x^*)}{f(x_{k-1}) - f(x_k)} \\
&= \frac{(x_k - x^*)f(x^* + x_{k-1} - x^*) - f(x^* + x_k - x^*)(x_{k-1} - x^*)}{f(x^* + x_{k-1} - x^*) - f(x^* + x_k - x^*)} \\
&= \frac{(x_k - x^*)[f(x^*) + (x_{k-1} - x^*)f'(\xi_{k-1})]}{f(x^* + x_{k-1} - x^*) - f(x^* + x_k - x^*)} \\
&- \frac{[f(x^*) + (x_k - x^*)f'(\xi_k)](x_{k-1} - x^*)}{f(x^* + x_{k-1} - x^*) - f(x^* + x_k - x^*)} \\
&= \frac{(x_k - x^*)(x_{k-1} - x^*)f'(\xi_{k-1})}{f(x^*) + (x_{k-1} - x^*)f'(\xi_{k-1})) - f(x^*) - (x_k - x^*)f'(\xi_k)} \\
&- \frac{[(x_{k-1} - x^*)(x_k - x^*)f'(\xi_k)]}{f(x^*) + (x_{k-1} - x^*)f'(\xi_{k-1})) - f(x^*) - (x_k - x^*)f'(\xi_k)},
\end{aligned}
$$

$\xi_k \in (x_k, x^*),\ \xi_{k-1} \in (x_{k-1}, x^*).$

(19)

Vi får:

$$
\begin{aligned}
x_{k+1} - x^* &= \frac{(x_k - x^*)(x_{k-1} - x^*)f'(\xi_{k-1})}{(x_{k-1} - x^*)f'(\xi_{k-1})) - (x_k - x^*)f'(\xi_k)} \\
&- \frac{[(x_{k-1} - x^*)(x_k - x^*)f'(\xi_k)]}{(x_{k-1} - x^*)f'(\xi_{k-1})) - (x_k - x^*)f'(\xi_k)} \\
&= \frac{(x_k - x^*)(x_{k-1} - x^*)[f'(\xi_{k-1}) - f'(\xi_k)]}{(x_{k-1} - x^*)f'(\xi_{k-1})) - (x_k - x^*)f'(\xi_k)} \\
&= (x_k - x^*)(x_{k-1} - x^*) \cdot M \\
&\xi_k \in (x_k, x^*),\ \xi_{k-1} \in (x_{k-1}, x^*).
\end{aligned}
$$

(20)

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. Vi kan skriva om det som:

$$|x_{k+1} - x^*| \approx C|x_k - x^*|^r \approx |x_k - x^*||x_{k-1} - x^*| \cdot |M|$$

eller

$$
\begin{aligned}
C|x_k - x^*|^r &\approx |x_k - x^*||x_{k-1} - x^*| \cdot |M|, \\
\frac{|x_k - x^*|^r}{|x_k - x^*|} &\approx |M|/|C| \cdot |x_{k-1} - x^*|, \\
|x_k - x^*|^{r-1} &\approx |M|/|C| \cdot |x_{k-1} - x^*|, \\
|x_k - x^*| &\approx \underbrace{(|M|/|C|)^{1/r-1}}_{B:=Const} \cdot (|x_{k-1} - x^*|)^{1/r-1}, \\
|x_k - x^*| &\approx B \cdot (|x_{k-1} - x^*|)^{1/r-1}
\end{aligned}
\tag{21}
$$

Vi tar $1/(r-1) = r$ konvergensordning (enligt definition för konvergensordningen). Darför $r(r-1) = 1; r^2 - r - 1 = 0$ och vi tar bara $r > 0$, eller $r = (1+\sqrt{5})/2 \approx 1.618 > 0$ som är konvergensordning för sekantmetoden (superlinjär).

Fixpunkter och lite teori

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. $x_0 = 10^{-1}, x_1 = 10^{-2}, x_2 = 10^{-4}$...
- $g(x) = x/2$. Fixpunkter: $x^* = x^*/2$ så $x^* = 0$. Konvergens? $g'(x^*) = 1/2$. Linjär konvergens: $x_0 = 1, x_1 = 1/2, x_2 = 1/4, ...$
- $g(x) = \cos x$. 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.

Fixpunkter och lite teori

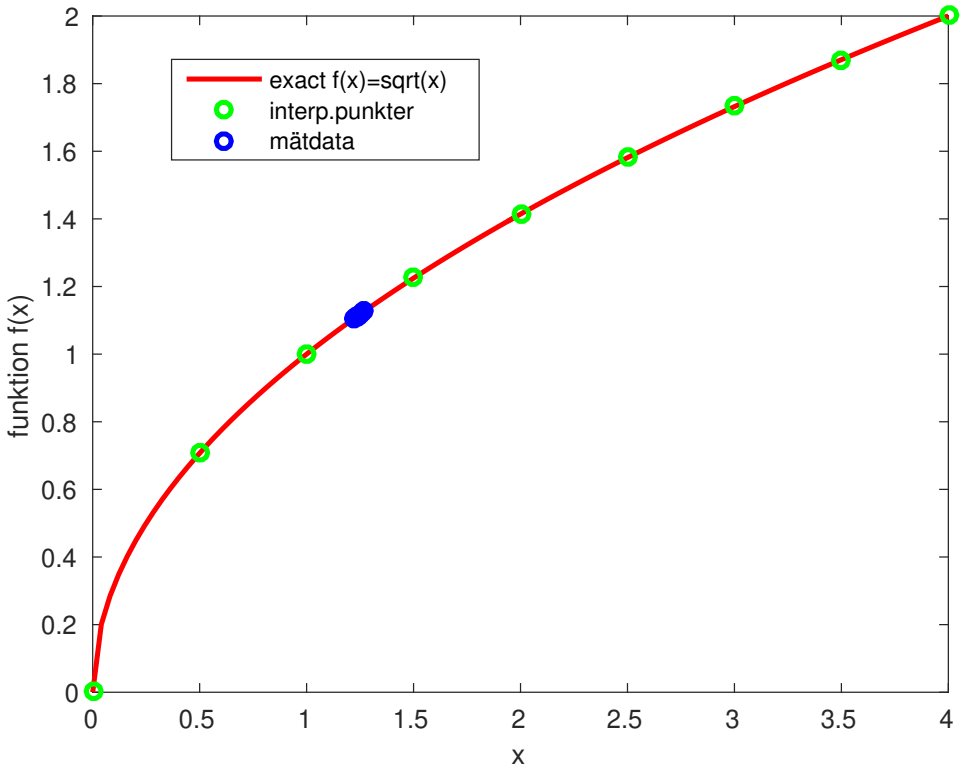
Konvergens? $g'(x) = 2x/\alpha + 1$. Tar vi t.ex. $\alpha = -3$ så får vi rätt snabb konvergens ty $|g'(\sqrt{2})\| = |-2\sqrt{2}/3 + 1| \approx 0.05719$.

```
>> 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  -4.6e-09  -2.4e-10  -1.4e-11

>> abs(d(2:end) ./ d(1:end-1))
 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

För i tiden gällde räknesticka och tabeller. Beräkna $\sqrt{1.244}$ givet en tabel över $y = \sqrt{t}$, y-värdena är givna med fem siffror, och $t = 0, 0.01, 0.02, ..., 9.99, 10.00$.
Mer realistiskt, nu för tiden, vore en tabell, $t_k, y_k, k = 1, ..., n$ där vi av någon anledning inte kan beräkna $y(t)$ för alla t (mättekniska problem, gamla data).
Hur ska vi gå tillväga? I skoltabellen fanns röda tal mellan y-värdena, differenser, för att underlätta linjär interpolation

t	y
	...
1.22	1.1045
1.23	1.1091
1.24	**1.1136**
1.25	**1.1180**
1.26	1.1225
1.27	1.1269



Interpolation

Givet (t_k, y_k) och (t_{k+1}, y_{k+1}) söker vi y-värden svarande mot t, $t_k < t < t_{k+1}$. Linjär interpolation ger

$$y \approx y_k + (t - t_k)\frac{y_{k+1} - y_k}{t_{k+1} - t_k}$$

Så i exemplet (vi ska räkna $y = \sqrt{t}$ för $t = 1.244, t_{k+1} = 1.25, t_k = 1.24, y(t_k) = y_k = 1.1136, y(t_{k+1}) = y_{k+1} = 1.1180, y_{k+1} - y_k = 0.0044$):

$$\sqrt{1.244} \approx 1.1136 + (1.244 - 1.24)\frac{0.0044}{1.25 - 1.24} = 1.11536$$

Felet $\approx -1.3 \cdot 10^{-5}$.

Andra tillämpningar som nyttjar interpolation är kvadratur (integration), lösning av randvärdesproblem, förenkling av funktioner, härledning av metoden (t.ex. sekantmetoden).

Interpolation

Allmänt har vi $(t_k, y_k), k = 1, ..., m$ med $t_1 < t_2 < ... < t_m$ och vill hitta en funktion (polynom i denna kurs), p, så att $p(t_k) = y_k, k = 1, ..., m$. Ibland lägger man dessutom på krav på derivator, s.k. Hermite-interpolation.

Låt oss anta att det finns en underliggande 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 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:

Interpolation

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

så att

$$p(t) = \underbrace{\frac{t_2 y_1 - t_1 y_2}{t_2 - t_1}}_{x_1} + \underbrace{\frac{y_2 - y_1}{t_2 - t_1}}_{x_2} t = y_1 + (t - t_1)\frac{y_2 - y_1}{t_2 - t_1}$$

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

$$\underbrace{p(t)}_{\text{beräknad}} - \underbrace{f(t)}_{\text{exakt}} = \underbrace{f(t_1) + \delta_1}_{y_1} + (t - t_1)\frac{\underbrace{f(t_2) + \delta_2}_{y_2} - \underbrace{(f(t_1) + \delta_1)}_{y_1}}{t_2 - t_1} - f(t) =$$

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

Interpolation

Låt oss införa de två polynomen p_f och p_δ sådana att

$$p_f(t_1) = f(t_1) = f(t_1) + \underbrace{(t_1 - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1}}_{0}, \tag{22}$$

$$p_f(t_2) = f(t_2) = f(t_1) + \underbrace{(t_2 - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1}}_{f(t_2) - f(t_1)} \tag{23}$$

resp.

$$p_\delta(t_1) = \delta_1 = \delta_1 + \underbrace{(t_1 - t_1)\frac{\delta_2 - \delta_1}{t_2 - t_1}}_{0}, \tag{24}$$

$$p_\delta(t_2) = \delta_2 = \delta_1 + \underbrace{(t_2 - t_1)\frac{\delta_2 - \delta_1}{t_2 - t_1}}_{\delta_2 - \delta_1} \tag{25}$$

Då är tydligen $p = p_f + p_\delta$.

Vi vet redan:

$$\underbrace{p(t)}_{\textit{beräknad}} - \underbrace{f(t)}_{\textit{exakt}} = \underbrace{f(t_1) + \delta_1}_{y_1} + (t - t_1)\frac{\underbrace{f(t_2) + \delta_2}_{y_2} - \underbrace{(f(t_1) + \delta_1)}_{y_1}}{t_2 - t_1} - f(t) =$$

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

Detta kan man direkt se från det linjära ekvationssystemet, lösningen x beror ju linjärt på högerledet.

$$\underbrace{p(t)}_{\textit{beräknad}} - \underbrace{f(t)}_{\textit{exakt}} = p_f(t) + p_\delta(t) - f(t) = \underbrace{p_f(t) - f(t)}_{\textit{approxim.fel}} + \underbrace{p_\delta(t)}_{\textit{avrundningsfel}}$$

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. $p_\delta(t)$ interpolerar (avrundnings) felen i tabellvärdena.

Interpolation (Felets utseende)

Låt oss nu uppskatta felet $e(t) = f(t) - p_f(t)$ (denna härledning kan rätt lätt 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 $g(t) = 0$:

$$g(t) = e(t) - \frac{(t - t_1)(t - t_2)}{(t - t_1)(t - t_2)} e(t) = 0, \tag{26}$$

$$g(t_1) = e(t_1) - \frac{(t_1 - t_1)(t_1 - t_2)}{(t - t_1)(t - t_2)} e(t) = 0, \tag{27}$$

$$g(t_2) = e(t_2) - \frac{(t_2 - t_1)(t_2 - t_2)}{(t - t_1)(t - t_2)} e(t) = 0. \tag{28}$$

g har alltså tre distinkta nollställen varför, enligt medelvärdessatsen, $g'(z)$ har två distinkta nolställen. $g''(z)$ 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 $p_f \leq 1$):

$$g''(z) = e''(z) - \frac{2e(t)}{(t - t_1)(t - t_2)} = f''(z) - \underbrace{p_f''(z)}_{=0} - \frac{2e(t)}{(t - t_1)(t - t_2)}.$$

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

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

$$g''(\theta) = f''(\theta) - \frac{2e(t)}{(t - t_1)(t - t_2)} = 0,$$

och får

$$e(t) = \underbrace{f(t) - p_f(t)}_{\textit{approxim.fel}} = \frac{f''(\theta)}{2}(t - t_1)(t - t_2),$$

var $\theta \in (t, t_1, t_2)$ (det minsta intervall som innehåller t, t_1, t_2).

Interpolation (Felets utseende)

Felet är noll då $t = t_1$ eller $t = t_2$. Faktorn $(t - t_1)(t - t_2)$ mäter hur långt t ligger från någondera ändpunkten. Om $t_1 < t < t_2$ så antar $(t - t_1)(t - t_2)$ sitt mest negativa värde då $t = (t_1 + t_2)/2$.

$f''(\theta)$ är ett mått på krökningen hos f, dvs. hur mycket f "buktar ut" från en rät linje. Om krökningen är noll, $f''(\theta) = 0$, så är f linjär och felet är noll.

Krökningsradien, $R = |(1 + (f'(\theta))^2)^{3/2}/f''(\theta)|$. Krökningen är $1/R$.

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:

Interpolation (Felets utseende)

Sats

Om p_f interpolerar f för de n $t-$värdena $t_1 < t_2 < ... < t_n$ så gäller att

$$\underbrace{f(t) - p_f(t)}_{approxim.fel} = \frac{f^{(n)}(\theta)}{n!}(t - t_1)(t - t_2)...(t - t_n)$$

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

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

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

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

Interpolation (Felets utseende)

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 (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.

Så det kan löna sig att höja gradtalet förutsatt att tabellvärdena inte har för stora fel. Polynom av höga gradtal är dock inte lätthanterliga, mer om detta senare.
Kan vi använda polynomet för att extrapolera (gå utanför $[t_1, t_n]$)? 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!

Interpolation (Entydighet)

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

Sats

Interpolationsproblemet: hitta ett polynom p med grad högst $n-1$ sådant att $p(t_k) = y_k, k = 1, 2, .., n$, där alla (t_k, y_k) är givna och $t_1 < t_2 < ... < t_n$.

Låt oss anta att existensen är given och studera entydigheten. Antag att det finns ett annat polynom q av grad $\leq n-1$ som interpolerar data. Det gäller då att $p(t_k) - q(t_k) = 0, k = 1, 2, .., m$. Polynomet $p - q$ av grad $\leq n-1$ har alltså n distinkta nollställen. $p - q$ måste således vara nollpolynomet och $p = q$. Polynomet behöver inte alltid ha grad $n-1$. Om vi t.ex. väljer punkterna $y_k = t_k^2, k = 1, 2, ..., 10$ så klarar vi oss med $p(t) = t^2$ fastän $n = 10$.

Interpolation (Lagranges form)

Nu till existensen. Den går att bevisa på flera sätt. Vi kommer att använda ett konstruktivt bevis. Antag att vi har $n = 3$ punkter.

Sats

Här följer interpolationspolynomet på Lagranges form i 3 punkter:

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

En fördel med denna form på polynomet är att den är lätt att ställa upp och att 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- och overflow om man inte tänker sig för.

Interpolation (Lagranges form) i $k+1$ datapunkter

För $k+1$ datapunkter $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$ sådana att $x_j \neq x_k$, interpolationspolynomet på Lagranges form är linjär kombination

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

med Lagrange-baspolynomier

$$\ell_j(x) := \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x-x_0)}{(x_j-x_0)} \cdots \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)},$$

$0 \leq j \leq k$.

Interpolation (Lagranges form)

För 9 punkter:
$(x_1, y_1), \ldots, (x_j, y_j), \ldots, (x_9, y_9)$:

$$L(x) := \sum_{j=1}^{9} y_j \ell_j(x) = y_1\ell_1(x) + \ldots + y_9\ell_9(x)$$

var Lagrange-baspolynomier är:

$$\ell_j(x) := \prod_{\substack{1 \le m \le 9 \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$$

var $1 \le j \le 9$.

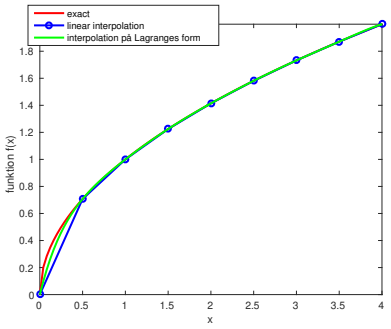
$$l_1 = \frac{(x-x_2)(x-x_3)(x-x_4)(x-x_5)(x-x_6)(x-x_7)(x-x_8)(x-x_9)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)(x_1-x_5)(x_1-x_6)(x_1-x_7)(x_1-x_8)(x_1-x_9)};$$

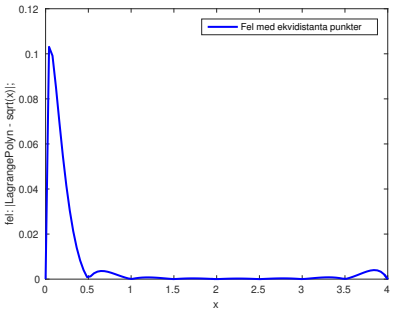
$$l_2 = \frac{(x-x_1)(x-x_3)(x-x_4)(x-x_5)(x-x_6)(x-x_7)(x-x_8)(x-x_9)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)(x_2-x_5)(x_2-x_6)(x_2-x_7)(x_2-x_8)(x_2-x_9)}, \ \ldots,$$

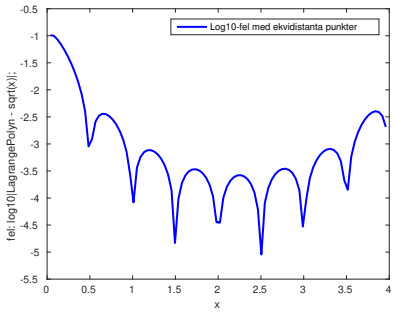
$\ldots$,

$$l_9 = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)(x-x_5)(x-x_6)(x-x_7)(x-x_8)}{(x_9-x_1)(x_9-x_2)(x_9-x_3)(x_9-x_4)(x_9-x_5)(x_9-x_6)(x_9-x_7)(x_9-x_8)}.$$

Interpolation i Lagrange form i 9 punkter för $f(x) = \sqrt{x}$







$e = |L(x) - \sqrt{x}|$ $\quad$ $e = log10|L(x) - \sqrt{x}|$

Example

Interpolera funktionen $f(t) = t^2$ på $1 \leq t \leq 3$, i punkter:
$t_1 = 1, t_2 = 2, t_3 = 3$.
Svar:
Vi har: $y_1 = f(t_1) = 1, y_2 = f(t_2) = 4, y_3 = f(t_3) = 9$.
Här följer interpolationspolynomet på Lagranges form:

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

$$L(t) = 1 \cdot \frac{t-2}{1-2} \cdot \frac{t-3}{1-3} + 4 \cdot \frac{t-1}{2-1} \cdot \frac{t-3}{2-3} + 9 \cdot \frac{t-1}{3-1} \cdot \frac{t-2}{3-2}$$

$$= t^2.$$

Övning

Interpolera funktionen $f(t) = t^3$ på $1 \leq t \leq 3$, i punkter:
$t_1 = 1, t_2 = 2, t_3 = 3$.
Här följer interpolationspolynomet på Lagranges form:

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

Example

Interpolera funktionen $f(t) = t^3$ på $1 \leq t \leq 3$, i punkter:
$t_1 = 1, t_2 = 2, t_3 = 3$.
Svar:
Vi har: $y_1 = f(t_1) = 1, y_2 = f(t_2) = 8, y_3 = f(t_3) = 27$.
Här följer interpolationspolynomet på Lagranges form:

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

$$L(t) = 1 \cdot \frac{t-2}{1-2} \cdot \frac{t-3}{1-3} + 8 \cdot \frac{t-1}{2-1} \cdot \frac{t-3}{2-3} + 27 \cdot \frac{t-1}{3-1} \cdot \frac{t-2}{3-2}$$

$$= 6t^2 - 11t + 6.$$

Interpolation (Vandermondematrisen)

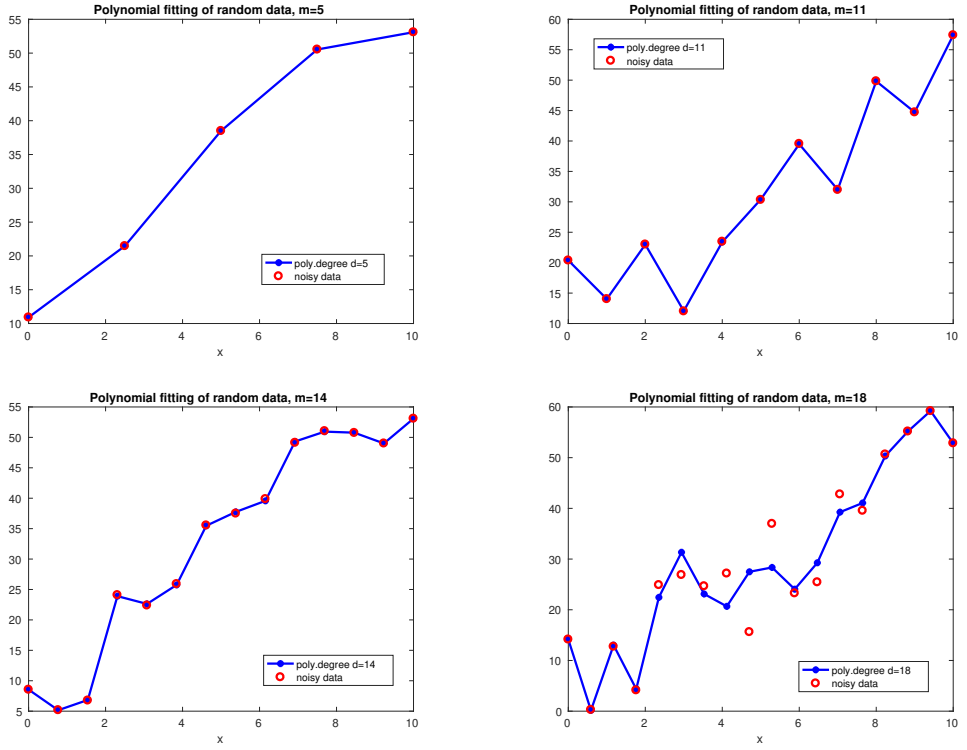
Ett annat sätt att konstruera polynomet är att sätta upp ett ekvationssystem som vi gjorde i det linjära fallet. Så vi ansätter $p(t) = x_1 + x_2 t + x_3 t^2$. Interpolationsvillkoren ger då:

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

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 utseende. Normalt har man dock inte speciellt höga gradtal. Det är dock billigt och stabilt att beräkna $p(t)$.

Exempel: Vandermondematrisen



Interpolation (Horners metod)

Det är dock billigt och stabilt att beräkna polynom $p(t) = \sum_{i=1}^{d} x_i t^{i-1}$. Man använder normalt Horners metod för detta. Exempel med $n = 4$.

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

Detta skrivs lämpligen i en loop, men jag har använd sekvensiell kod: $p = x_4$, $p = x_3 + tp$, $p = x_2 + tp$, $p = x_1 + tp$.
Man kan se Vandermonte-härledningen som ett specialfall av följande. Vi ansätter

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

ϕ_k kallas basfunktion och i Vandermonte-matrisen använder vi $\phi_k(t) = t^{k-1}$.
Ett problem med Vandermontematriser är att de kan bli illakonditionerade.

Interpolation (Horners metod)

Exempel

Antag $n = 4$ och tag t-värdena
$t_1 = 0.1 = 10^{-1}, t_2 = 0.2 = 2 \cdot 10^{-1}, t_3 = 0.3 = 3 \cdot 10^{-1}$ och
$t_4 = 0.4 = 4 \cdot 10^{-1}$. 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 $\phi_k(t)$ för $p(t)$:

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

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 $[-1, 1]$:

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

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

Interpolation (Newtons form)

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 det möjligt att lägga till nya punkter utan att börja om med polynomberäkningen.

Sats

Den allmänna Newtons formen är:

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

Interpolation (Newtons form)

Låt oss se på specialfallet $n = 3$.

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

Observera:

$$y_1 = p(t_1) = x_1 + x_2(t_1 - t_1) + x_3(t_1 - t_1)(t_1 - t_2) = x_1, \tag{29}$$

$$y_2 = p(t_2) = x_1 + x_2(t_2 - t_1) + x_3(t_2 - t_1)(t_2 - t_2) = x_1 + x_2(t_2 - t_1), \tag{30}$$

$$y_3 = p(t_3) = x_1 + x_2(t_3 - t_1) + x_3(t_3 - t_1)(t_3 - t_2). \tag{31}$$

Vi får det undertriangulära systemet:

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

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.

Interpolation

Nu ett exempel där vi konstruerar polynomet med de tre metoderna.

Exempel

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

Lösning: $p(t) = [1,4,11]^T$, $t_1 = 1, t_2 = 2, t_3 = 3$.
1) Vandermontes form. Ansätt $p(t) = x_1 + x_2 t + x_3 t^2$

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 11 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 2 \\ -3 \\ 2 \end{bmatrix}$$

Så $p(t) = 2 - 3t + 2t^2$ eller $p(t) = 2t^2 - 3t + 2$.

2) Langranges 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) $p(t) = 2t^2 - 3t + 2$.

Övning

Finn $p(t)$ i Vandermontes form som interpolerar funktionen $f(t) = t^3$ på $1 \leq t \leq 4$, i punkter: $t_1 = 1, t_2 = 2, t_3 = 3, t_4 = 4$.
Notera: Vandermontes matris för 4 punkter:

$$\begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \\ 1 & t_4 & t_4^2 & t_4^3 \end{bmatrix}$$

Interpolation

Exempel (fortsättning)

Notera: $p(t) = [1, 4, 11]^T$, $t_1 = 1, t_2 = 2, t_3 = 3$.

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 $p(t) = 2t^2 - 3t + 2$.

Övning

Finn $p(t)$ i Newtons form som interpolerar funktionen $f(t) = t^3$ på $1 \leq t \leq 4$, i punkter: $t_1 = 1, t_2 = 2, t_3 = 3, t_4 = 4$.
Notera: Newtons form:

$$p(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + x_4(t-t_1)(t-t_2)(t-t_3).$$

Interpolation (Newtons form)

Svar

$$p(t) = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + x_4(t - t_1)(t - t_2)(t - t_3).$$

Observera:

$$\begin{aligned}
p(t_1) &= x_1 + x_2(t_1 - t_1) + x_3(t_1 - t_1)(t_1 - t_2) + x_4(t_1 - t_1)(t_1 - t_2)(t_1 - t_3),\\
p(t_2) &= x_1 + x_2(t_2 - t_1) + x_3(t_2 - t_1)(t_2 - t_2) + x_4(t_2 - t_1)(t_2 - t_2)(t_2 - t_3),\\
p(t_3) &= x_1 + x_2(t_3 - t_1) + x_3(t_3 - t_1)(t_3 - t_2) + x_4(t_3 - t_1)(t_3 - t_2)(t_3 - t_3),\\
p(t_4) &= x_1 + x_2(t_4 - t_1) + x_3(t_4 - t_1)(t_4 - t_2) + x_4(t_4 - t_1)(t_4 - t_2)(t_4 - t_3).
\end{aligned}$$

Interpolation (Newtons form)

Vi får det undertriangulära systemet med $y_i = p(t_i),\ i = 1, 2, 3, 4$: :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_2 - t_1 & 0 & 0 \\ 1 & t_3 - t_1 & (t_3 - t_1)(t_3 - t_2) & 0 \\ 1 & t_4 - t_1 & (t_4 - t_1)(t_4 - t_2) & (t_4 - t_1)(t_4 - t_2)(t_4 - t_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix},$$

som vi skriver om med $t_1 = 1, t_2 = 2, t_3 = 3, t_4 = 4$ och $f(t) = t^3$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 - 1 & 0 & 0 \\ 1 & 3 - 1 & (3 - 1)(3 - 2) & 0 \\ 1 & 4 - 1 & (4 - 1)(4 - 2) & (4 - 1)(4 - 2)(4 - 3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1^3 \\ 2^3 \\ 3^3 \\ 4^3 \end{bmatrix}$$

som ju är enkelt att lösa (framåtsubstitution) för att få
$x_1 = 1, x_2 = 7, x_3 = 6, x_4 = 1$ så att

$$p(t) = 1 + 7(t - 1) + 6(t - 1)(t - 2) + (t - 1)(t - 2)(t - 3).$$

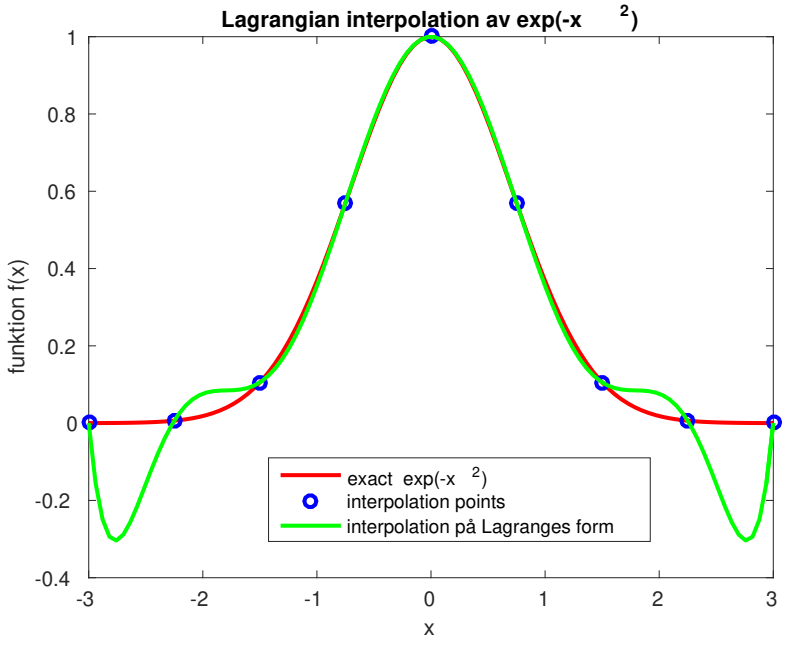
Interpolation (Problem med interpolation)

För vissa funktioner kan interpolationsfelet bli stort i intervallets ändar (t.ex för $f(t) = e^{-t^2}$). Detta problem kan förvärras när antalet punkter ökar (Runges fenomen). p behöver inte alltid konvergera mot f, utan felet kan öka med ökande antal punkter.

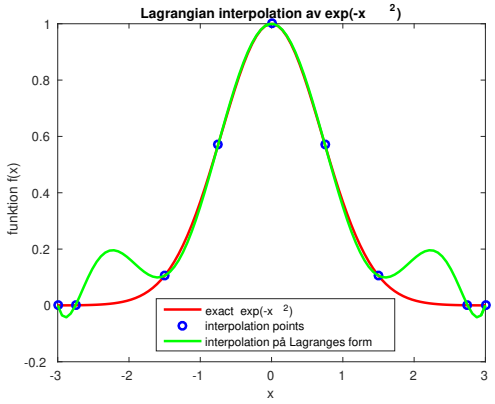
Det är inte ovanligt att polynom av högt gradtal svänger kraftigt när man använder ekvidistant interpolation (samma avstånd mellan t_k-värdena.

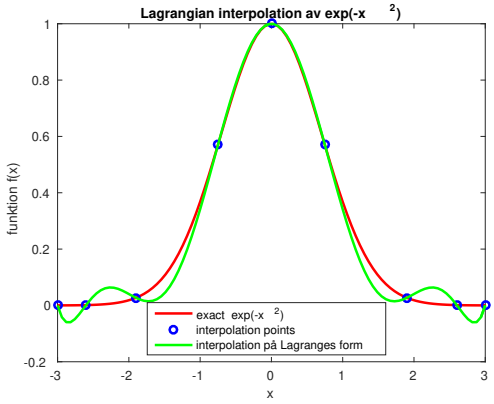
Vi kan försöka att "hålla nere" polynomet i ändarna genom att lägga punkterna tätare där. Då svänger polynomet mindre.

Lagrange interpolation i 9 ekvidistanta punkter för $f(x) = e^{-x^2}$



Interpolation i Lagrange form i 9 punkter för $f(x) = e^{-x^2}$





Interpolation (Problem med interpolation)

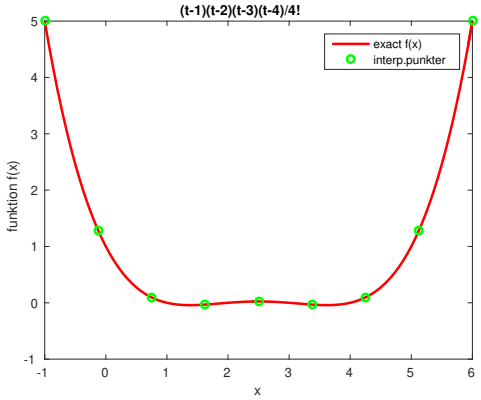
Vad är ett bra sätt att välja punkterna (om vi får välja)? Låt oss studera felets utseende igen (vi kan tänka oss exakta data, så att $p_f = p$).

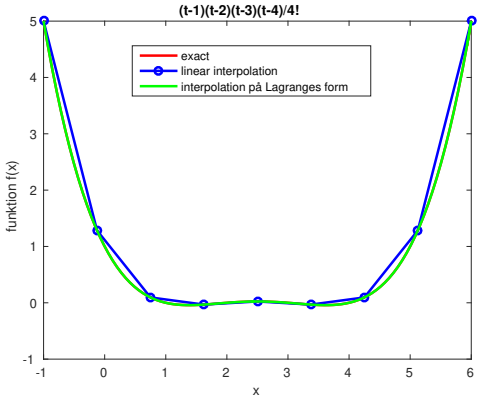
$$\underbrace{p(t)}_{\text{beräknad}} - \underbrace{f(t)}_{\text{exakt}} = \frac{f^{(n)}(\theta)}{n!}(t-t_1)(t-t_2)...(t-t_n)$$

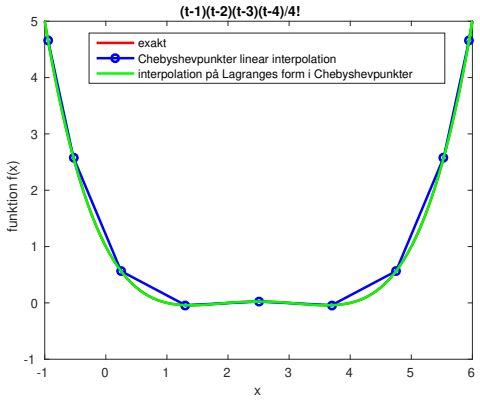
där $\theta \in (t, t_1, t_2, ..., t_n)$. Antag att $|f^{(n)}(\theta)| \leq M$ för alla $\theta \in (t_1, t_2, ..., t_n)$. Vi har då

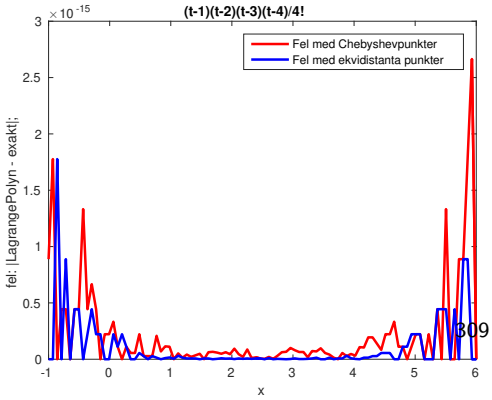
$$|f(t) - p(t)| \leq \frac{M}{n!}|(t-t_1)(t-t_2)...(t-t_n)|$$

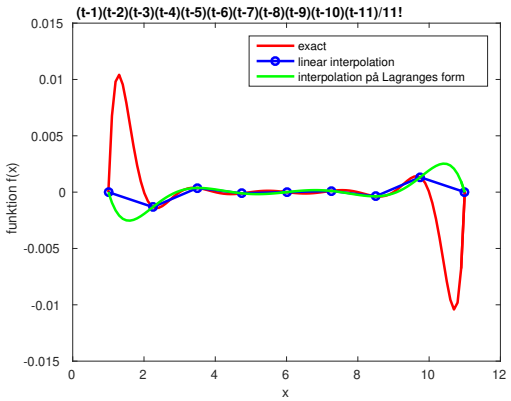
Låt oss specialstudera funktionen $(t-t_1)(t-t_2)...(t-t_n)/n!$. Den växer snabbt utanför $[t_1, t_n]$. Extrapolation är farligt!

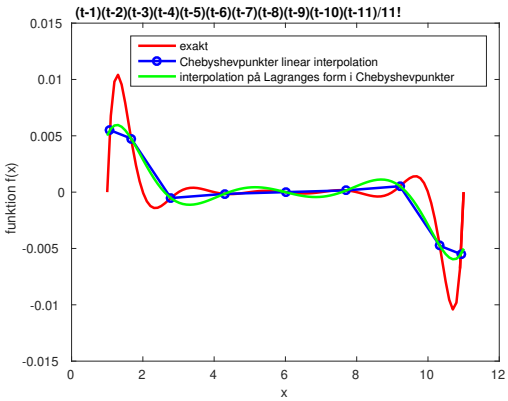

(t-1)(t-2)(t-3)(t-4)/4!
exact f(x)
interp.punkter
funktion f(x)
x

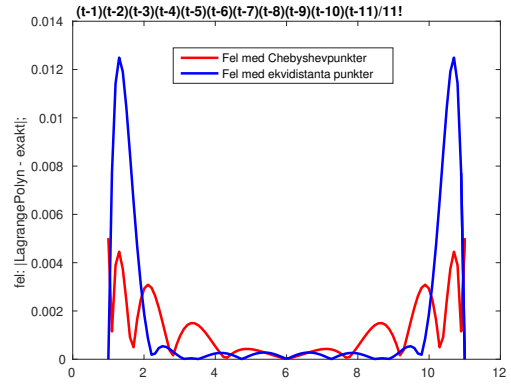

(t-1)(t-2)(t-3)(t-4)/4!
exact
linear interpolation
interpolation på Lagranges form
funktion f(x)
x

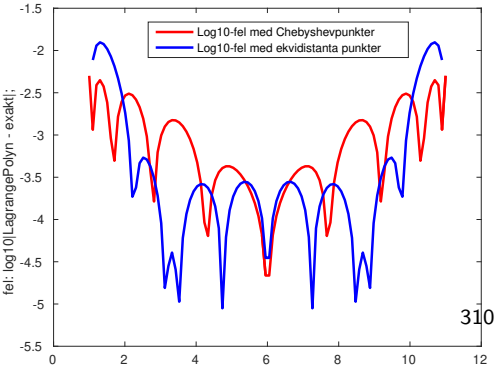

(t-1)(t-2)(t-3)(t-4)/4!
exakt
Chebyshevpunkter linear interpolation
interpolation på Lagranges form i Chebyshevpunkter
funktion f(x)
x


(t-1)(t-2)(t-3)(t-4)/4!
Fel med Chebyshevpunkter
Fel med ekvidistanta punkter
fel: |LagrangePolyn - exakt|;
x


(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)(t-7)(t-8)(t-9)(t-10)(t-11)/11!
exact
linear interpolation
interpolation på Lagranges form
funktion f(x)
x


(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)(t-7)(t-8)(t-9)(t-10)(t-11)/11!
exakt
Chebyshevpunkter linear interpolation
interpolation på Lagranges form i Chebyshevpunkter
funktion f(x)
x


(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)(t-7)(t-8)(t-9)(t-10)(t-11)/11!
Fel med Chebyshevpunkter
Fel med ekvidistanta punkter
fel: |LagrangePolyn - exakt|;


Log10-fel med Chebyshevpunkter
Log10-fel med ekvidistanta punkter
fel: log10|LagrangePolyn - exakt|;

Interpolation (Chebyshevpunkter)

De så kallande Chebyshevpunkterna har egenskapen att de gör det maximala värdet av $|(t-t_1)(t-t_2)...(t-t_n)|$ så litet som möjligt.

Sats

Om $t_k, t \in [-1,1],\ k = 1,2,...,n$ *gäller det att*

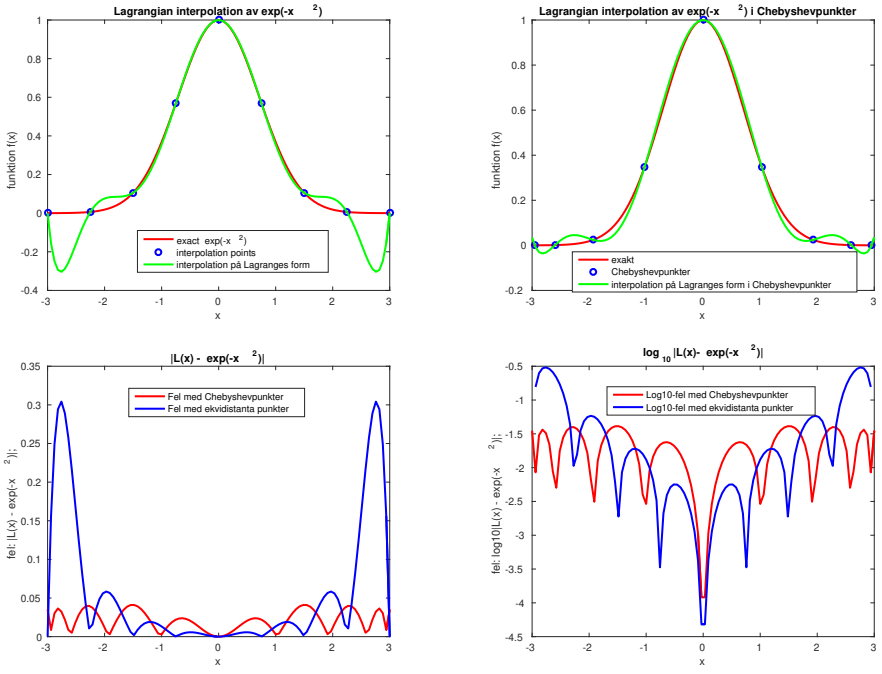
$$\max_{-1 \le t \le 1} |(t-t_1)(t-t_2)...(t-t_n)|$$

minimeras då

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

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

Interpolation i Lagrange form i 9 Chebyshevpunkter för $f(x) = e^{-x^2}$



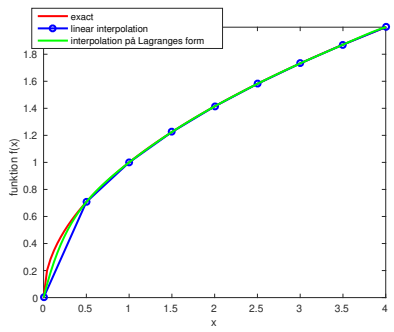
Interpolation (Chebyshevpunkter)

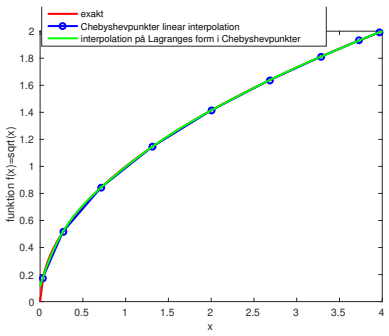
Om en funktion har $n+1$ antal kontinuerliga derivator så kan den utvecklas i en Taylorutveckling:

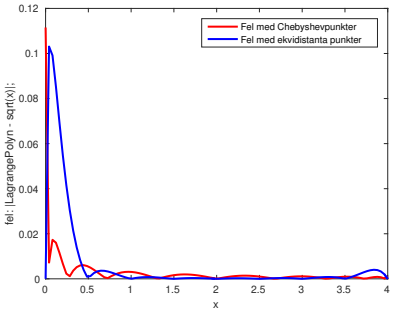
$$f(t) = f(a) + \frac{f'(a)}{1!}(t-a) + \frac{f''(a)}{2!}(t-a)^2 + ... + \frac{f^{(n)}(a)}{n!}(t-a)^n + R(t)$$

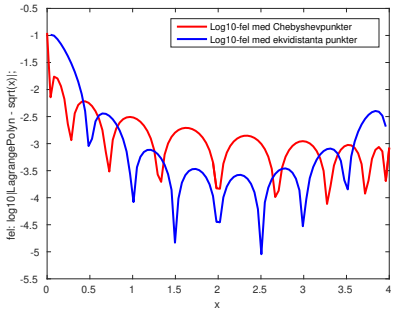
där resttermen $R(t) = c(\xi)(t-a)^{n+1}$, $\xi \in (a,t)$ och $|c(\xi)|$ är uppåt begränsad. Detta innebär att en sådan funktion (som har Taylorutveckling) liknar ett polynom på ett tillräckligt litet intervall.
Om inte alla $f^{(k)}(a) = 0,\ k = 0, 1, ..., n$ kan vi göra $R(t)$ godtyckligt liten jämfört med resten av Taylorutveckningen, genom att ta $|t-a|$ tillräckligt litet. På ett stort intervall behöver inte funktionen likna ett polynom.

Interpolation för $f(x) = \sqrt{x}$









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

Interpolation (Chebyshevpunkter)

När t ligger i ett annat intervall, $[\alpha, \beta]$ säg, får vi göra en linjär avbilding $kx + b$ av Chebyshevpunkterna $[-1, 1]$ till detta intervall $[\alpha, \beta]$:

$$\begin{aligned} k \cdot (-1) + b &= \alpha, \\ k \cdot 1 + b &= \beta, \end{aligned}$$

då

$$\begin{aligned} b &= \alpha + k, \\ k \cdot 1 + \alpha + k &= \beta, \end{aligned}$$

och från andra ekvation i systemet ovan har vi

$$\begin{aligned} k &= \frac{\beta - \alpha}{2}, \\ b &= \alpha + k = \alpha + \frac{\beta - \alpha}{2} = \frac{\alpha + \beta}{2}, \end{aligned}$$

och linjär avbilding av Chebyshevpunkterna till intervall $[\alpha, \beta]$ är:

Interpolation (Chebyshevpunkter)

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

så de transformerade Chebyshevpunkterna

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

blir

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

Ibland är det ändå problem. Det kan tänkas att M, begränsningen av $|f^{(n)}(\theta)|$ ej existerar.

Example

$f(t) = \sqrt{t}$ på intervallet $[0, 3]$. Redan $f'(0)$ är ju obegränsad (eftersom $f'(t) = (\sqrt{t})'_t = 1/(2\sqrt{t})$), man säger att derivatan har en singularitet. I vissa fall visar sig singulariteten först i högre derivator (t.ex. $f(t) = t^{5/2}$).

Övning

Vi bestämmer interpolationspolynomet, p_n, på $[0,1]$ som interpolerar e^t i punkterna $0 = t_1 < t_2 < ... < t_n = 1$. Visa att oavsett hur vi väljer t_k-punkterna (i övrigt) så gäller:

$$\lim_{n\to\infty} \max_{0\le t\le 1} |e^t - p_n(t)| = 0.$$

Visa att om vi väljer Chebyshevpunkterna så gäller att:

$$\max_{0\le t\le 1} |e^t - p_n(t)| \le \frac{e}{n!2^{2n-1}}.$$

Svar : vi vet att

$$\underbrace{p_n(t)}_{\text{beräknad}} - \underbrace{f(t)}_{\text{exakt}} = \frac{f^{(n)}(\theta)}{n!}(t-t_1)(t-t_2)...(t-t_n)$$

där $\theta \in (t, t_1, t_2, ..., t_n)$. Vi vet att $(e^t)^{(n)} = e^t$ och $|t - t_k| \le 1$ då

$$\underbrace{p_n(t)}_{\text{beräknad}} - \underbrace{e^t}_{\text{exakt}} = \frac{e^t(\theta)}{n!}(t-t_1)(t-t_2)...(t-t_n) \le \frac{e}{n!}(t-t_1)(t-t_2)...(t-t_n) \le \frac{e}{n!}.$$

Övning

$$\underbrace{p_n(t)}_{\text{beräknad}} - \underbrace{e^t}_{\text{exakt}} = \frac{e^t(\theta)}{n!}(t-t_1)(t-t_2)...(t-t_n) \le \frac{e}{n!}(t-t_1)(t-t_2)...(t-t_n) \le \frac{e}{n!}.$$

Observera att det ger oss ett konvergensresultat för varje funktion vars alla derivator är begränsade på $[0,1]$, så $|f^{(n)}(t)| \le M, 0 \le t \le 1$:

$$\lim_{n\to\infty} \max_{0\le t\le 1} |e^t - p_n(t)| = \lim_{n\to\infty} \frac{e}{n!} = 0.$$

Övning

Man kan få snabbare konvergens med Chebyshevpunkterna $c_k = -\cos\left[\frac{(2k-1)\pi}{2n}\right]$, $k = 1, 2, ..., n$. Vi transformerar de:

$$\frac{\beta-\alpha}{2}\underbrace{[-1,1]}_{c_k}+\frac{\alpha+\beta}{2}=\underbrace{[\alpha,\beta]}_{t_k}$$

så de transformerade Chebyshevpunkterna blir

$$\frac{\beta-\alpha}{2}c_k+\frac{\alpha+\beta}{2}=t_k,$$

I övningen har vi att $[\alpha,\beta]=[0,1]$ eller $\alpha=0, \beta=1$ så att $1/2c_k+1/2=t_k$ och $c_k=2t_k-1$ då $t_k=\frac{c_k+1}{2}$. Vi redan vet att

$$\underbrace{p_n(t)}_{\text{beräknad}}-\underbrace{e^t}_{\text{exakt}}=\frac{e^t(\theta)}{n!}(t-t_1)(t-t_2)...(t-t_n)\leq\frac{e}{n!}(t-t_1)(t-t_2)...(t-t_n)\leq\frac{e}{n!}.$$

Vi ska visa att om vi väljer Chebyshevpunkterna så gäller att:

$$\max_{0\leq t\leq 1}|e^t-p_n(t)|\leq\frac{e}{n!2^{2n-1}}.$$

Övning

Vi redan vet att Chebyshevpunkterna minimerar $\prod_{k=1}^{n}|t-t_k|$ när $|t|\leq 1$ och maximala värdet på $|(t-t_1)(t-t_2)...(t-t_n)|$ är då $1/2^{n-1}$. Nu har vi intervallet $[0,1]$ så vi får transformera punkterna c_k. Vi vet att $c_k=2t_k-1$ då $t_k=\frac{c_k+1}{2}$ och

$$\max_{0\leq t\leq 1}\prod_{k=1}^{n}|t-t_k|=\max_{0\leq t\leq 1}\prod_{k=1}^{n}\left|t-\frac{c_k+1}{2}\right|=$$

$$=\max_{0\leq t\leq 1}\prod_{k=1}^{n}\left|\frac{\underbrace{2t-1}_{c}-c_k}{2}\right|=\frac{1}{2^n}\max_{-1\leq c\leq 1}\prod_{k=1}^{n}|c-c_k|=\frac{1}{2^{2n-1}}.$$

$$\lim_{n\to\infty}\max_{0\leq t\leq 1}|\underbrace{p_n(t)}_{\text{beräknad}}-\underbrace{e^t}_{\text{exakt}}|=\lim_{n\to\infty}\max_{0\leq t\leq 1}\frac{e}{n!2^{2n-1}}=0.$$

Splinefunktioner

Polynom av höga gradtal är svårhanterliga men har samtidigt lokalt goda approximationsegenskaper och är enkla att beskriva, lagra, beräkna, integrera, derivera, etc. En vanlig kompromiss är styckvisa polynom av låga gradtal. Man behåller polynomens enkelhet men slipper svängningarna.

Definition

En interpolerande splinefunktion av grad j är en funktion som interpolerar (t_k, y_k), $k = 1, 2, ..., n$ och som består av styckvisa polynom på intervallen $[t_1, t_2], [t_2, t_3], ...$ 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 delinterval).

Splinefunktioner

- 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 möjligt att åstadkomma och varför man inte kan kräva kontinuerlig tredjederivata.

Splinefunktioner

Exempel

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:

$$\underbrace{2(n-1)}_{\text{interp.krav}} + \underbrace{n-2}_{p_k \in C^1} + \underbrace{n-2}_{p_k \in C^2} = 4n - 6 \neq 4(n-1).$$

Splinefunktionern: kubisk spline

Här är några vanliga tilläggsvillkor (s är splinefunktionen):

- $s''(t_1) = s''(t_n) = 0$ s.k. 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 $[t_{n-2}, t_n]$).

Kubisk spline för 3 punkter t_1, t_2, t_3

Exempel

En kubisk spline för 3 punkter t_1, t_2, t_3 kan skrivas som:

$$p_1(t) = \alpha_1 + \alpha_2 t + \alpha_3 t^2 + \alpha_4 t^3, \ \ t \in [t_1, t_2] \tag{32}$$
$$p_2(t) = \beta_1 + \beta_2 t + \beta_3 t^2 + \beta_4 t^3, \ \ t \in [t_2, t_3]. \tag{33}$$

Koefficienterna $\alpha_i, \beta_i, i = 1, 2, 3, 4$ ska bestämmas.
Interpolationskravet ger 4 villkor (ty varje polynom måste interpolera 2 knutpunkter). Detta ger oss kontinuiteten gratis. Kontinuerlig förstaderivata $p_1'(t), p_2'(t)$ ger 1 villkor (inre punkt) och lika många för andraderivatan.
Så vi har: $4 + 2 = 6$ villkor.

$$\begin{aligned} p_1(t) &= \alpha_1 + \alpha_2 t + \alpha_3 t^2 + \alpha_4 t^3 \\ p_2(t) &= \beta_1 + \beta_2 t + \beta_3 t^2 + \beta_4 t^3 \end{aligned}$$

1)

$$\begin{aligned} p_1(t_1) &= y_1 = \alpha_1 + \alpha_2 t_1 + \alpha_3 t_1^2 + \alpha_4 t_1^3 \\ p_1(t_2) &= y_2 = \alpha_1 + \alpha_2 t_2 + \alpha_3 t_2^2 + \alpha_4 t_2^3 \end{aligned}$$

2)

$$\begin{aligned} p_2(t_2) &= y_2 = \beta_1 + \beta_2 t_2 + \beta_3 t_2^2 + \beta_4 t_2^3 \\ p_2(t_3) &= y_3 = \beta_1 + \beta_2 t_3 + \beta_3 t_3^2 + \beta_4 t_3^3 \end{aligned}$$

$p'_1(t_2) \in C \Longrightarrow p'_1(t_2) = p'_2(t_2)$

$$\begin{aligned} p'_1(t) &= \alpha_2 + 2\alpha_3 t + 3\alpha_4 t^2 \\ p'_2(t) &= \beta_2 + 2\beta_3 t + 3\beta_4 t^2 \end{aligned}$$

3) $p'_1(t_2) = p'_2(t_2)$:

$$p'_1(t_2) = \alpha_2 + 2\alpha_3 t_2 + 3\alpha_4 t_2^2 =$$
$$= \beta_2 + 2\beta_3 t_2 + 3\beta_4 t_2^2 = p'_2(t_2)$$

4) $p''_1(t_2) \in C \Longrightarrow p''_1(t_2) = p''_2(t_2)$

$$\begin{aligned} p''_2(t) &= 2\beta_3 + 6\beta_4 t \\ p''_1(t) &= 2\alpha_3 + 6\alpha_4 t \end{aligned}$$
$$p''_2(t_2) = 2\beta_3 + 6\beta_4 t_2 =$$
$$= 2\alpha_3 + 6\alpha_4 t_2 = p''_1(t_2)$$

Notera, att vi har skrivit $4 + 2 = 6$ villkor, behöver 2 till (vi har 8 koefficienter, som ska bestämmas). Vi väljer följande 2 tillägsvillkor: $p''_1(t_1) = 0; p''_2(t_3) = 0$:

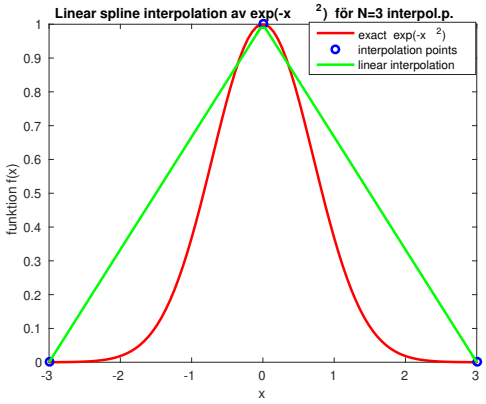
$$2\alpha_3 + 6\alpha_4 t_1 = 0,$$
$$2\beta_3 + 6\beta_4 t_3 = 0.$$

Nu samlar vi alla villkor för att hitta alla koefficienter $\alpha_i, \beta_i, i = 1, 2, 3, 4$ från lösning av följande system av linjära ekvationer:

$$\begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\ 1 & t_2 & t_2^2 & t_2^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\ 0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3 \\ 0 & 1 & 2t_2 & 3t_2^2 & 0 & -1 & -2t_2 & -3t_2^2 \\ 0 & 0 & 2 & 6t_2 & 0 & 0 & -2 & -6t_2 \\ 0 & 0 & 2 & 6t_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_3 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_2 \\ y_3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad (34)$$

Övning

Definera splinefunktion av grad 1 som interpolerar $(t_1, y_1), (t_2, y_2), (t_3, y_3)$ och som består av styckvisa polynom av grad 1 på intervallen $[t_1, t_2], [t_2, t_3]$.



Övning

Definera splinefunktion av grad 1 som interpolerar $(t_1, y_1), (t_2, y_2), (t_3, y_3)$ och som består av styckvisa polynom av grad 1 på intervallen $[t_1, t_2], [t_2, t_3]$.

Svar

Splinefunktion av grad 1 skrivs som:

$$p_1(t) = \alpha_1 + \alpha_2 t, \;\; t \in [t_1, t_2],$$
$$p_2(t) = \beta_1 + \beta_2 t, \;\; t \in [t_2, t_3].$$

Koefficienterna $\alpha_i, \beta_i, i = 1, 2$ ska bestämmas. I exemplet har vi ingen kontinuerlig derivata utan bara kontinuitet hos splinefunktionen. Interpolationskravet ger 4 villkor (ty varje polynom måste interpolera 2 knutpunkter).

Detta ger oss kontinuiteten och möjlighet för beräkning av alla koefficienter $\alpha_i, \beta_i, i = 1, 2$:

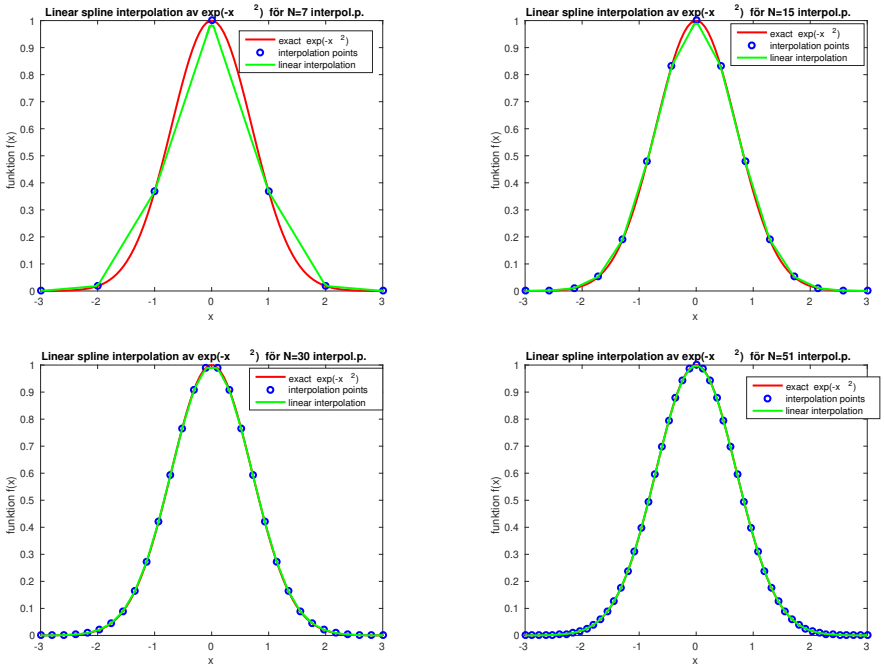
$$p_1(t_1) = \alpha_1 + \alpha_2 t_1 = y_1,$$
$$p_1(t_2) = \alpha_1 + \alpha_2 t_2 = y_2,$$
$$p_2(t_1) = \beta_1 + \beta_2 t_2 = y_2,$$
$$p_2(t_3) = \beta_1 + \beta_2 t_3 = y_3.$$

$$\begin{bmatrix} 1 & t_1 & 0 & 0 \\ 1 & t_2 & 0 & 0 \\ 0 & 0 & 1 & t_2 \\ 0 & 0 & 1 & t_3 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{35}$$

Vi kan hitta $\alpha_i, i = 1, 2$ från första två ekvationer is systemet (35), och $\beta_i, i = 1, 2$ - från sista två ekvationer:

$$\alpha_2 = \frac{y_2 - y_1}{t_2 - t_1}, \quad \alpha_1 = y_1 - \frac{y_2 - y_1}{t_2 - t_1} \cdot t_1,$$
$$\beta_1 = \frac{y_3 - y_2}{t_3 - t_2}, \quad \beta_1 = y_2 - \frac{y_3 - y_2}{t_3 - t_2} \cdot t_2.$$

Linear spline interpolation för $f(x) = e^{-x^2}$



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 eller).
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 $\int_0^5 e^{-0.1\cdot x^2}\sin(5x)dx$.
Facit: $\int_0^5 e^{-0.1\cdot x^2}\sin(5x)dx \approx 0.1863$.

I Matlab finns den äldre adaptiva metoden `quadl` och den nyare `integral`.

```
fun = @(x) exp(-0.1*x.^2).*sin(5*x)
Q1 = quadl(fun,0.0, 5.0)
Q2 = integral(fun,0.0,5.0)
```


Kvadratur - numerisk integration

Metoderna bygger på två principer:

1. indelning av $[a, b]$ i intervall (inte alltid lika långa)
2. approximation av f med polynom på varje delintervall följd av integration av polynom

Man använder ofta adaptiva metoder som försöker anpassa längden på delintervallen så att det sammanlagda felet blir mindre än en given tolerans. Det är då vanligt att man har olika långa intervall och det är inte ovanligt att man har olika gradtal på polynomen. Som vi redan vet, i Matlab finns den äldre adaptiva metoden `quadl` och den nyare `integral`.

Newton-Cotes quadrature

Att integrera interpolationspolynom ger Newton-Cotes metoder. Man skiljer mellan öppna Newton-Cotes metoder där ändpunkterna är med:

$$x_i = a + \frac{i(b-a)}{n+1}, i = 1, ..., n.$$

resp. slutna Newton-Cotes metoder där ändpunkterna ej tas med:

$$x_i = a + \frac{(i-1)(b-a)}{n-1}, i = 1, ..., n.$$

Vi ska studera:

- Rektangelmetoden eller mittpunktsmetoden: enklaste metoden är mittpunktsmetoden (rektangelmetoden) där vi approximerar $f(x)$ med $f((x_k + x_{k+1})/2)$ i intervallet $[x_k, x_{k+1}]$.
- Trapetsmetoden: approximation av f med ett linjärt interpolationspolynom
- Simpson's metod: approximation av f med ett kvadratisk interpolationspolynom

Kvadratur (Trapetsmetoden)

Trapetsmetoden: approximation av f med ett linjärt interpolationspolynom (se förel. 13, linjärt interpolation i 2 punkter):

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

på varje delintervall (beräkna integralet):

$$\int_a^b f(x)dx \approx \int_a^b \left(f(a) + (x-a)\frac{f(b)-f(a)}{b-a} \right) dx = \frac{b-a}{2}(f(a)+f(b)).$$

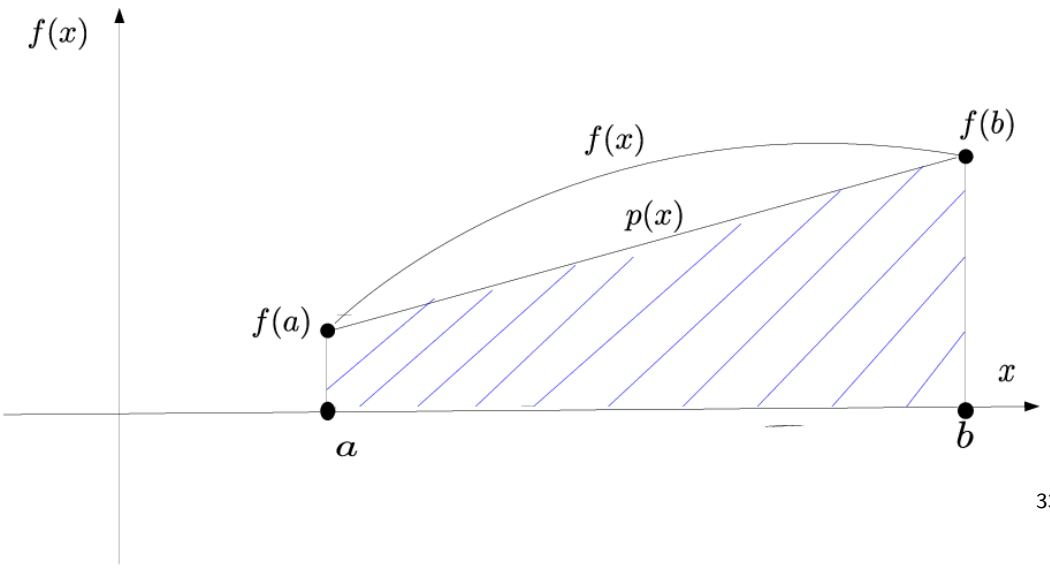
På intervallet $[a, b]$ approximerar vi integralen med arean av en parallelltrapets (därav namnet):

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a)+f(b)), \; h = b-a$$

Kvadratur (Trapetsmetoden)

På intervallet $[a, b]$ approximerar vi integralen med arean av en parallelltrapets (därav namnet):

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a) + f(b)),\ h = b - a$$



Kvadratur: komposit trapetsmetoden

Vi delar nu in $[a, b]$ i $n - 1$ lika långa delintervall (en del författare börjar med x_0):

$$x_k = a + (k - 1)h,\ k = 1, ..., n,\ h = (b - a)/(n - 1).$$

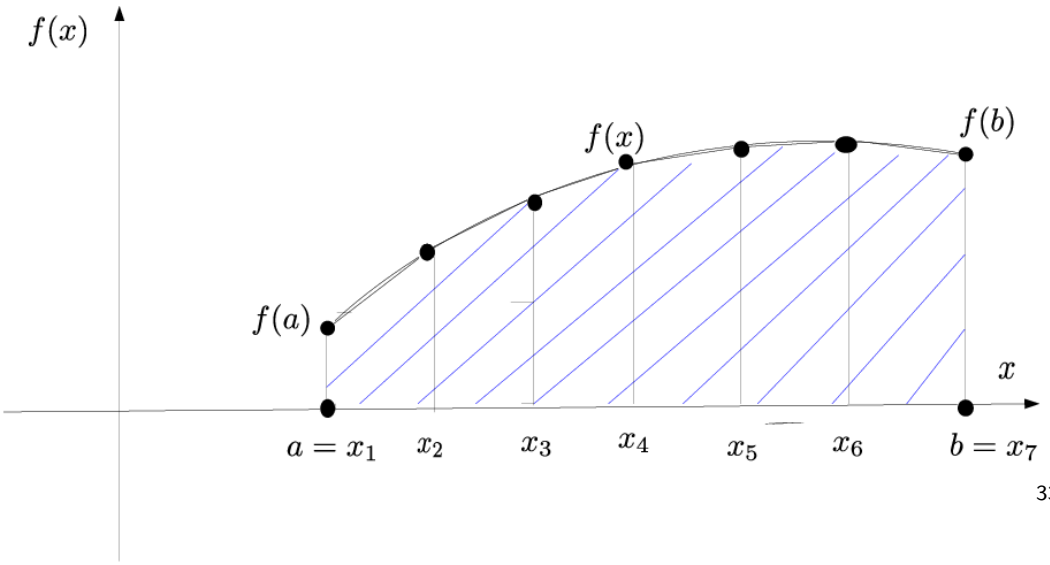
så att $x_1 = a$ och $x_n = b$.
Beteckna den approximation vi får med $T_n(f)$. Den blir:

$$\frac{h}{2}[f(x_1) + f(x_2)) + (f(x_2) + f(x_3)) + ... + (f(x_{n-1}) + f(x_n))] =$$

$$h\left[\frac{f(x_1)}{2} + f(x_2) + f(x_3) + ... + f(x_{n-1}) + \frac{f(x_n)}{2}\right]$$

Kvadratur: komposit trapetsmetoden

$$\frac{h}{2}[f(x_1)+f(x_2))+(f(x_2)+f(x_3))+...+(f(x_{n-1})+f(x_n))] =$$
$$h\left[\frac{f(x_1)}{2}+f(x_2)+f(x_3)+...+f(x_{n-1})+\frac{f(x_n)}{2}\right]$$



Övning

Använd trapetsmetoden för att beräkna $\int_0^1 x^2 dx$.
Trapetsmetoden:

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a)+f(b)),\ h=b-a$$

Trapetsmetoden för $\int_0^1 f(x)dx$ är:

$$\int_0^1 f(x)dx \approx \frac{1}{2}(f(1)+f(0))\cdot(1-0).$$

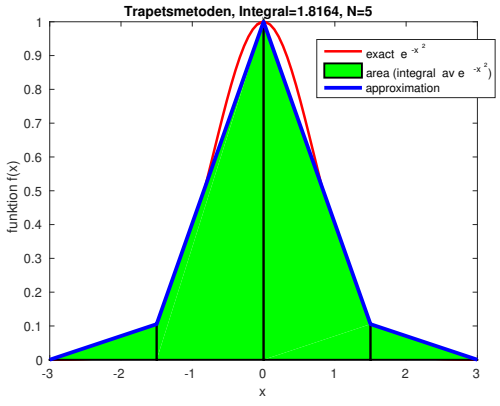
I vårt fall vi har $f(x)=x^2$, då trapetsmetoden för $\int_0^1 x^2 dx$ ger oss:

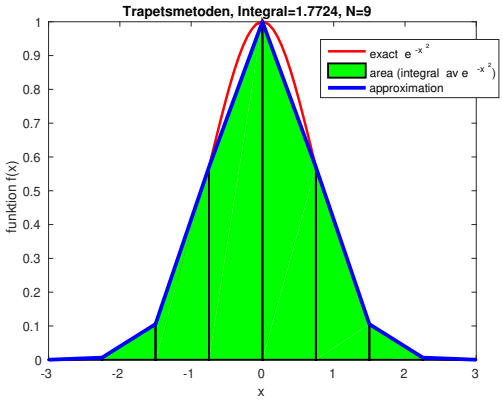
$$\int_0^1 x^2 dx \approx \frac{1}{2}(1^2+0^2)=\frac{1}{2}.$$

Trapetsmetoden i Matlab för $f(x) = e^{-x^2}$

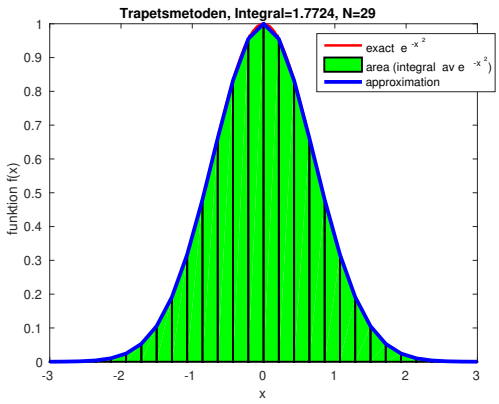
```
fun = @(x) exp(-x.^2)

Q = integral(fun,-3.0,3.0);

N_calc = 11;
x_calc = linspace(-3.0, 3.0, N_calc);

for i = 1:N_calc
fun_calc(i) =fun(x_calc(i));
end
int_calc = trapz(x_calc, fun_calc);
```

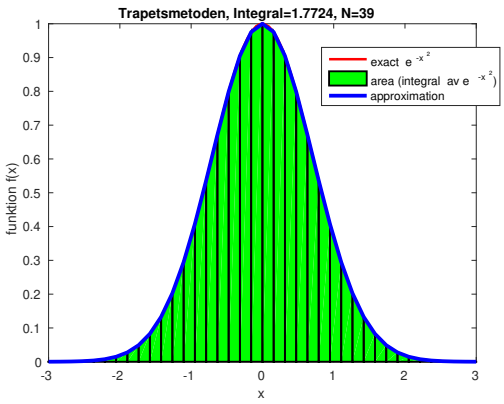

Trapetsmetoden för $f(x) = e^{-x^2}$





Trapetsmetoden för $f(x) = e^{-x^2}$





Kvadratur (Trapetsmetoden)

Om man kör Trapetsmetoden på vårt första exempel för beräkning av $\int_0^5 e^{-0.1 \cdot x^2} \sin(5x) dx$
med $n = 11, 21, 41, 81$ verkar felet ha utseendet ch^2 när $h \to 0, c = const.$. Kan man bevisa att felet har utseendet ch^2?

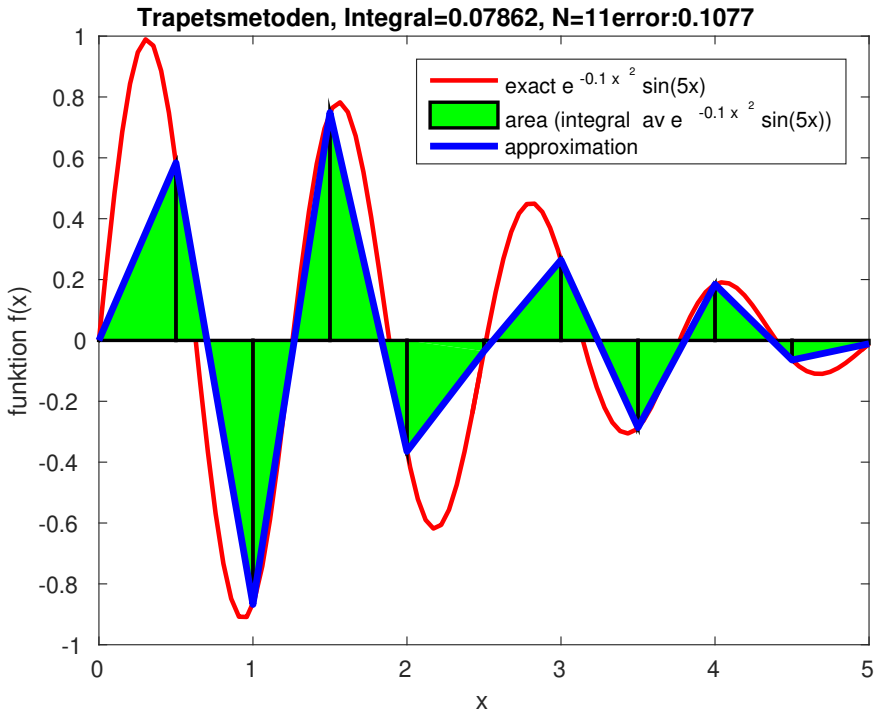
l	N	Fel e_l	e_l/e_{l+1}	q	h	h^2
1	11	0.1077			0.5	0.25
2	21	0.0245	4.3959	2.1362	0.25	0.0625
3	41	0.0060	4.0833	2.0297	0.125	0.0156
4	81	0.0015	4.0000	2	0.0625	0.0039

Ordningsfelet q:

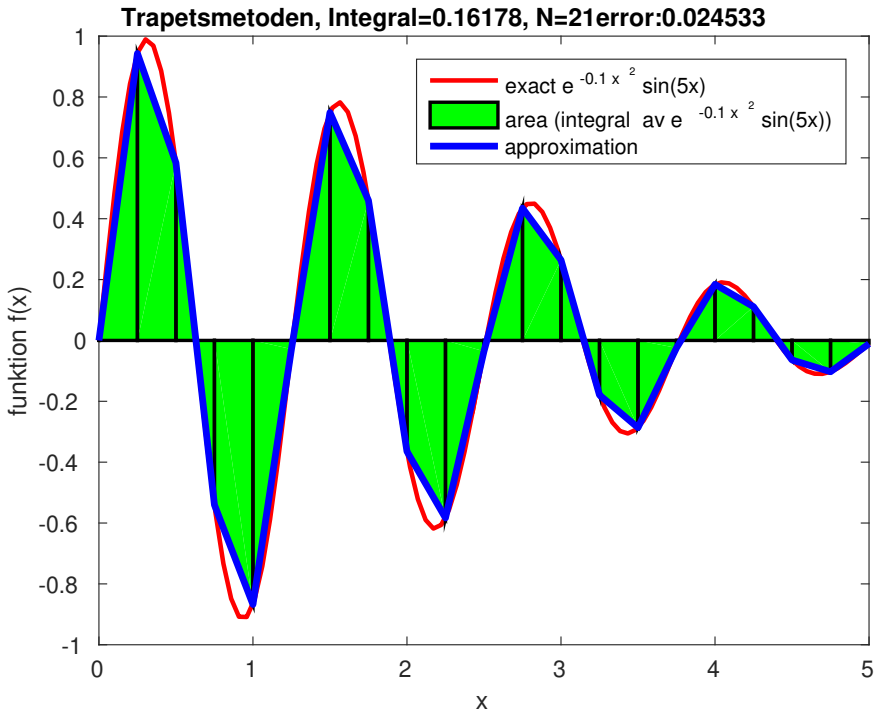
$$q = \frac{\log\left(\frac{e_{l+1}}{e_l}\right)}{\log(0.5)} \approx \frac{log(0.5^k)}{\log(0.5)} = \frac{klog(0.5)}{\log(0.5)} \approx k,$$

eftersom $\log\left(\frac{e_{l+1}}{e_l}\right) = \log\left(\frac{e_h}{e_{2h}}\right) = \log\left(\frac{Ch^k}{C(2h)^k}\right) \approx \log(0.5^k)$ för fel $e_h \approx Ch^k, e_{2h} \approx C(2h)^k$.

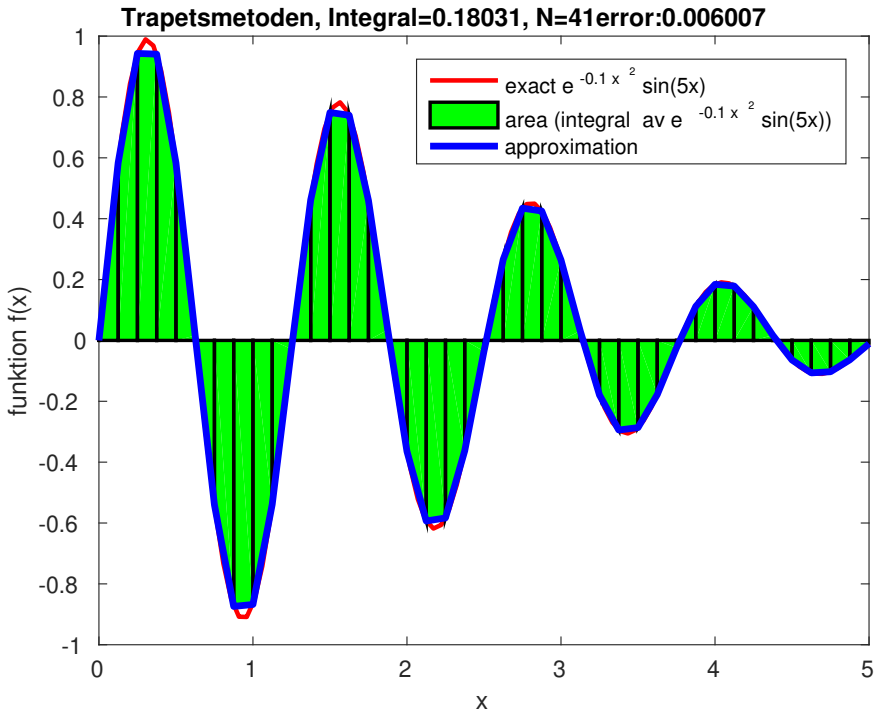
Trapetsmetoden för $f(x) = e^{-0.1 \cdot x^2} \sin(5x)$



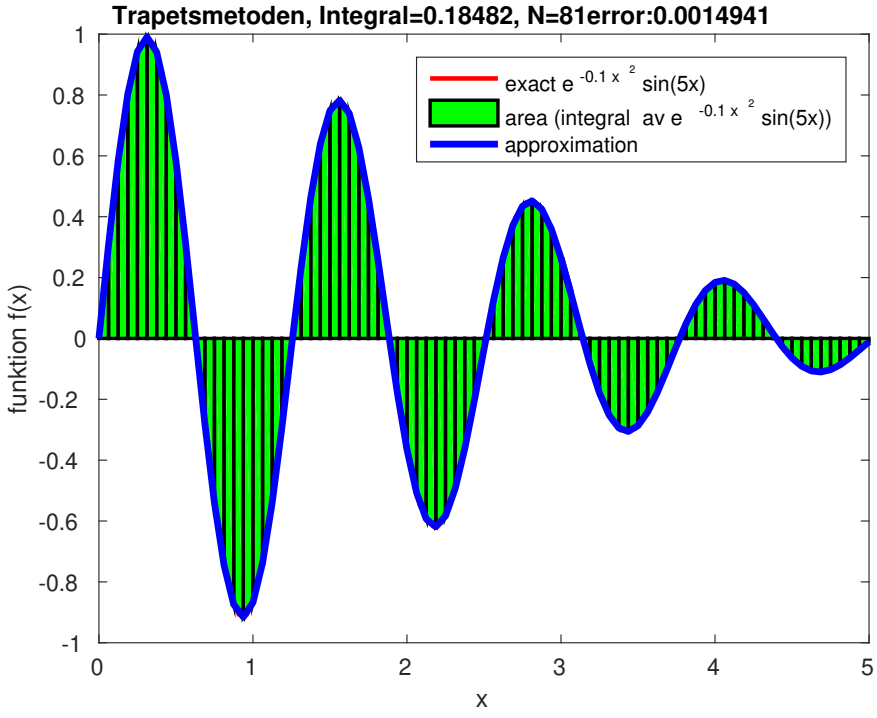
Trapetsmetoden för $f(x) = e^{-0.1 \cdot x^2} \sin(5x)$



Trapetsmetoden för $f(x) = e^{-0.1 \cdot x^2} \sin(5x)$



Trapetsmetoden för $f(x) = e^{-0.1 \cdot x^2} \sin(5x)$



Kvadratur (Trapetsmetoden)

Från interpolationsteorin vet vi att:

$$f(x) - p(x) = \frac{f''(\xi)}{2}(x-a)(x-b),\ \xi \in (a,b)$$

med ett intervall. Alltså

$$\int_a^b f(x)dx - \int_a^b p(x)dx = \int_a^b \frac{f''(\xi)}{2}(x-a)(x-b)dx =$$
$$\frac{f''(\xi)}{2}\int_a^b (x-a)(x-b)dx = -\frac{(b-a)^3 f''(\xi)}{12},\ \xi \in (a,b).$$

Detta följer av integralkalkylens medelvärdessats ($(x-a)(x-b)$ byter inte tecken på $[a,b]$).

Fel

I det allmänna fallet, med $n-1$ delintervall 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 alla mellanliggande värden):

$$\frac{1}{n-1}\sum_{k=1}^{n-1} f''(\xi_k) = f''(\xi) \quad (\text{nytt } \xi)$$

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 $h(n-1) = b-a$.

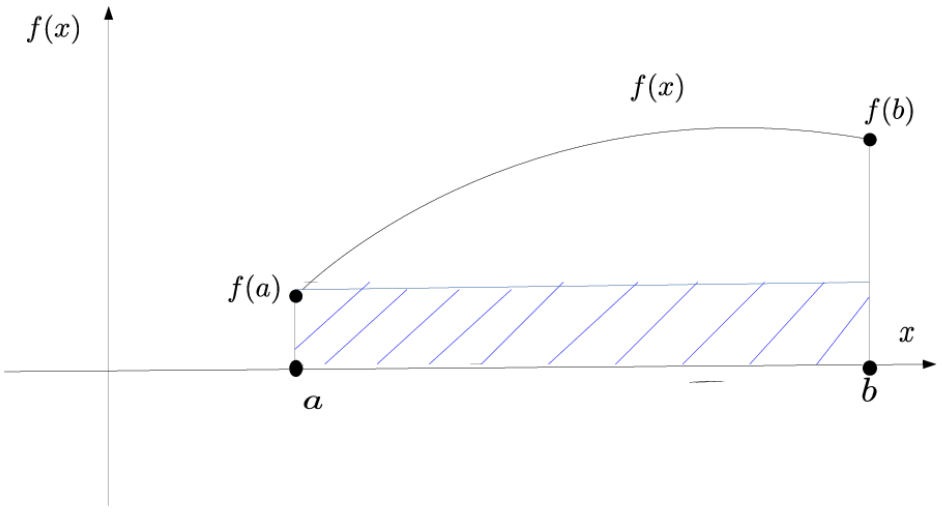
Kvadratur (Trapetsmetoden)

- 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 till är ju $(x_k, f(x_k)),\ k = 1, ..., n$ men det finns oändligt många funktioner som interpolerar dessa punkter (med olika värden på integralen). Detta är ett allmänt beräkningsproblem (ändliga punktmändger från oändliga punktmängder).

Rektangelmetoden

Enklaste metoden är rektangelmetoden där vi approximerar $f(x)$ med $f(x_k)$ eller med $f(x_{k+1})$ i intervallet $[x_k, x_{k+1}]$:

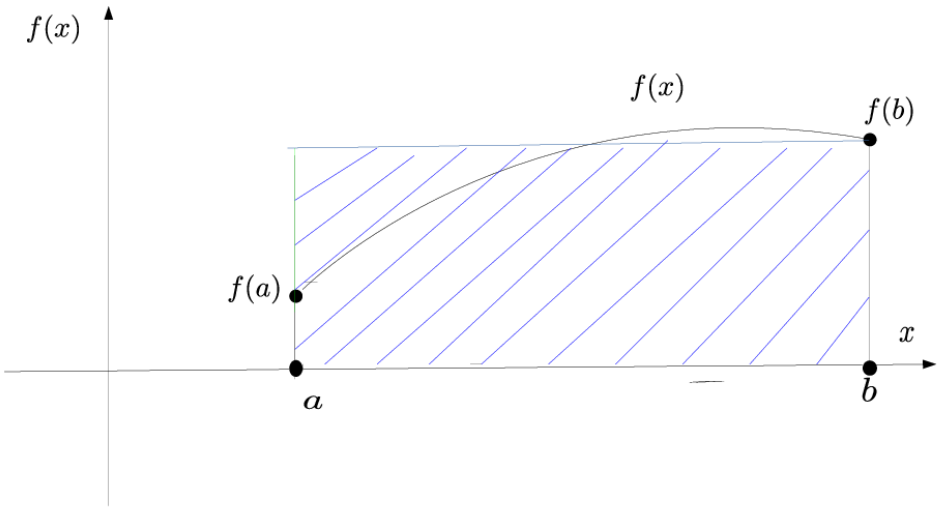
$$\int_a^b f(x)dx \approx \int_a^b f(a)dx = (b-a)f(a).$$



Rektangelmetoden

Enklaste metoden är rektangelmetoden där vi approximerar $f(x)$ med $f(x_k)$ eller med $f(x_{k+1})$ i intervallet $[x_k, x_{k+1}]$:

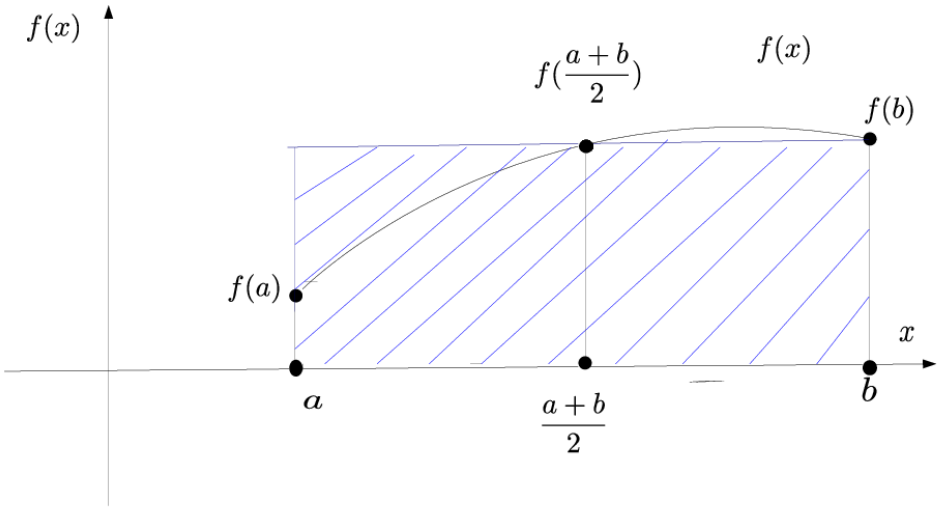
$$\int_a^b f(x)dx \approx \int_a^b f(b)dx = (b-a)f(b).$$



Rektangelmetoden: mittpunktsmetoden

Mittpunktsmetoden är också rektangelmetoden där vi approximerar $f(x)$ med $f((x_k + x_{k+1})/2)$ i intervallet $[x_k, x_{k+1}]$:

$$\int_a^b f(x)dx \approx \int_a^b f((a+b)/2)dx = (b-a)f\left(\frac{a+b}{2}\right).$$



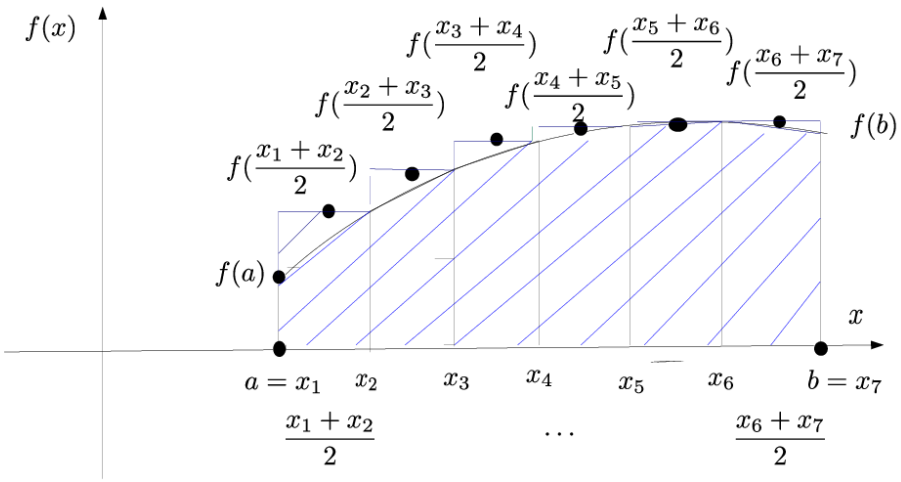
Komposit mittpunktsmetoden

Vi delar nu in $[a, b]$ i $n-1$ lika långa delintervall:

$$x_k = a + (k-1)h,\ k = 1, ..., n,\ h = (b-a)/(n-1).$$

så att $x_1 = a$ och $x_n = b$.

$$\int_a^b f(x)dx \approx h[f((x_1+x_2)/2) + f((x_2+x_3)/2) + ... + f((x_{n-1}+x_n)/2)]$$



Fel i rektangelmetoden för $\hat{x} = x_{i-1}, i = 1, ..., n$

Integralkalkylens medelvärdessats:

$$f'(x) \in C[a, b] \to f(x) = f(\hat{x}) + f'(\xi)(x - \hat{x}),\ \xi \in [a, b].$$

Vi ska använda den för att räkna fel för $\hat{x} = x_{i-1}, i = 1, ..., n$:
$\left|\int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i\right|$ för $\hat{x} = x_{i-1}$:

$$\int_{x_{i-1}}^{x_i} f(x)\ dx \approx f(\hat{x}_i)h_i,$$

$$\left|\int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i\right| = \left|\int_{x_{i-1}}^{x_i} [f(\hat{x}_i) + f'(\xi_i)(x - \hat{x}_i)]\ dx - f(\hat{x}_i)h_i\right|$$

$$\leq \max_{\hat{\xi}_i \in I_i} |f'(\xi_i)| \int_{x_{i-1}}^{x_i} (x - \hat{x}_i)\ dx = \max_{\xi_i \in I_i} |f'(\xi_i)| \frac{h_i^2}{2},\quad \xi_i \in [x_{i-1}, x_i].$$

Fel i rektangelmetoden för $\hat{x} = x_{i-1}, i = 1, ..., n$

Summerar fel på alla intervaler $I_i, i = 1, ..., n$ för att få felen i allmänna fallet:

$$\sum_{i=1}^{n-1} \int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i \leq 1/2 \sum_{i=1}^{n-1} \max_{\xi_i \in I_i} |f'(\xi_i)| h_i^2$$
$$= 1/2 \sum_{i=1}^{n-1} \max_{\xi_i \in I_i} |f'(\xi_i)| h_i h_i$$
$$= \frac{(b-a)|f'(\xi)|h}{2}, \quad \xi \in [a, b].$$

var $b - a = h(n-1)$ för $h = h_i, i = 1, ..., n-1$. Vi har använt att vi antar att f' är kontinuerlig: vi antar f' min/max på $[a, b]$ så att

$$\min_{a \leq x \leq b} f'(x) \leq \frac{1}{n-1} \sum_{i=1}^{n-1} f'(\xi_i) \leq \max_{a \leq x \leq b} f'(x)$$

så att (en kontinuerlig funktion antar alla mellanliggande värden):

$$\frac{1}{n-1} \sum_{i=1}^{n-1} f'(\xi_i) = f'(\xi) \quad \xi \in [a, b], \quad \xi_i \in [x_{i-1}, x_i].$$

Fel i mittpunktsmetoden

Taylor's formel:

$$f''(x) \in C[a, b] \rightarrow f(x) = f(\hat{x}_i) + f'(\hat{x}_i)(x - \hat{x}_i) + f''(\xi_i)(x - \hat{x}_i)^2/2,$$
$$\hat{x}_i \in [x_{i-1}, x_i] : \hat{x}_i = \frac{x_i + x_{i-1}}{2}, \quad \xi_i \in [x_{i-1}, x_i].$$

Vi ska använda den för att räkna fel $\left| \int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i \right|$:

$$\int_{x_{i-1}}^{x_i} f(x)\ dx \approx f(\hat{x}_i)h_i; \quad \left| \int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i \right|$$
$$= \left| \int_{x_{i-1}}^{x_i} [f(\hat{x}_i) + f'(\hat{x}_i)(x - \hat{x}_i) + f''(\xi_i)(x - \hat{x}_i)^2/2]\ dx - f(\hat{x}_i)h_i \right| =$$
$$\left| \underbrace{\int_{x_{i-1}}^{x_i} f(\hat{x}_i)\ dx - f(\hat{x}_i)h_i}_{=0} + \int_{x_{i-1}}^{x_i} f''(\xi_i)(x - \hat{x}_i)^2/2\ dx + \underbrace{\int_{x_{i-1}}^{x_i} f'(\hat{x}_i)(x - \hat{x}_i)\ dx}_{=0} \right|.$$

Fel i mittpunktsmetoden

$$\left| \int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i \right| \leq \max_{\xi_i \in I_i} |f''(\xi_i)| \int_{x_{i-1}}^{x_i} (x - \hat{x}_i)^2/2\ dx$$
$$= 1/2 \max_{\xi_i \in I_i} |f''(\xi_i)| \int_{x_{i-1}}^{x_i} (x - \hat{x}_i)^2\ dx = 1/2 \max_{\xi_i \in I_i} |f''(\xi_i)| \frac{2h_i^3}{24}$$
$$= \max_{\hat{x}_i \in I_i} |f''(\xi_i)| \frac{h_i^3}{24}, \quad \xi_i \in [x_{i-1}, x_i], I_i = [x_{i-1}, x_i].$$

Fel i mittpunktsmetoden

Summerar fel för alla intervaler $I_i, i = 1, ..., n-1$ på intervalet $[a, b]$ för att få felen i allmänna fallet:

$$\sum_{i=1}^{n-1} \left| \int_{x_{i-1}}^{x_i} f(x)\ dx - f(\hat{x}_i)h_i \right|$$
$$\leq 1/24 \sum_{i=1}^{n-1} \max_{\xi_i \in I_i} |f''(\xi_i)| h_i h_i^2 = \frac{(b-a)|f''(\xi)|h^2}{24}, \quad \xi \in [a, b].$$

var $b - a = h(n-1)$ för $h = h_i, i = 1, ..., n-1$.
Vi har använt att vi antar att f'' är kontinuerlig: vi antar f'' min/max på $[a, b]$ så att

$$\min_{a \leq x \leq b} f''(x) \leq \frac{1}{n-1} \sum_{i=1}^{n-1} f''(\xi_i) \leq \max_{a \leq x \leq b} f''(x)$$

så att (en kontinuerlig funktion antar alla mellanliggande värden):

$$\frac{1}{n-1} \sum_{i=1}^{n-1} f''(\xi_i) = f''(\xi) \quad \xi \in [a, b].$$

Kvadratur (Newton-Cotes-kvadratur)

Felet för den sammansatta mittpunktsmetoden har utseendet:

$$(b-a)h^2 f''(\xi)/24$$

vilket lustigt nog är mindre än för trapetsmetoden.
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.

Example

En linjär approximation av t.ex. $f(x) = x$ över $[-1, 1]$ ger felet noll och en exakt integral. Approximation 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.

Example

Använd mittpunktsmetoden (rektangelmetoden) för att beräkna integralen $\int_{-1}^{1} x dx$.
Svar:
Rektangelmetoden för $\int_a^b f(x)dx$ är:

$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right).$$

I vårt fall vi har $f(x) = x$, då rektangelmetoden ger oss:

$$\int_{-1}^{1} x dx \approx (1-(-1)) f\left(\frac{-1+1}{2}\right) = 2 \cdot f(0) = 0.$$

Observera, att exakt integral är:

$$\int_{-1}^{1} x dx = \frac{x^2}{2}\Big|_{-1}^{1} = 0.$$

Övning

Använd mittpunktsmetoden (rektangelmetoden) för att beräkna integralen $\int_0^1 4x^3 dx$.
Svar:
Rektangelmetoden för $\int_a^b f(x)dx$ är:

$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right).$$

I vårt fall vi har $f(x) = 4x^3$, då rektangelmetoden för $\int_0^1 4x^3 dx$ ger oss:

$$\int_0^1 4x^3 dx \approx (1-0)f\left(\frac{1+0}{2}\right) = f(1/2) = 4 \cdot (1/2)^3 = 1/2.$$

Simpson's metod

Vi har tittat på trapetsmetoden där man använder en linjär approximation. Använder man en kvadratisk approximation - interpolationspolynomet på Lagranges form med $t_1 = a, t_2 = m = (a+b)/2, t_3 = b$:

$$P(t) = f(a)\frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} + f(m)\frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} + f(b)\frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)}$$

eller med $t = x$ och $t_1 = a, t_2 = m = (a+b)/2, t_3 = b$:

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

får man Simpsons formel:

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Simpson's metod: lokalt fel

Vi använder Taylor's formel för $\hat{x} = (a+b)/2, h = (b-a)/2, \xi \in [a,b]$ för att få Simpson's regel och estimera fel i den:

$$\int_a^b f(x)dx = \int_a^b \Big[f(\hat{x}) + f'(\hat{x})(x-\hat{x}) + \frac{f''(\hat{x})(x-\hat{x})^2}{2!} + \frac{f^{(3)}(\hat{x})(x-\hat{x})^3}{3!} + \frac{f^{(4)}(\xi)(x-\hat{x})^4}{4!} \Big]\, dx = \Big[f(\hat{x})\cdot x + f'(\hat{x})\frac{(x-\hat{x})^2}{2} + f''(\hat{x})\frac{(x-\hat{x})^3}{2!\cdot 3} + f^{(3)}(\hat{x})\frac{(x-\hat{x})^4}{3!\cdot 4} + f^{(4)}(\xi)\frac{(x-\hat{x})^5}{4!\cdot 5} \Big]\Big|_a^b = \Big[f(\hat{x})(b-a) + f'(\hat{x})\Big(\frac{(b-\hat{x})^2}{2} - \frac{(a-\hat{x})^2}{2}\Big) + f''(\hat{x})\Big(\frac{(b-\hat{x})^3}{2!\cdot 3} - \frac{(a-\hat{x})^3}{2!\cdot 3}\Big) + f^{(3)}(\hat{x})\Big(\frac{(b-\hat{x})^4}{3!\cdot 4} - \frac{(a-\hat{x})^4}{3!\cdot 4}\Big) + f^{(4)}(\xi)\Big(\frac{(b-\hat{x})^5}{4!\cdot 5} - \frac{(a-\hat{x})^5}{4!\cdot 5}\Big) = f(\hat{x})2h + \frac{f'(\hat{x})}{2}(h^2-h^2) + \frac{f''(\hat{x})}{6}\cdot(h^3+h^3) + \frac{f^{(3)}(\hat{x})}{24}(h^4-h^4) + \frac{f^{(4)}(\xi)}{120}[h^5+h^5] = f(\hat{x})2h + \frac{h^3}{3}\Big[\frac{f(a)-2f(\hat{x})+f(b)}{h^2} - \frac{h^2}{12}f^{(4)}(\xi)\Big] + \frac{h^5 f^{(4)}(\xi)}{60}$$

Simpson's metod: lokalt fel

Vi har använt följande approximation för $f''(\hat{x}) \approx \frac{f(a)-2f(\hat{x})+f(b)}{h^2}$ och fel i den (se övning 11, kapitel 7):

$$f''(\hat{x}) = \frac{f(a)-2f(\hat{x})+f(b)}{h^2} - \frac{h^2}{12}f^{(4)}(\xi),\ \xi \in [a,b].$$

Vi har fått för $\hat{x} = (a+b)/2, h = (b-a)/2, \xi \in [a,b]$:

$$\int_a^b f(x)dx = f(\hat{x})2h + \frac{h^3}{3}\Big[\underbrace{\frac{f(a)-2f(\hat{x})+f(b)}{h^2}}_{approximation} - \underbrace{\frac{h^2}{12}f^{(4)}(\xi)}_{fel,\ sekap.7,\ övn.11}\Big] + \frac{h^5 f^{(4)}(\xi)}{60}$$

$$= \frac{4h}{3}f(\hat{x}) + \frac{h}{3}(f(a)+f(b)) - \frac{h^5}{90}f^{(4)}(\xi)$$

$$= \underbrace{\frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]}_{Simpson's\ \ formel} - \frac{h^5}{90}f^{(4)}(\xi).$$

Kvadratur (Newton-Cotes-kvadratur)

Simpsons formel, som också har ett udda antar 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 $f^{(4)}$).

En allmän kvadraturmetod kan skrivas

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k)$$

där 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 $[x_5, x_7]$.

Kvadratur (Newton-Cotes-kvadratur)

$$\int_{x_1}^{x_3} f(x)dx + \int_{x_3}^{x_5} f(x)dx + \int_{x_5}^{x_7} f(x)dx \approx$$
$$\frac{x_3 - x_1}{6}\left[f(x_1) + 4f\left(\frac{x_1+x_3}{2}\right) + f(x_3)\right] +$$
$$\frac{x_5 - x_3}{6}\left[f(x_3) + 4f\left(\frac{x_3+x_5}{2}\right) + f(x_5)\right] +$$
$$\frac{x_7 - x_5}{6}\left[f(x_5) + 4f\left(\frac{x_5+x_7}{2}\right) + f(x_7)\right].$$

$\frac{x_1+x_3}{2} = x_2$ etc. och $h = x_{k+1} - x_k$ så 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. Med $f(x) = e^{-x^2}$ och ett absolut fel $\leq 1.2 \cdot 10^{-9}$ tar trapetsmetoden 7150 funktionsevalueringar, mittpunktsmetoden 5055 och Simpsons formel 52. Matlabs `quadl`, som är adaptiv, tar 18 (`integral` tycks alltid börja med 150, felet blir 10^{-16}).

Komposit Simpson's metod

Simpson's formel:

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \quad (36)$$

Komposit Simpon's formel för n punkter:

$$\int_a^b f(x)dx \approx \frac{h}{3}\sum_{j=1}^{n/2}[f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})]$$

$$= \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right] = S(x), \quad (37)$$

$$x_j = a + jh, j = 0, ..., n, h = \frac{b-a}{2(n-1)}.$$

För $n = 2$ från (37) få vi Simpson's formel (36) på $[a, b]$ för 2 punkter: a och b.

Komposit Simpson's metod

Fel i komposit Simpson's formel är för
$\xi_i \in [x_{2i}, x_{2i+2}]$, $i = 0, 1, 2, ..., n, h = (b-a)/2(n-1)$:

$$\int_a^b f(x)dx - S(x) = -\sum_{i=1}^{n/2}\frac{h^5}{90}f^{(4)}(\xi_i) = -\sum_{i=1}^{n/2}\frac{h^4}{90}f^{(4)}(\xi_i)\frac{b-a}{2\cdot(n-1)}$$

$$\leq -\frac{(b-a)h^4}{180}\max_{\xi\in[a,b]} f^{(4)}(\xi). \quad (38)$$

Vi har använt att vi antar att $f^{(4)}$ är kontinuerlig: vi antar $f^{(4)}$ min/max på $[a, b]$ så att

$$\min_{a\leq x\leq b} f^{(4)}(x) \leq \frac{1}{n-1}\sum_{k=1}^{n/2} f^{(4)}(\xi_k) \leq \max_{a\leq x\leq b} f^{(4)}(x)$$

så att (en kontinuerlig funktion antar alla mellanliggande värden):

$$\frac{1}{n-1}\sum_{i=1}^{n/2} f^{(4)}(\xi_i) = f^{(4)}(\xi) \quad \xi \in [a, b].$$

Övning

Använd Simpsons metod för att beräkna $\int_0^1 x^2 dx$.
Simpsons metod :

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Vi har: $a = 0, b = 1, f(x) = x^2, f(a) = a^2, f(0) = 0, f(b) = f(1) = 1^2 = 1, f\left(\frac{a+b}{2}\right) = f((0+1)/2) = f(1/2) = (1/2)^2 = 1/4.$

$$\int_0^1 x^2 dx \approx \frac{1-0}{6}[0 + 4 \cdot 1/4 + 1] = 1/3.$$

Kvadratur (Newton-Cotes-kvadratur)

Det spelar stor roll vilken metod man använder och h^m-faktorn är viktig. Låt oss anta att vi har en uppsättning metoder med feltermer(c konstant, m heltal och $h = 1/(n-1)$)

$$c(b-a)h^m f^{(m)}(\xi)$$

Om $f^{(m)}(\xi)$ är konstant kan felet skrivas Ch^m, C konstant. För att feltermen skall bli $\approx \tau$ en given tolerans, krävs alltså:

$$Ch^m \approx \tau, \quad n \approx \frac{1}{(\tau/C)^{1/m}}, \; n \propto \frac{1}{\tau^{1/m}}$$

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

Numerisk integration av data i Matlab

Vi vet hastighet $V(m/sec)$ av en bil i discreta tidspunkter $t_k(sec)$ i tiden $[0, T]$. Vi vill approximera distansen $S(m)$, som bilen har kört under tiden.
Distansen S räknas som $S = V \cdot t$ eller med hjälp av trapetsmetoden :

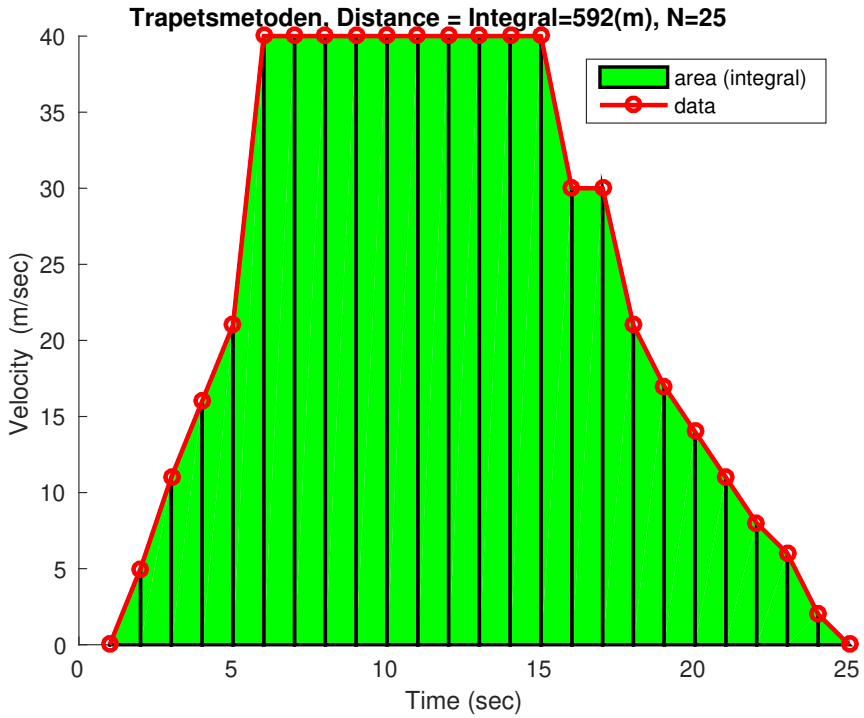
$$S = \int_0^T V dt \approx \sum_{k=1}^{N-1} \tau \frac{V(t_k) + V(t_{k+1})}{2}$$

och
$\tau = T/(N-1)$ för N diskr.punkter t_k.

Numerisk integration av data i Matlab

```
x_calc = 0:24;
N_calc = size(x_calc,2);
data = [0.0 5 11 16 21 40 ...
40 40 40 40 40 40 40 40 40 30 30 21 17 14 11 ...
8 6 2 0];
int_calc = trapz(x_calc, data);
figure
xvert = [x_calc(1:end-1);x_calc(1:end-1);x_calc(2:end); ...
x_calc(2:end)];
yvert = [zeros(1,N_calc-1);data(1:end-1);data(2:end);...
zeros(1,N_calc-1)];
patch(xvert, yvert,'g','LineWidth',1.5)
hold on
plot(x_calc, data, ' r- o', 'LineWidth',2)
```

Numerisk integration av data i Matlab



Kvadratur (Singulariteter)

Om någon av f:s lägre derivator har en singularitet i $[a, b]$ kan dock metoderna konvergera avsevärt långsammare.

Example

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 $x \in [0, h]$:

$$\int_0^h x^p dx - \frac{h[0^p + h^p]}{2} = \frac{x^{p+1}}{p+1}\Big|_0^h - \frac{h^{p+1}}{2} = \frac{(1-p)}{2(1+p)} h^{1+p}$$

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.

Kvadratur (Singulariteter)

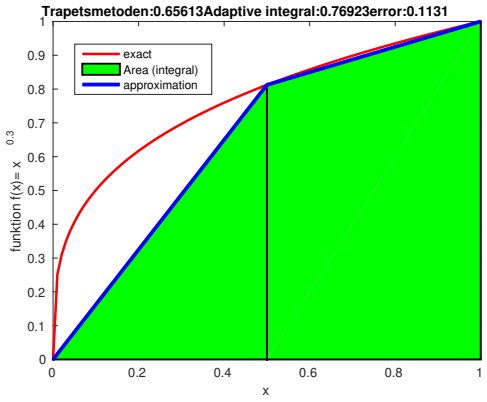
Tar vi $p = 0.3$ med samma tolerans som i föregående exempel, så kräver Simpson inte 52 funktionsberäkningar utan 1 697 157. Problemet är väsentligen av samma slag som när vi interpolerade $\sqrt{t}$ kring $t \geq 0$. Vad kan man göra? Man kan byta till en bättre metod, `integral` t.ex. som kräver 150 funktionsberäkningar. Kanske kan man 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).

I exemplet: vi gör variabelbyte $y = x^{0.3}$ ger $x = y^{1/0.3}$ som har en singularitet först i fjärdederivatan: $dx = 1/0.3 y^{1/0.3-1} dy$

$$\int_0^1 x^{0.3} dx = \int_0^1 \underbrace{y}_{x^{0.3}} \cdot \underbrace{\frac{1}{0.3} \frac{y^{1/0.3}}{y} dy}_{dx} \approx 0.7692$$

Simpson tar 83 funktionsberäkningar för y-integralen, `quadl` tar 48, `integral` tar 150, med väsentligen nollfel.

Numerisk integration av $\int_0^1 x^{0.3} dx$



Matlab's program Integrsingular_ex3.m som beräknar $\int_0^1 x^{0.3} dx$ (finns på kursens hemsida): Integral beräknad med adaptivt metod (Matlabs funktion **integral**: 0.7692, integral beräknad med Simpson's formel i 3 punkter: 0.7082, integral beräknad med trapetsmetoden i 3 punkter: 0.6561.

Kvadratur (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 det 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 metoden. Man kan också använda samma metoden 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 mittpunkter, $m = (a + b)/2$ och räknar ut en ny approximation, nu med tre punkter. Observera att detta kräver ett nytt funktionsvärde, $f(m)$.

Kvadratur (Adaptivitet)

Vi fortsätter nu så rekursivt på intervallen $[a, m]$ och $[m, b]$. När felet över ett intervall är tillräckligt litet 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).

$$I = T_h - h^3 f''(\xi)/12 \text{ resp. } I = T_{h/2} - h(h/2)^2 f''(\theta)/12$$

Antag att $c = -f''(\xi) \approx -f''(\theta)$ (behöver inte vara sant). Då gäller:

$$\begin{aligned} 0 \approx I - I &= T_h + h^3 c/12 - (T_{h/2} + h(h/2)^2 c/12) \\ &= T_h - T_{h/2} + ch^3(1 - 1/4)/12 = T_h - T_{h/2} + 3\underbrace{ch^3/(4 \cdot 12)}_{I - T_{h/2}} \\ &= T_h - T_{h/2} + 3(I - T_{h/2}). \end{aligned}$$

Men felet $I - T_{h/2}$ är ju $ch^3/(4 \cdot 12) = ch(h/2)^2/12$. Alltså

$$I \approx T_{h/2} + \frac{T_{h/2} - T_h}{3} = (4T_{h/2} - T_h)/3.$$

Kvadratur (Adaptivitet)

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:

$$I \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Bevis:
Notera för $h = (b-a)/2$

$$\begin{aligned} I &\approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \\ &= \frac{4h}{3} f\left(\frac{a+b}{2}\right) + \frac{h}{3}(f(a) + f(b)). \end{aligned} \tag{39}$$

Kvadratur (Adaptivitet)

Från andra sidan, från trapetsmetoden följer:

$$\begin{aligned} T_{h/2} &= \left(f(a) + f\left(\frac{a+b}{2}\right)\right)\frac{h}{2}\cdot\frac{1}{2} + \left(f\left(\frac{a+b}{2}\right) + f(b)\right)\frac{h}{2}\cdot\frac{1}{2} \\ &= \left(f(a) + f\left(\frac{a+b}{2}\right) + f(b)\right)\frac{h}{4}, \\ T_h &= \frac{f(a)+f(b)}{2}\cdot h, \\ (4T_{h/2} - T_h)/3 &= \frac{4h}{3} f\left(\frac{a+b}{2}\right) + \frac{h}{3}(f(a) + f(b)). \end{aligned} \tag{40}$$

Vi observerar, att (39) = (40).

Richardsonextrapolation

Ovanstående är ett specialfall av Richardsonextrapolation. Man kan visa att det existerar en serieutveckling av felet

$$\left(\int_a^b f(x)dx = \right) = I = T_h + a_1h^2 + a_2h^4 + a_3h^6 + ... \quad (41)$$

Vi halverar nu h och får

$$I = T_{h/2} + a_1h^2/4 + a_2h^4/16 + a_3h^6/64 + ...$$

Vi ser att

$$4I = 4T_{h/2} + a_1h^2 + a_2h^4/4 + a_3h^6/16 + ... \quad (42)$$

Vi beräknar nu differens (42) - (41) för att bli av med h^2-termen:

$$3I = 4I - I = 4T_{h/2} - T_h + \underbrace{(a_1h^2 - a_1h^2)}_{=0} + \underbrace{(a_2h^4/4 - a_2h^4)}_{\frac{-3a_2h^4}{4}} + ...$$

så att

$$I = \frac{4T_{h/2} - T_h}{3} - \frac{a_2h^4}{4} + ...$$

Rombergkvadratur

Detta kan man upprepa (med $T_{h/4}$) för att bli av med h^4-termen. Denna process (upprepad Richardsonextrapolation) kallas Rombergkvadratur. Rombergs metoden kan definieras induktivt för $h_n = \frac{1}{2^n}(b-a)$:

$$R(0,0) = h_1(f(a) + f(b)), \quad (43)$$

$$R(n,0) = \tfrac{1}{2}R(n-1,0) + h_n \sum_{k=1}^{2^{n-1}} f(a + (2k-1)h_n), \quad (44)$$

$$R(n,m) = R(n,m-1) + \tfrac{1}{4^m-1}(R(n,m-1) - R(n-1,m-1)), \quad (45)$$

eller

$$R(n,m) = \tfrac{1}{4^m-1}(4^mR(n,m-1) - R(n-1,m-1))$$

var $n \geq m$ och $m \geq 1$. Felet för $R(n,m)$ är (Mysovskikh 2002):

$$O\left(h_n^{2m+2}\right).$$

Noll extrapoleringen, $R(n,0)$, motsvarar trapezmetoden med $2^n + 1$ punkter. Den första extrapoleringen, $R(n,1)$, motsvarar Simpsons formel med $2^n + 1$ punkter.

Rombergkvadratur

Noll extrapoleringen, $R(n,0)$, motsvarar trapezmetoden med 2^n+1 punkter för $h_n = \frac{1}{2^n}(b-a)$:

$$\begin{aligned} R(n,0) &= \tfrac{1}{2}R(n-1,0) + h_n \sum_{k=1}^{2^{n-1}} f(a+(2k-1)h_n) \\ &= \frac{h}{2}[f(x_1)+f(x_2)) + (f(x_2)+f(x_3)) + ... + (f(x_{k-1})+f(x_k))] \\ &= h\left[\frac{f(x_1)}{2} + f(x_2) + f(x_3) + ... + f(x_{k-1}) + \frac{f(x_k)}{2}\right] \end{aligned}$$

Den första extrapoleringen, $R(n,1)$, motsvarar Simpsons formel med 2^n+1 punkter:

$$\begin{aligned} R(n,1) &= R(n,0) + \tfrac{1}{3}(R(n,0) - R(n-1,0)) \\ &= \frac{4}{3}R(n,0) - \frac{1}{3}R(n-1,0) = (4T_{h/2} - T_h)/3. \end{aligned}$$

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 $x_1 = a$, $x_2 = (a+b)/2$ och $x_3 = b$ 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 $0, 1, ..., m$ 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.

Gausskvadratur

Gausskvadratur är konstruerad för att ge ett exakt resultat för polynomier av grad $2n-1$ eller mindre med ett lämpligt val av punkterna x_i och vikterna w_i, $i = 1, ..., n$:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

Gausskvadratur ger bra resultat om funktionen $f(x)$ är väl approximerad av en polynomfunktion inom intervallet $[-1, 1]$. Metoden är exempelvis inte lämplig för funktioner med singulariteter.

Kvadratur (Gausskvadratur)

Tag intervallet $[-1, 1]$. Ska välja x_1, x_2, x_3 och vikter w_1, w_2, w_3 s.a.

$$\int_{-1}^{1} x^k dx = w_1 x_1^k + w_2 x_2^k + w_3 x_3^k, \ k = 0, 1, ..., m$$

för maximalt m. Integralens värde blir 0 om k är uddda och $2/(k+1)$ annars. Vi få lösa det ickelinjära ekvationssystemet:

$$\begin{aligned}
2 &= w_1 + w_2 + w_3 && k = 0 \\
0 &= w_1 x_1 + w_2 x_2 + w_3 x_3 && k = 1 \\
2/3 &= w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 && k = 2 \\
0 &= w_1 x_1^3 + w_2 x_2^3 + w_3 x_3^3 && k = 3 \\
2/5 &= w_1 x_1^4 + w_2 x_2^4 + w_3 x_3^4 && k = 4 \\
0 &= w_1 x_1^5 + w_2 x_2^5 + w_3 x_3^5 && k = 5
\end{aligned}$$

Det verkar inte rimligt att ta med en ekvation till. Vi har ju $3 + 3 = 6$ obekanta och sex ekvationer. För att lösa systemet kan man använda "brute force", men vi utnyttjar symmetri och antar att $x_1 < x_2 < x_3$ med $x_2 = 0$ och $x_1 = -x_3$.

Kvadratur (Gausskvadratur)

Detta 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:

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9} f\left(-\sqrt{3/5}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{3/5}\right)$$

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).

Gausskvadratur:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

för Gausspunkter x_i och Gaussvikter $\omega_i, i = 1, ..., n$.

n	Gausspunkter, x_i	Vikter, ω_i	Gausskvadratur	pol. gradtal $2n-1$
1	0	2	$w_1 f(x_1)$	$f(x) \approx p(x) = x^1 = x$
2	$\pm\sqrt{1/3}$	1	$\sum_{i=1}^{2} w_i f(x_i)$	$f(x) \approx p(x) = x^3$
3	0 $\pm\sqrt{3/5}$	$8/9$ $5/9$	$\sum_{i=1}^{3} w_i f(x_i)$	5, $x^k, k = 5$ $f(x) \approx p(x) = x^5$
4	$\pm\sqrt{3/7 - 2/7\sqrt{6/5}}$ $\pm\sqrt{3/7 + 2/7\sqrt{6/5}}$	$\frac{18+\sqrt{30}}{36}$ $\frac{18-\sqrt{30}}{36}$	$\sum_{i=1}^{4} w_i f(x_i)$	$f(x) \approx p(x) = x^7$
5	0 $\pm\frac{1}{3}\sqrt{5 - 2\sqrt{\frac{10}{7}}}$ $\pm\frac{1}{3}\sqrt{5 + 2\sqrt{\frac{10}{7}}}$	$128/225$ $\frac{322+13\sqrt{70}}{900}$ $\frac{322-13\sqrt{70}}{900}$	$\sum_{i=1}^{5} w_i f(x_i)$	9 $f(x) \approx p(x) = x^9$

Övning

Vi har följande kvadraturformel:

$$\int_0^1 f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

där vi vet att vi approximerar $f(x)$ med polynom $p(x)$ som har polynomiella gradtalet minst ett. Visa att $\sum_{i=1}^{n} w_i = 1$.
Svar:
Metoden är exakt för polynom av åtminstone grad noll (konstant polynom $p(x) = 1$). Då gäller:

$$1 = \int_0^1 1dx = \sum_{i=1}^{n} w_i p(x_i) = \sum_{i=1}^{n} w_i.$$

Övning

Välj w_1, w_2, x_1, x_2, i kvadraturformeln nedan, så att den får så högt polynomiellt gradtal m som möjligt. Vad blir detta gradtal ?

$$\int_0^1 x^k dx = w_1 x_1^k + w_2 x_2^k, \quad k = 0, 1, ..., m.$$

Övning

Välj w_1, w_2, x_1, x_2, i kvadraturformeln nedan, så att den får så högt polynomiellt gradtal m som möjligt. Vad blir detta gradtal ?

$$\int_0^1 x^k dx = w_1 x_1^k + w_2 x_2^k, \quad k = 0, 1, ..., m.$$

Svar: Formeln skall vara exakt för polynom $x^k, k = 0, 1, ..., m$ för maximalt m. Vi beräknar först

$$\int_0^1 x^k dx = \frac{x^{k+1}}{k+1}\Big|_0^1 = 1/(k+1).$$

Ekvationerna blir:

$$\begin{aligned} 1 &= w_1 + w_2, k = 0, \\ 1/2 &= w_1 x_1 + w_2 x_2, k = 1, \\ 1/3 &= w_1 x_1^2 + w_2 x_2^2, k = 2, \\ 1/4 &= w_1 x_1^3 + w_2 x_2^3, k = 3, \\ 1/5 &= w_1 x_1^4 + w_2 x_2^4, k = 4. \end{aligned}$$

Övning

$$\begin{aligned} 1 &= w_1 + w_2, k = 0, \\ 1/2 &= w_1 x_1 + w_2 x_2, k = 1, \\ 1/3 &= w_1 x_1^2 + w_2 x_2^2, k = 2, \\ 1/4 &= w_1 x_1^3 + w_2 x_2^3, k = 3, \\ 1/5 &= w_1 x_1^4 + w_2 x_2^4, k = 4. \end{aligned}$$

Första ekvationen ger $w_{1,2} = 1/2$. Lös ut för $k = 1, 2$ ekvationen $2x_2^2 - 2x_2 + \frac{1}{3} = 0$ (vi noterar, att $x_1 < x_2$) för att få $x_1 = \frac{1-1/\sqrt{3}}{2}, x_2 = \frac{1+1/\sqrt{3}}{2}$. Vi kollar nu fall $k = 3$. Utnyttjar vi binomialsatsen ser vi att $(1+c)^3 + (1-c)^3 = 2(1+3c^2)$ så att $w_1 x_1^3 + w_2 x_2^3 = (1/2^4) \cdot 2(1 + 3/3) = 1/4$, vilket är lika med det exakta värdet. Stämmer det för k = 4? Inte. Så, det polynomiella gradtalet är 3.

Kvadratur (Gausskvadratur)

Det allvarligaste problemet är dock att man inte kan återanvända funktionsvärdena när man gör adaptiva metoder. Det finns dock varianter, Gauss-Kronrodkvadratur, där man har en kompromiss mellan optimaliteten i Gausskvadratur och kraver på återanvändning av funktionsvärden, se boken. Finns `quadgk` i Matlab.

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:

$$\int_a^b f(z)dz = \int_{-1}^{1} f(\alpha x + \beta)\alpha\, dx \approx \sum_{k=1}^{3} (\alpha w_k) f(\alpha x_k + \beta)$$

Kvadratur (Gausskvadratur)

Om vi ska approximera integral

$$I(g) = \int_a^b g(t)dt,$$

och t ligger i ett annat intervall, $[a, b]$, och x ligger på $[-1, 1]$, får vi göra en linjär avbilding till detta intervall:

$$\frac{b-a}{2}[-1, 1] + \frac{a+b}{2} = [a, b],$$

eller

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

och integral $I(g) = \int_a^b g(t)dt$ kan beräknas som

$$I(g) = \int_a^b g(t)dt = \frac{b-a}{2} \int_{-1}^{1} g\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)\, dx \qquad (46)$$

$$\approx \frac{b-a}{2} \sum_{i=1}^{n} \omega_i g\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right). \qquad (47)$$

Kvadratur (Gausskvadratur)

Example

Vi vill beräkna

$$\int_0^3 f(x)dx = \int_0^3 e^{-x^2}dx$$

med hjälp av Gausskvadratur med 3 vikter.
Metoden (Gausskvadratur med 3 vikter) är:

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f\left(-\sqrt{3/5}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{3/5}\right)$$

Vi transformerar interval $[0,3]$ för x, till $[-1,1]$ för t, med hjälp av formula:

$$x = \frac{b-a}{2}t + \frac{a+b}{2} = \frac{3-0}{2}t + \frac{3+0}{2}$$

Example

och integral $\int_0^3 e^{-x^2}dx$ för $f(x) = e^{-x^2}$ kan beräknas som

$$\begin{aligned}\int_0^3 e^{-x^2}dx &= \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\,dt \\ &\approx \frac{b-a}{2}\sum_{i=1}^{3}\omega_i f\left(\frac{b-a}{2}t_i + \frac{a+b}{2}\right) \\ &= \frac{3-0}{2}\cdot\left(\frac{5}{9}\cdot f\left(\frac{3-0}{2}t_1 + \frac{3+0}{2}\right)\right. \\ &\left. + \frac{8}{9}\cdot f\left(\frac{3-0}{2}t_2 + \frac{3+0}{2}\right) + \frac{5}{9}\cdot f\left(\frac{3-0}{2}t_3 + \frac{3+0}{2}\right)\right)\end{aligned}$$

i Gausspunkter

$$t_1 = -\sqrt{3/5}; t_2 = 0; t_3 = \sqrt{3/5};$$

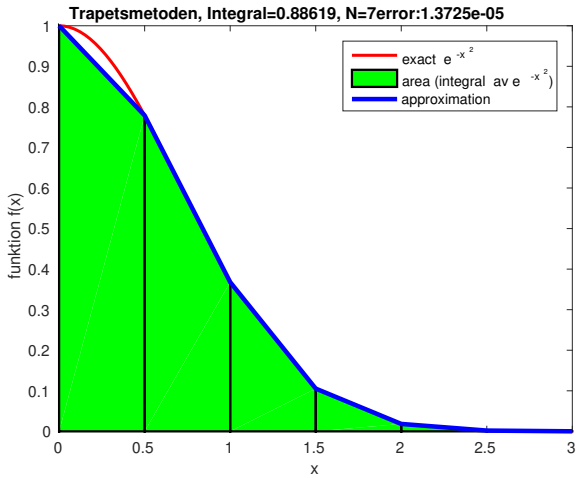
med vikter

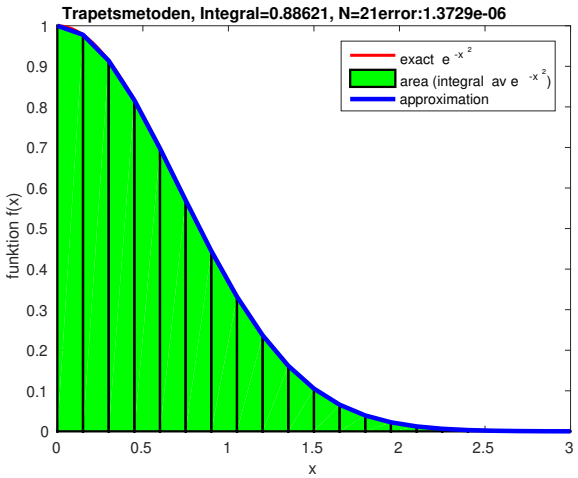
$$\omega_1 = 5/9; \omega_2 = 8/9; \omega_3 = 5/9.$$

Exempel: numerisk integration av $\int_0^3 f(x) = exp(-x^2)$

```
fun = @(x) exp(-x.^2)
% definition av integration intervalet [n,p]
n= 0.0; = 3.0
% adaptivt integral
Q =  integral(fun,n,p)
%  Gauss punkter
x(1) = -sqrt(3/5); x(2) = 0; x(3) = sqrt(3/5);
for i=1:3
t(i) = ((p-n)/2.0)*x(i) + (p+n)/2.0;
end
%vikter
omega(1)= 5/9;
omega(2)= 8/9;
omega(3) = 5/9;
Int = ((p-n)/2.0)*(omega(1)*fun(t(1))+ omega(2)*fun(t(2)) + ...
omega(3)*fun(t(3)))
```


Exempel: numerisk integration av $\int_0^3 f(x) = exp(-x^2)$





Vi får följande svar:
Exak lösningt: adaptivt metod i Matlab (integral): $Q = 0.8862$
Gausskvadratur med 3 vikter: $I_h = 0.8845$
Trapetsmetoden med 21 punkt: $I_h = 0.8862$
Trapetsmetoden med 3 punkter: $I_h = 0.9082$

Trapetsmetoden med 7 punkter: $I_h = 0.8862$

Ordinära differentialekvationer (ODE)

Vi kommer enbart att studera begynnelsevärdesproblem. Mer generellt, k-th ordningens ODE har implicit form

$$f(t, y, y', y'', ..., y^{(k)}) = 0,$$

var $y(t) \in \mathbb{R}^n$ är en vektor, som ska bestämmas, $f^{kn+n+1} \to \mathbb{R}^n$ är känd funktion.
Explicit form för ODE är följande:

$$y^{(k)} = f(t, y, y', y'', ..., y^{(k-1)}),$$

var $f^{kn+1} \to \mathbb{R}^n$ och $y(t) \in \mathbb{R}^n$ är en vektor, som ska bestämmas.

Ordinära differentialekvationer

Till exempel, följande ODE är i explicit form:

$$y'(t) = t^2 + \sin y(t),\ 3 < t \leq 10,\ y(3) = 4$$

t är tiden och $3 < t \leq 10$ anger det intervall där vi vill approximera lösningen. $y(3) = 4$ är ett begynnelsevillkor som anger y:s begynnelsevärde, 4, vid tiden $t = 3$. Normalt skriver man aldrig ut t i $y(t)$. Vi struntar även i intervallet (tiden i begynnelsevillkoret är vänster ändpunkt, och man får anta något lämpligt slutvärde). Problemet kan då formuleras

$$y' = \underbrace{t^2 + \sin y(t)}_{f(t,y)},\ y(3) = 4$$

Normalt vill vi studera ett generellt problem, vi skriver:

$$y' = f(t, y),\ y(t_0) = y_0$$

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.

ode45 i Matlab för $y' = t^2 + sin(y(t)),\ y(3) = 4$

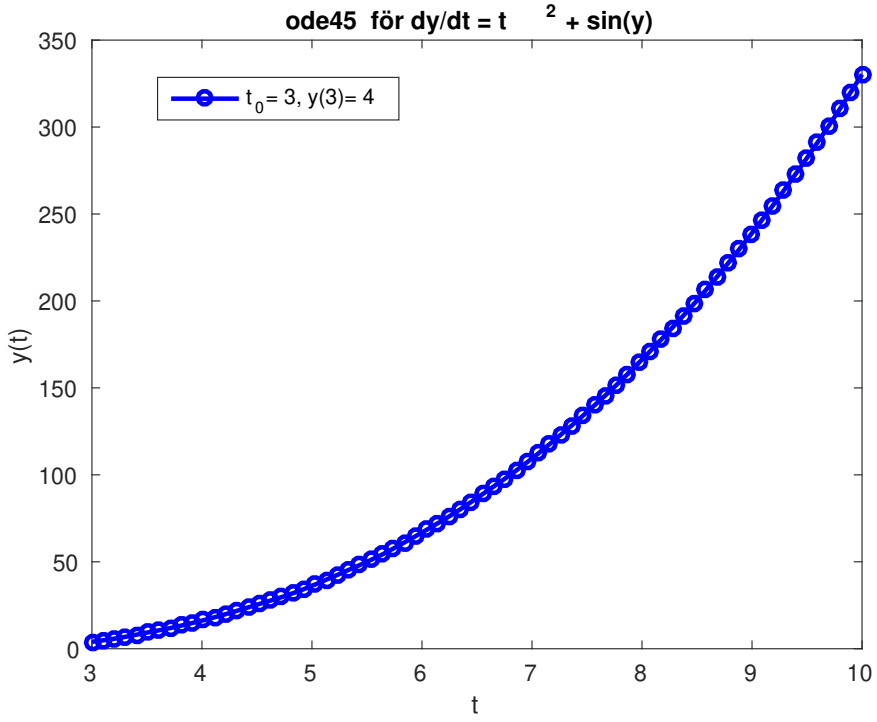
```
y0 = 4;            % begynnelsevarden
t0 = 3;           % begynnelsetid
ts = 10;            % slut-tid

h= 0.1;            %steglangd h
N = (ts - t0)/h    %antal punkter

[t, y] = ode45(@func_example1, linspace(t0, ts, N), y0);

figure
plot(t, y, 'b -o', 'LineWidth',2)

  xlabel('t');
ylabel('y(t)')
legend('t_0= 3, y(3)= 4')
title('ode45  f\"or dy/dt = t^2 + sin(y)')
```


ODE (Matlabs ode45)

Vi definerar separat matlabs-file func_example1.m med funktion

```
function [dy] = func_example1(t, y)

dy = 0;
dy =  t^2 + sin(y);
```

Exempel: begynnelsevärdesproblem för $y' = t^2 + \sin y(t)$



Ordinära differentialekvationer

Lösningsmetoderna genererar approximationer till lösningen för en uppsättning tidpunkter: $(t_0, y_0), (t_1, y_1), ..., (t_n, y_n)$, 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 värde är $y(t_k)$.

Senare kommer system av ekvationer. Sådana behövs för att vi skall kunna lösa problem som innehåller högre derivator., t.ex.

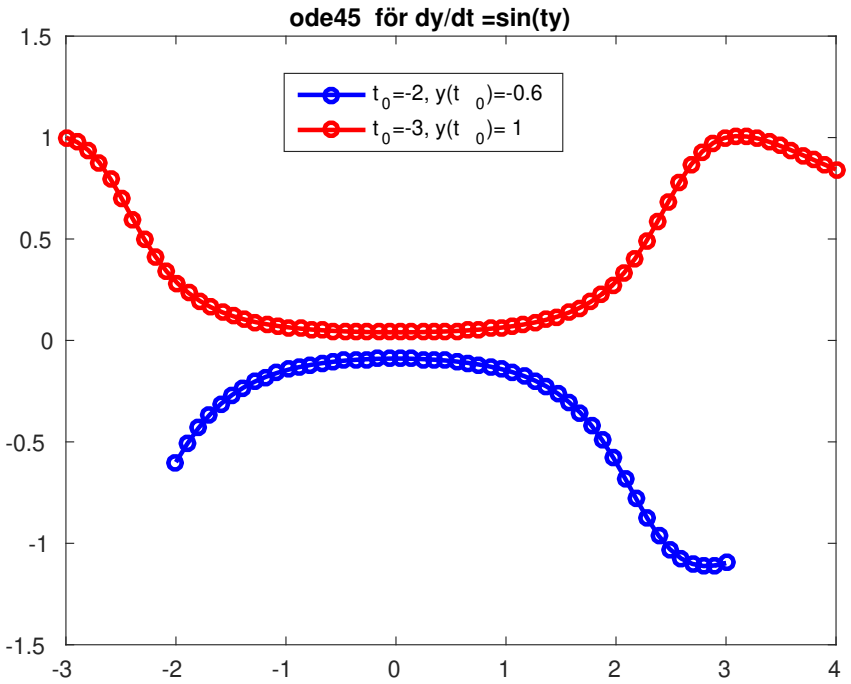
$$y''' = t + 2y'' + (y')^2 + \sin y,\ y(0) = 2,\ y'(0) = -3,\ y''(0) = -4$$

Ordinära differentialekvationer

Låt oss studera problemet $y' = 1$. Detta är inget begynnelsevärdesproblem (eftersom vi saknar $y(t_0) = y_0$). Ett problem av detta slag har oändligt många lösningar, i detta fall $y(t) = t + c$, där c är ett godtyckligt reellt tal. När vi ger ett begynnelsevillkor 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 av lösningsmängden även för problemet $y' = f(t, y)$. Detta kan göras genom att i ett lämpligt antal punkter i (t, y)-planet rita en vektor som svarar mot den derivata som lösningen måste ha enligt ekvationen $y' = f(t, y)$.

Det finns begynnelsevärdesproblem som saknar, eller har flera lösningar. Det kan också vara så att $y(t)$ inte existerar för alla $t > t_0$ (om $y(t) \to \infty$ t.ex.).

Example: begynnelsevärdesproblem för $y' = sin(ty)$



Ordinära differentialekvationer (Eulers metod)

En enkel lösningsmetod: Vi startar i (t_0, y_0) (som är känd) och tar sedan ett litet steg utmed tangenten till lösningen (som kan beräknas 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:

$$y_{k+1} = y_k + hf(t_k, y_k),\ k = 0, 1, 2, \ldots$$

eller utskrivet
$y_1 = y_0 + hf(t_0, y_0),\ y_2 = y_1 + hf(t_1, y_1),\ y_3 = y_2 + hf(t_2, y_2), \ldots$

Example

$y' = \sin(ty),\ y(-1) = 1$. Så $t_0 = -1$ och $y_0 = 1$ och $f(t, y) = \sin(ty)$. Om $h = 0.1$ får vi approximationerna

$$y_1 = y_0 + hf(t_0, y_0) = 1 + 0.1 \sin(-1 \cdot 1) \approx 0.9159$$
$$y_2 = y_1 + hf(t_1, y_1) \approx 0.9159 + 0.1 \sin(-0.9 \cdot 0.9159) \approx 0.8425$$
$$y_3 = y_2 + hf(t_2, y_2) \approx 0.8425 + 0.1 \sin(-0.8 \cdot 0.8425) \approx 0.7801 \text{ osv.}$$

Framåt Eulers metod i Matlab för $y' = \sin(ty),\ y(-1) = 1$

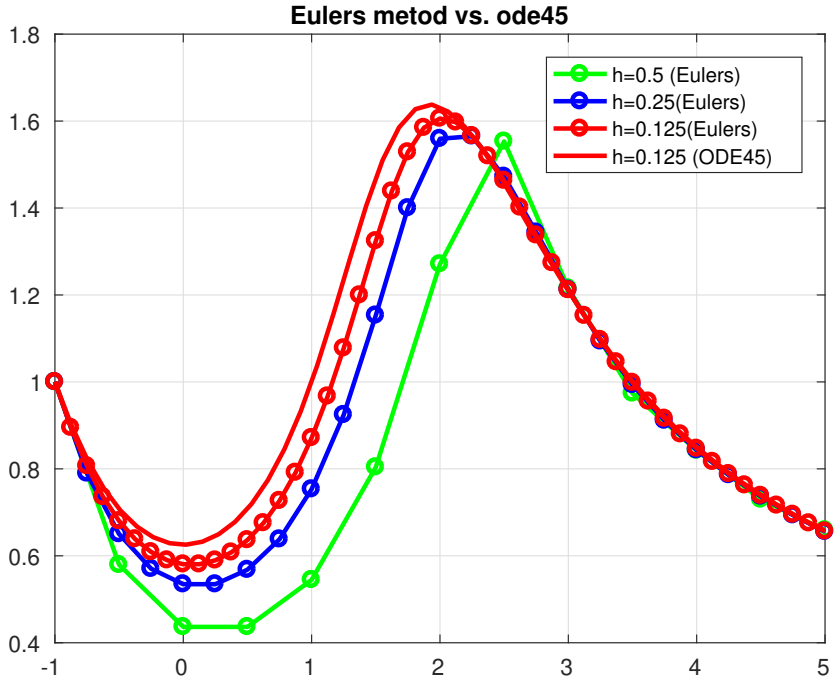
```
t0 = -1;          % begynnelsetid
ts = 5;           % slut-tid
h= 0.5;           %steglangd h
N = (ts - t0)/h   % antal punkter

  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
  y(1) = 1;         % begynnelsevarden

for k = 1:N
    y(k+1) = y(k) + h*func_example3(t(k),y(k));
    t(k+1) = t(k) + h;

 end
 figure
plot(t, y, 'g -o ', 'LineWidth',2)
```

Framåt Eulers metod i Matlab för $y' = \sin(ty),\ y(-1) = 1$

Vi definerar separat matlabs-file func_example3.m med funktion

```
function dy = func_example3(t, y)

dy = 0;
dy =  sin(t*y);
```

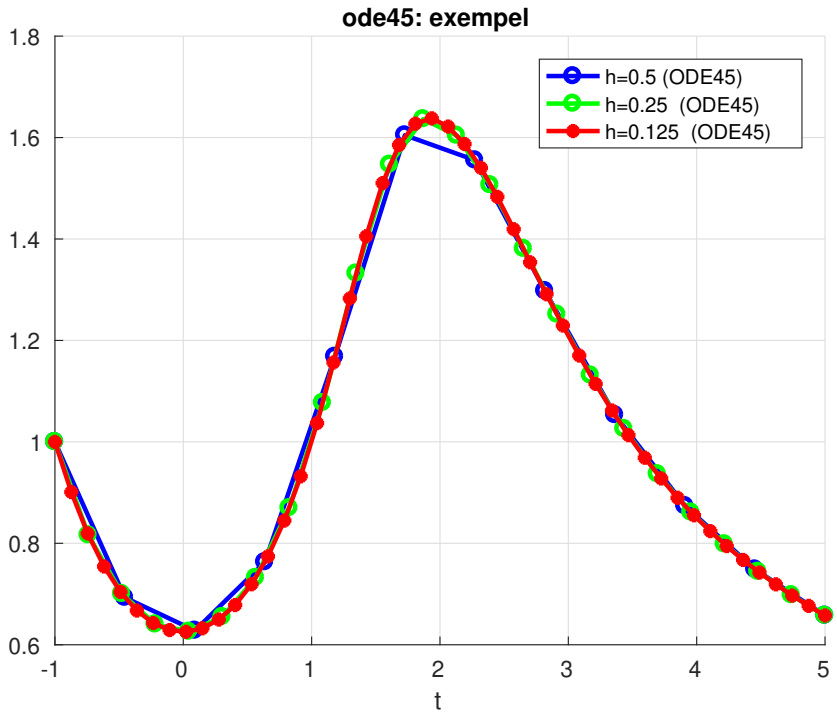

ODE: Eulers metod vs ode45 för $y' = \sin(ty),\ y(-1) = 1$



ode45 i Matlab för $y' = \sin(ty),\ y(-1) = 1$

```
y0 = 1;              % begynnelsevarden
t0 = -1;             % begynnelsetid
ts = 5;              % slut-tid

h= 0.5;              %steglangd h
N = (ts - t0)/h      %antal punkter

[t, y] = ode45(@func_example3, linspace(t0, ts, N), y0);

figure
hold on
plot(t, y(:, 1), 'b -o', 'LineWidth',2)
```


ODE: ode45 i Matlab



Ordinära differentialekvationer (Eulers metod)

Alternativa härledningar av Eulers metod: Först Taylorutveckling:

$$y(t_k + h) = y(t_k) + hy'(t_k) + \frac{h^2}{2}y''(t_k) + ...$$

Nu är $y'(t_k) = f(t_k, y(t_k))$ och $t_{k+1} = t_k + h$ så att:

$$y(t_{k+1}) \approx y(t_k) + hf(t_k, y(t_k))$$

Vi approximerar nu $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ och får:

$$y_{k+1} = y_k + hf(t_k, y_k)$$

Härledning med hjälp av 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 $y_{k+1} = y_k + hf(t_k, y_k)$.

ODE (system av ekvationer)

$$u''' = u'' - 2tu' + u^2 - t + 1 \quad \begin{cases} u(3) &= 2 \\ u'(3) &= -1 \\ u''(3) &= 0 \end{cases}$$

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.

ODE (system av ekvationer)

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

$$f(t, y) = \begin{bmatrix} y_2 \\ y_3 \\ y_3 - 2ty_2 + y_1^2 - t + 1 \end{bmatrix}, \; y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

Alla metoder vi har sett kan enkelt generaliseras till systemfallet. Skalära y_k byts ut 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 $t_0 = 3$ och $h = 0.1$:

$$y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \; y^{(1)} = y^{(0)} + hf(t_0, y^{(0)})$$

ODE (system av ekvationer)

Dvs.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \end{bmatrix} = \begin{bmatrix} y_1^{(0)} \\ y_2^{(0)} \\ y_3^{(0)} \end{bmatrix} + h \begin{bmatrix} y_2^{(0)} \\ y_3^{(0)} \\ y_3^{(0)} - 2t_0 y_2^{(0)} + \left(y_1^{(0)}\right)^2 - t_0 + 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.9 \\ -1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} + 0.1 \begin{bmatrix} -1 \\ 0 \\ 0 - 2 \cdot 3(-1) + 2^2 - 3 + 1 \end{bmatrix}$$

och så vidare.

Lösning av system av ekvationer med Matlabs ode45

För att lösa system av ekvationer

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = y_3 - 2ty_2 + y_1^2 - t + 1 \end{cases}, \quad \begin{cases} y_1(3) = 2 \\ y_2(3) = -1 \\ y_3(3) = 0 \end{cases}$$

i tiden $t = [3, 10]$ vi kan också använda Matlabs ode45.

ODE (Matlabs ode45)

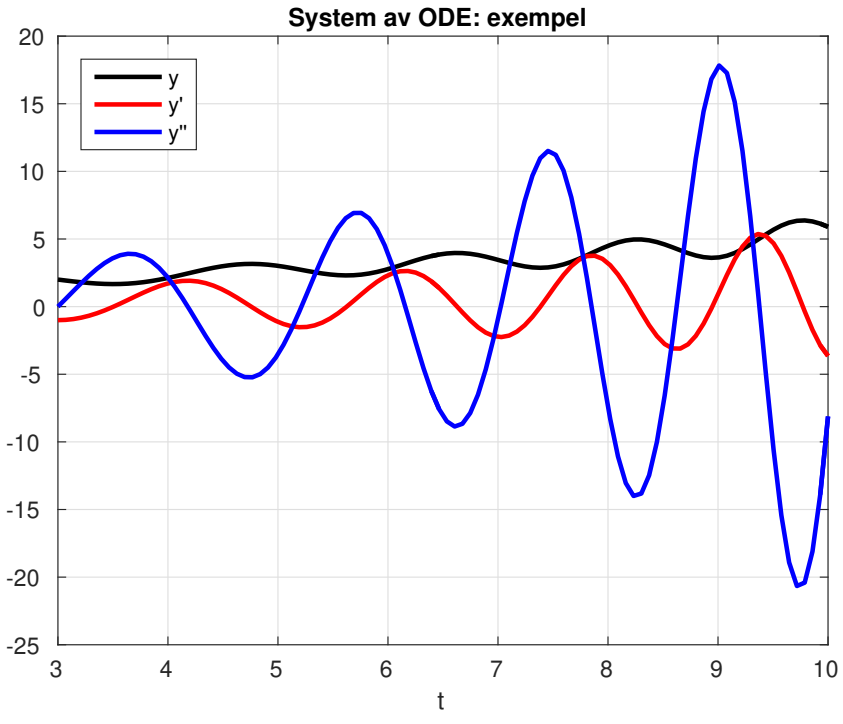
```
y0 = [2 -1 0]';  % begynnelsevarden
t0 = 3;          % begynnelsetid
ts = 10;         % slut-tid

[t, y] = ode45(@f, linspace(t0, ts, 100), y0);

figure(1); hold off
plot(t, y(:, 1), 'k-', t, y(:, 2), 'r-', ...
     t, y(:,3), 'b-')
legend({'y', 'y''', 'y'''''}, ...
       'Location', 'NorthWest','LineWidth',2)
xlabel('t')
grid on
title(' System av ODE: exempel')
```

ODE (Matlabs ode45)

Vi måste definera separat matlabs-file f.m med funktion

```
function dy = f(t, y)

dy = zeros(3,1);

dy(1) = y(2);
dy(2) = y(3);
dy(3) = y(3)-2*t*y(2)+y(1)^2-t+1;
```


ODE (Matlabs ode45)



Övning

Sätt upp Eulers metod för problemet

$$y' = t + 2y, \ y(0) = 1$$

och beräkna $y_k, k = 0, 1, 2, 3$ med $h = 0.1$.
Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k), y_0 = y(t_0).$$

Övning

Svar:
Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k), y_0 = y(t_0).$$

I detta fall är

$$f(t, y) = t + 2y, f(t_k, y_k) = t_k + 2y_k, t_0 = 0, y(0) = 1, h = 0.1. \quad (48)$$
$$y_{k+1} = y_k + h(t_k + 2y_k), y_0 = 1. \quad (49)$$

så vi får följande approximationer:

$$y_0 = 1, \quad (50)$$
$$y_1 = y_0 + 0.1(t_0 + 2 \cdot y_0) = 1 + 0.1(0 + 2 \cdot 1) = 1.2, \quad (51)$$
$$y_2 = y_1 + 0.1(t_1 + 2 \cdot y_1) = 1.2 + 0.1(0.1 + 2 \cdot 1.2) = 1.45, \quad (52)$$
$$y_3 = y_2 + 0.1(t_2 + 2 \cdot y_2) = 1.45 + 0.1(0.2 + 2 \cdot 1.45) = 1.76. \quad (53)$$

Framåt Eulers metod i Matlab för $y' = t + 2y,\ y(0) = 1$

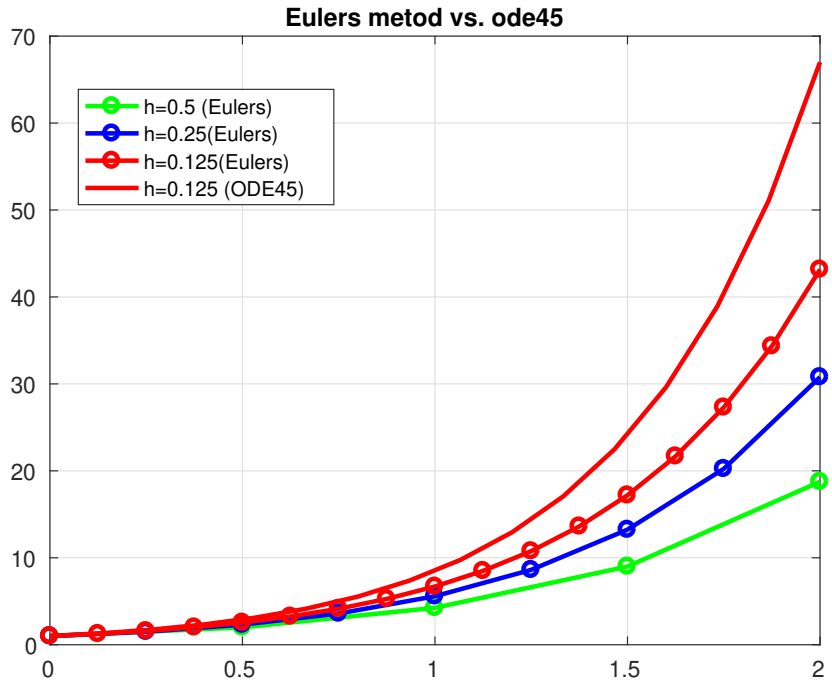
```
t0 = 0;          % begynnelsetid
ts = 2;           % slut-tid
h= 0.1;           %steglangd h
N = (ts - t0)/h   % antal punkter

  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
  y(1) = 1;         % begynnelsevarden

for k = 1:N
    y(k+1) = y(k) + h*func_example4(t(k),y(k));
    t(k+1) = t(k) + h;

 end
 figure
plot(t, y, 'g -o ', 'LineWidth',2)
```


Framåt Eulers metod i Matlab för $y' = t + 2y,\ y(0) = 1$

Vi definerar separat matlabs-file func_example4.m med funktion

```
function dy = func_example4(t, y)

dy = 0;
dy = t + 2*y;
```

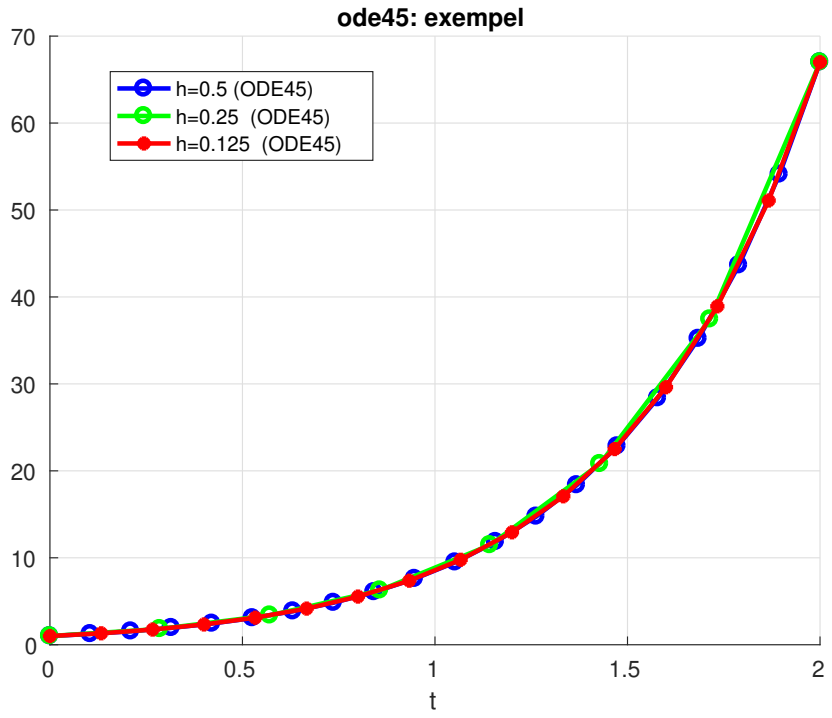
ODE: Eulers metod vs ode45 för $y' = t + 2y,\ y(0) = 1$



ode45 i Matlab för $y' = t + 2y,\ y(0) = 1$

```
y0 = 1;              % begynnelsevarden
t0 = 0;             % begynnelsetid
ts = 2;              % slut-tid

h= 0.1;              %steglangd h
N = (ts - t0)/h     %antal punkter

[t, y] = ode45(@func_example4, linspace(t0, ts, N), y0);

figure
hold on
plot(t, y(:, 1), 'b -o', 'LineWidth',2)
```

ODE: ode45 i Matlab för $y' = t + 2y,\ y(0) = 1$



ODE (felkällor)

Felkällor :

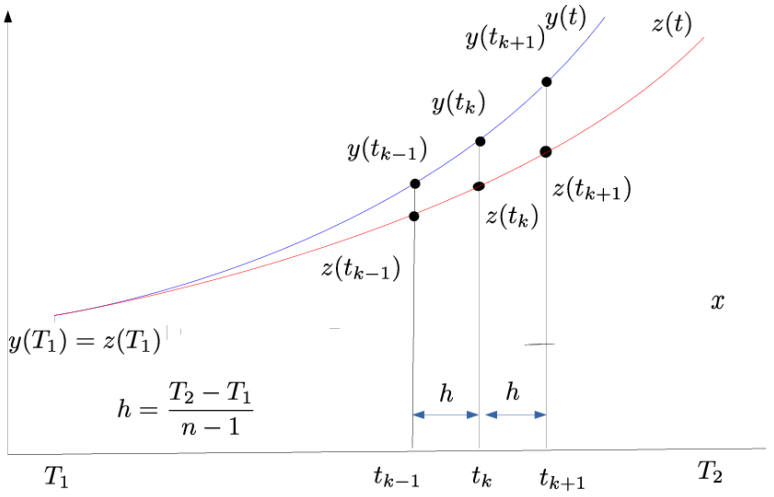
- trunkeringsfel; i Eulers metod trunkerar vi Taylorutvecklingen (approximerar med tangenten)
- avrundningsfel; normalt inte så viktigt

ODE (felkällor)

Trunkeringsfel i Taylorutvecklingen har 2 olika typer:

Lokalt fel: felet som uppstår i ett steg när man betraktar startpunkten, (t_{k-1}, y_{k-1}), som exakt.

Globalt fel: felet mellan approximativ och exakt lösning, $y_k - y(t_k)$



ODE (Ordning)

Olika metoder har olika ordning: en metod har ordning p om det lokala felet är av storleksordning 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:

$$z' = \lambda z,\ z(t_{k-1}) = y_{k-1} \Rightarrow z(t) = e^{\lambda(t-t_{k-1})} y_{k-1}$$

Härledning. Lösningen sökes på formen $z(t) = Ce^{\lambda t}$, då $z(t_{k-1}) = Ce^{\lambda t_{k-1}} = y_{k-1} \Rightarrow C = y_{k-1}e^{-\lambda t_{k-1}} \Rightarrow z(t) = e^{\lambda(t-t_{k-1})} y_{k-1}$.
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:

$$y_k = y_{k-1} + hf(t_{k-1}, y_{k-1}) = (1 + \lambda h) y_{k-1}$$

ODE: lokala felet, ordning

Det lokala felet blir:

$$y_k - z(t_k) = (1+\lambda h)y_{k-1} - e^{\lambda h}y_{k-1} =$$

$$\left[1+\lambda h - \left[1+\lambda h + \frac{(\lambda h)^2}{2} + ...\right]\right] y_{k-1} = -\left[\frac{(\lambda h)^2}{2} + ...\right] y_{k-1}$$

som är $\mathcal{O}(h^2)$, så Eulers metod har ordning ett (är en första ordningens metod).

Härledning. Från Taylor's formel:

$$f(t+\lambda h) = f(t) + f'(t)\lambda h + \frac{f''(t)(\lambda h)^2}{2!} + ...$$

För $t = 0$ kan vi skriva om den:

$$f(\lambda h) = f(0) + f'(0)\lambda h + \frac{f''(0)(\lambda h)^2}{2!} + ...$$

För $f(t) = e^t$ har vi: $f(t) = e^t, f'(t) = ...f^k(t) = e^t$ så att $f(0) = e^0 = 1, f'(0) = ...f^k(0) = e^0 = 1$. Därför

$$f(\lambda h) = 1 + 1\cdot\lambda h + 1\cdot\frac{(\lambda h)^2}{2!} + ...$$

ODE: globala felet, ordning

Nu till det globala felet, $\underbrace{y_k}_{approxim} - \underbrace{y(t_k)}_{exakt}$, 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

$$y(t_k) = e^{\lambda t_k}y_0 \text{ och } y_k = (1+\lambda h)^k y_0,$$

Varför?

$y_1 = y_0 + h\lambda y_0 = (1+h\lambda)y_0$.

$y_2 = y_1 + h\lambda y_1 = (1+h\lambda)y_1 = (1+\lambda h)^2 y_0$ etc.

ODE (Ordning)

Eftersom $t_k = kh$, får vi följande uttryck för det globala felet:

$$\underbrace{y_k}_{\text{approxim}} - \underbrace{y(t_k)}_{\text{exakt}} = \underbrace{(1+\lambda h)^k y_0}_{\text{approxim}} - \underbrace{e^{\lambda t_k} y_0}_{\text{exakt}} = \underbrace{(1+\lambda h)^k y_0}_{\text{approxim}} - \underbrace{e^{\lambda k h} y_0}_{\text{exakt}} =$$

$$\left[1 + k\lambda h + \frac{k(k-1)}{2}(\lambda h)^2 + ...\right] y_0 - \left[1 + k\lambda h + \frac{(k\lambda h)^2}{2} + ...\right] y_0.$$

Härledning. Från Taylor's formel:

$$f(t+\lambda h) = f(t) + f'(t)\lambda h + \frac{f''(t)(\lambda h)^2}{2!} + ...$$

$$f(t+k\lambda h) = f(t) + f'(t)k\lambda h + \frac{f''(t)(k\lambda h)^2}{2!} + ...$$

För $t = 0$ kan vi skriva om de:

$$f(\lambda h) = f(0) + f'(0)\lambda h + \frac{f''(0)(\lambda h)^2}{2!} + ...$$

$$f(k\lambda h) = f(0) + f'(0)k\lambda h + \frac{f''(0)(k\lambda h)^2}{2!} + ...$$

ODE (Ordning)

För $f(t) = e^t$ har vi: $f(t) = e^t, f'(t) = ...f^{(k)}(t) = e^t$ så att
$f(0) = e^0 = 1, f'(0) = ...f^{(k)}(0) = e^0 = 1$. Därför

$$f(k\lambda h) = 1 + 1 \cdot k\lambda h + 1 \cdot \frac{(k\lambda h)^2}{2!} + ...$$

För $f(t) = (1+t)^k$ har vi:
$f'(t) = k(1+t)^{k-1}, f''(t) = k(k-1)(1+t)^{k-2}$ så at
$f(0) = 1, f'(0) = k, f''(0) = k(k-1)$.

$$f(\lambda h) = 1 + k\lambda h + \frac{k(k-1)(\lambda h)^2}{2} + ...$$

ODE (Ordning)

Eftersom $t_k = kh$, får vi följande uttryck för det globala felet:

$$\underbrace{y_k}_{approxim} - \underbrace{y(t_k)}_{exakt} = \underbrace{(1+\lambda h)^k y_0}_{approxim} - \underbrace{e^{\lambda t_k} y_0}_{exakt} = \underbrace{(1+\lambda h)^k y_0}_{approxim} - \underbrace{e^{\lambda k h} y_0}_{exakt} =$$

$$\left[1 + k\lambda h + \frac{k(k-1)}{2}(\lambda h)^2 + ...\right] y_0 - \left[1 + k\lambda h + \frac{(k\lambda h)^2}{2} + ...\right] y_0 =$$

$$\frac{-k}{2}(\lambda h)^2 y_0 + ... = -\frac{1}{2}\lambda^2 \underbrace{(hk)}_{t_k} y_0 h + ... = -\frac{1}{2}\lambda^2 t_k y_0 h + ...$$

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.

ODE (Ordning)

Vi kan försöka skapa metoder av högre ordning, t.ex. genom att använda tidigare punkter; en så kallad flerstegsmetod. T.ex.

$$y_{k+1} = y_k + h\left[\frac{3}{2}f(t_k, y_k) - \frac{1}{2}f(t_{k-1}, y_{k-1})\right],$$

som har ordning två. För att starta metoden kan vi ta ett Euler-steg.

Här är en tredje ordningens metod:

$$y_{k-1} = y_k + h\left[\frac{23}{12}f(t_k, y_k) - \frac{4}{3}f(t_{k-1}, y_{k-1}) + \frac{5}{12}f(t_{k-2}, y_{k-2})\right]$$

En annan metod av andra ordningen är Heuns metod:

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))]$$

Detta är en enstegsmetod.

ODE: flerstegsmetoder

Här är härledningen för flerstegsmetod, som har ordning två:

$$y_{k+1} = y_k + h\left[\frac{3}{2}f(t_k, y_k) - \frac{1}{2}f(t_{k-1}, y_{k-1})\right].$$

Approximera

$$y''(t) \approx \frac{y'(t) - y'(t-h)}{h}$$

så att i Taylorsutvecklingen kan vi skriva den approximation:

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + O(h^3) \approx y(t) + hy'(t) + \frac{h^2}{2}\frac{y'(t) - y'(t-h)}{h}$$
$$= y(t) + hf(t,y) + \frac{h}{2}(f(t,y) - f(t-h, y-h)) = y(t) + \frac{h}{2}(3f(t,y) - f(t-h, y-h)).$$

Detta leder till metoden:

$$y_{k+1} = y_k + \frac{h}{2}(3f(t_k, y_k) - f(t_{k-1}, y_{k-1})),$$

som är andra ordningens flerstegsmetod.

ODE: flerstegsmetoder

Här är härledningen för Heuns metod:

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))]. \qquad (54)$$

Approximera

$$y''(t) \approx \frac{y'(t+h) - y'(t)}{h}$$

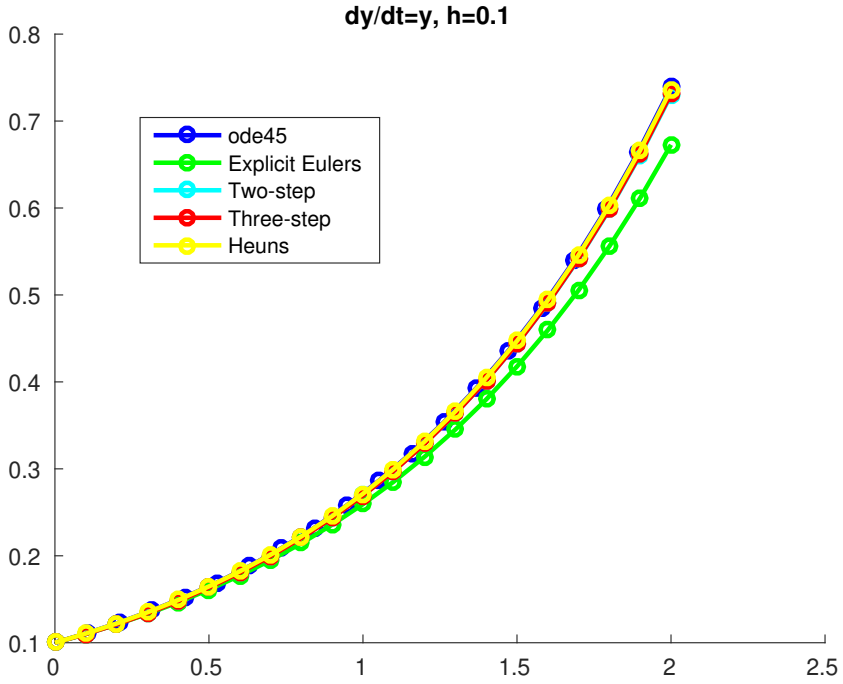
så att

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + O(h^3) \approx y(t) + hy'(t) + \frac{h^2}{2}\frac{y'(t+h) - y'(t)}{h}$$
$$= y(t) + hf(t,y) + \frac{h}{2}(f(t+h, y+h) - f(t,y)) = y(t) + \frac{h}{2}(f(t+h, y+h) + f(t,y)).$$

Detta leder till enstegsmetoden (54) ovan, and kan också skrivas som

$$\tilde{y}_{k+1} = y_k + hf(t_k, y_k),$$
$$y_{k+1} = y_k + \frac{h}{2}(f(t_{k+1}, \tilde{y}_{k+1}) + f(t_k, y_k)).$$

ODE: olika metoder



ODE: existens och entydighet

Låt $D = [a, b] \times \Omega \subset \mathbb{R}^{n+1}$ är en avgränsad domän. Anta att $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$ är en kontinuerlig funktion för t i $[a, b]$ och Lipschitz kontinuerlig funktion för y i D, i andra ord
$\exists L = const. : \forall t \in [a, b], \forall y_1, y_2 \in D$,

$$\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\|. \tag{55}$$

Funktionen ska vara Lipschitz kontinuerlig om f är deriverbar, och i så fal man kan ta

$$L = \max_{t,y \in D} \|J_f(t, y)\|$$

var $J_f(t, y)$ är Jacobian matrix av funktion f med avseende på y: $\{J_f(t, y)_{ij}\} = \frac{\partial f_i}{\partial y_j}$.
Man kan bevisa att för all punkter $(t_0, y_0) \in D$ existerar subinterval i $[a, b]$, som innehåller t_0, så att där existerar entydig lösning y för problemet $y' = f(t, y), \quad y(t_0) = y_0$ så att den lösningen är också entydig på randen av D.

ODE: existens och entydighet

Sats

Låt y_1, y_2 ska vara lösningar till IVP $y_i' = 0, \quad y_i(t_0) = {y_i}^0, i = 1, 2$. Då

$$\|y_1 - y_2\| \leq e^{L(t-t_0)} \|{y_1}^0 - {y_2}^0\|$$

Sats

Låt y_1, y_2 ska vara lösningar till IVP $y_i' = f_i(t, y_i), \quad y_i(t_0) = {y_i}^0, i = 1, 2$ med störda initvillkor och störda höger leden, då

$$\|y_1 - y_2\| \leq e^{L(t-t_0)} \|{y_1}^0 - {y_2}^0\| + \frac{e^{L(t-t_0)} - 1}{L} \|f_1 - f_2\|,$$

var

$$\|f_1 - f_2\| = \max_{t,y \in D} \|f_1 - f_2\|$$

Båda satser visar att den unika lösningen är en kontinuerlig funktion av problemdata och därmed är problemet välkonditionerat. Men termen $e^{L(t-t_0)}$ innebär att lösningen kan vara mycket känslig för störningar och kan vara instabillt i tiden. Proof för båda satser finns i Theorem 2.8 av

ODE (Stabilitet)

Hur ändras lösningen vid små ändringar i problemet? Om lösningskurvorna går ihop eller isär avgörs av det lokala utseendet på riktningsfältet.
En lösning av ODE

$$y' = f(t, y), y(t_0) = y_0$$

är stabil om för varje $\epsilon > 0$ existerar $\delta > 0$ så att för annan lösning $\tilde{y}$, som satisfierar ekvationen

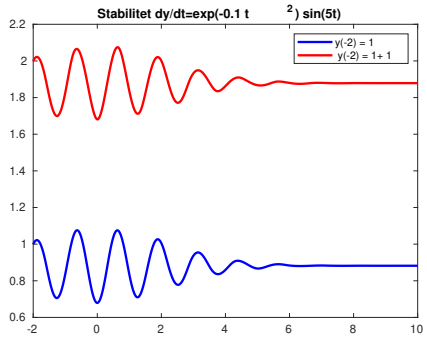
$$\tilde{y}' = f(t, \tilde{y}), \tilde{y}(t_0) = \tilde{y}_0$$

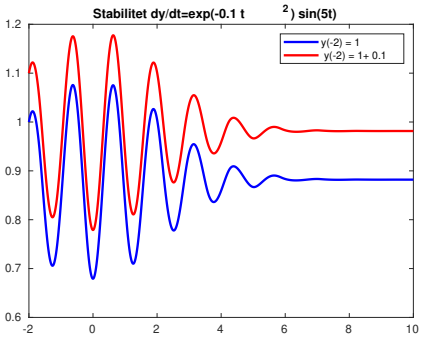
gäller att $\|\tilde{y}(t_0) - y(t_0)\| \leq \delta$ då $\|\tilde{y}(t) - y(t)\| \leq \epsilon \quad \forall t \geq t_0$.

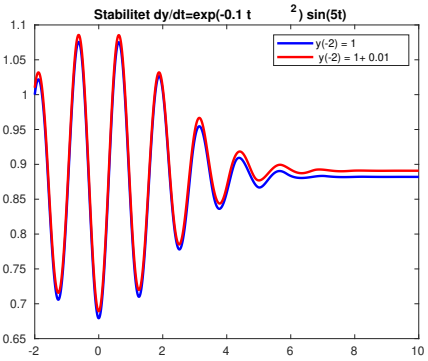
I andra ord, en lösning är stabil om två lösningar kan fås att ligga godtyckligt nära varandra (för $t \geq t_0$) givet att vi stör tillräckligt lite.
Lösningar är asymptotically stabila om

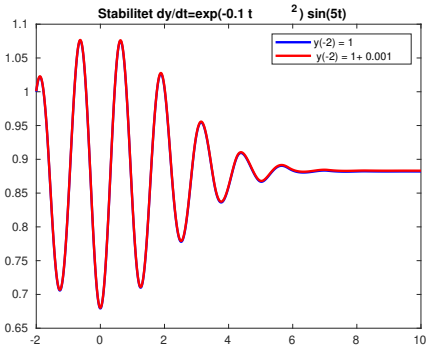
$$\lim_{t \to \infty} \|\tilde{y}(t) - y(t)\| = 0.$$

Exempel på stabilitet för lösningen av $dy/dt = exp(3sin(t))sin(5t)$, $y(-2) = 1$ för små ändringar δ i $y(-2)$, var $\delta = 1, 0.1, 0.01, 001$. Vi observerar att två lösningar har ett avgränsat avstånd från varandra och darför lösningar är stabila, men inte asymptotically stabila.









Givet modellproblemet:

$$y' = \lambda y,\ y(0) = y_0,\ \lambda \in \mathbb{C} \tag{56}$$

så är lösningen stabil om λ har negativ realdel. Om realdelen är positiv är lösningen instabil. Detta kan generaliseras till ickelinjära problem och system av sådana.
Vi kan se att lösningen för (56) är:

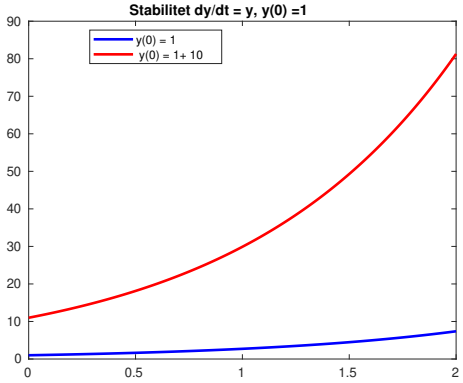
$$y(t) = y_0 e^{\lambda t}, y(0) = y_0. \tag{57}$$

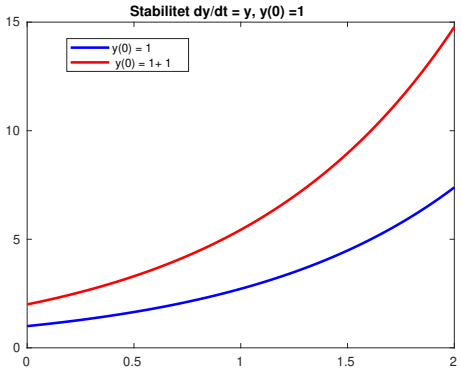
Om λ är komplex då $e^{\lambda t} = e^{(a+ib)t} = e^{at}e^{ibt} = e^{at}(\cos(bt) + i\sin(bt))$.

- Om $Re(\lambda) = a > 0$ i (57), då alla lösningar växer exponentiellt och varje två lösningar avviker från varandra, lösningar är instabila.
- Om $Re(\lambda) = a < 0$ i (57), då alla lösningar förfaller exponentiellt och varje två lösningar konvergerar till varandra, lösningar är stabila och också asymptotically stabila.
- Om $Re(\lambda) = a = 0$ i (57), då alla lösningar oscillate, men varje två lösningar har ett avgränsat avstånd från varandra och darför lösningar är stabila, men inte asymptotically stabila.

ODE: exempel på stabilitet

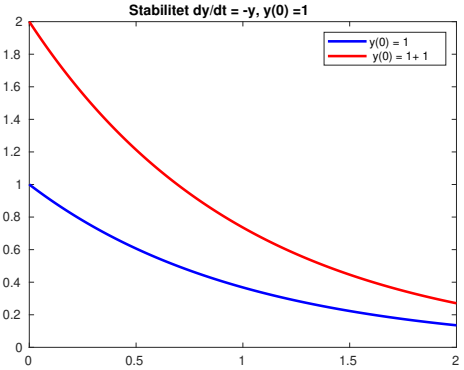
Exempel på stabilitet för lösningen av $dy/dt = y(t), y(0) = 1$ för ändringar δ i $y(0)$, var $\delta = 1, 10$. Vi observerar att två lösningar växer exponentiellt och avviker från varandra, lösningar är instabila.

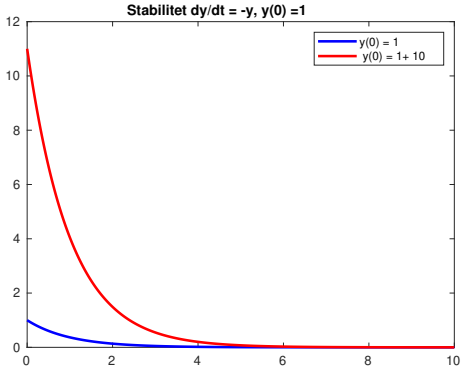




ODE: exempel på stabilitet

Exempel på stabilitet för lösningen av $dy/dt = -y(t), y(0) = 1$ för ändringar δ i $y(0)$, var $\delta = 1, 10$. Vi observerar att två lösningar konvergerar till varandra, lösningar är stabila och också asymptotically stabila.





ODE (Adaptivitet)

De flesta ODE-lösare är adaptiva, dvs. de försöker anpassa steglängden τ så att det lokala felet

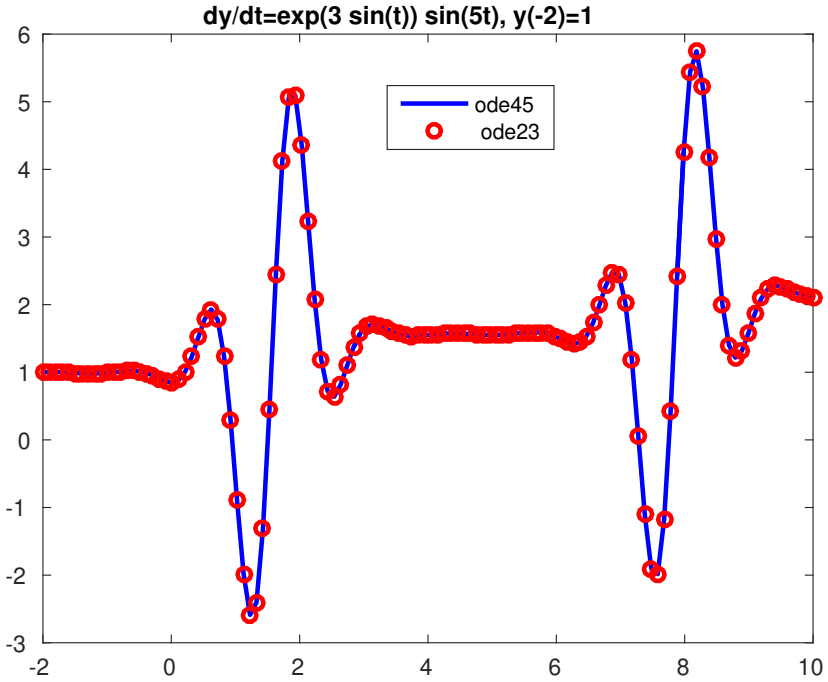
$$e_{loc} = C\tau^p R([x_{i-1}, x_i])$$

(med residualen R och constant C) underskrider en tolerans *tol* given av programmets användare: $e_{loc} \leq tol$:

$e_{loc} = Const * \tau^p * R([x_{i-1}, x_i])$
`if` $e_{loc} > tol$
$\tau = \tau/2$ $\quad$ *%delar intervalet* $[x_{i-1}, x_i]$ *till* 2
`end`

I vissa fall består programmet av en familj av metoder av olika ordning. Programmet kan då även variera ordningen.

ode45 vs. ode23



ODE (Styva problem)

Det är vanligt med så kallade styva problem (stiff). Dessa uppkommer t.ex. när man har snabba transienter: vi har icke stabilt problem. Om vi använder en vanlig ode-lösare på ett styvt problem tvingas lösaren ta mycket korta steg för att 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 $k \to \infty$?

$$y_{k+1} = y_k + hf(t_k, y_k) = y_k + h\lambda y_k = (1 + h\lambda)y_k.$$

När gäller att $y_k \to 0$? Jo, då:

$$|1 + h\lambda| < 1$$

ODE (Styva problem)

dvs. om $\lambda \in \mathbb{R}$ (och $\lambda < 0$),

$$0 < h|\lambda| < 2$$

Antag nu att λ är et 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.

ODE: implicita metoder

Lösningen är implicita metoder. Bakåt Euler:

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

Stabilitet? Testa $y' = \lambda y$

$$y_{k+1} = y_k + h\lambda y_{k+1}$$

så att

$$y_{k+1} = (1 - h\lambda)^{-1} y_k \Rightarrow y_{k+1} = (1 - h\lambda)^{-k} \text{ ty } y_0 = 1.$$

När är $|(1 - h\lambda)^{-1}| < 1$? Om $\lambda < 0$ (reellt) så är $|(1 - h\lambda)^{-1}| < 1$ för alla $h > 0$!

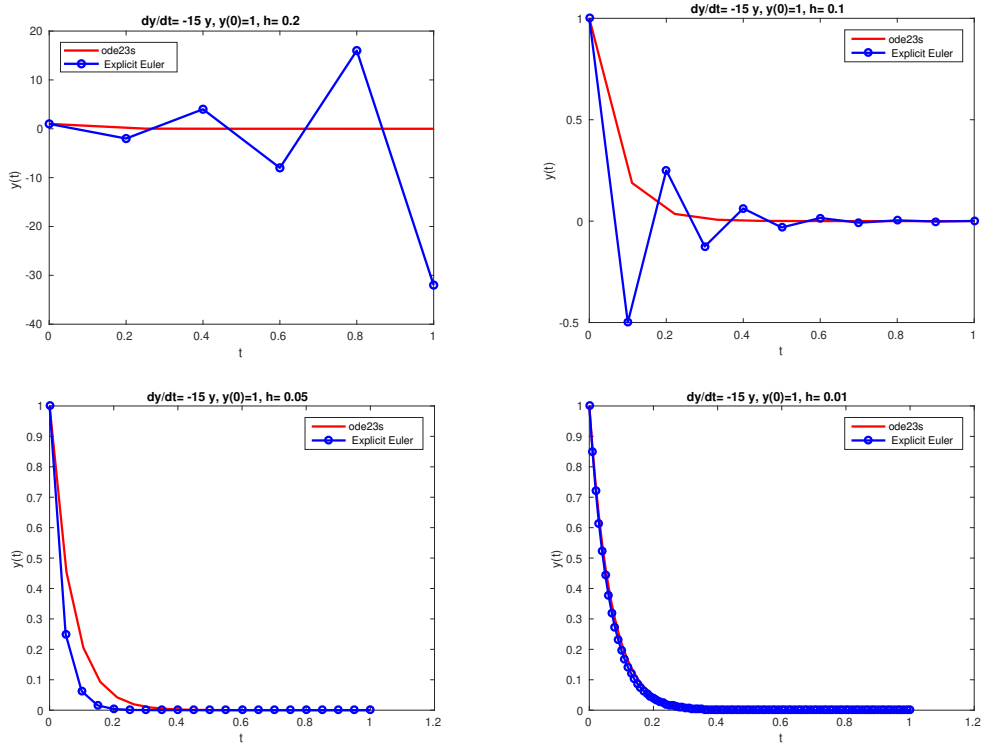
Detta innebär givetvis inte att vi kan ta godtyckligt långa steg. Tar vi för långa steg blir felet för stort. Implicita metoder har den nackdelen att vi måste lösa en (normalt ickelinjär) ekvation för att bestämma y_{k+1}.

I en explicit metod, som Eulers metod, är detta inte nödvändigt. Det finns implicita metoder av högre ordning, 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.

ODE (Styva problem)



ODE (Styva problem): ode23s i Matlab för $y' = -20y,\ y(0) = 1$

```
y0 = 1;            % begynnelsevarden
t0 = 0;           % begynnelsetid
ts = 1.5;            % slut-tid
h= 0.1;            %steglangd h
N = (ts - t0)/h    %antal punkter
[t, y] = ode23s(@func_stiff, linspace(t0, ts, N), y0);
figure
plot(t, y, 'b -o', 'LineWidth',2)
  xlabel('t');
ylabel('y(t)')
legend('t_0= 0, y(0)= 1')
title('ode23s  f\"or dy/dt = -20y')
```


ODE (Matlabs ode23s)

Vi definerar separat matlabs-file func_stiff.m med funktion

```
function [dy] = func_stiff(t, y)

dy = -20.0*y
```

Framåt Eulers metod för $y' = -20y,\ y(0) = 1$

```
% stabilitet villkor  för lambda=-20: 0 < h |-20| < 2
h= 0.01;
t0 = 0;           % begynnelsetid
ts = 1.5;            % slut-tid
N = (ts - t0)/h   %antal punkter
  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
y(1) = 1;  % begynnelsevarden

for k = 1:N
   y(k+1) = y(k) + h*func_stiff(t(k),y(k));
            t(k+1) = t(k) + h;
 end
```


Bakåt Eulers metod för $y' = -20y,\ y(0) = 1$

```
h= 0.05;
t0 = 0;           % begynnelsetid
ts = 1.5;            % slut-tid
N = (ts - t0)/h   %antal punkter
  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
y=0;
y(1) = 1;  % begynnelsevarden

for k = 1:N

 y(k+1) = y(k)/(1.0 + 20.0*h);
            t(k+1) = t(k) + h;
 end
```

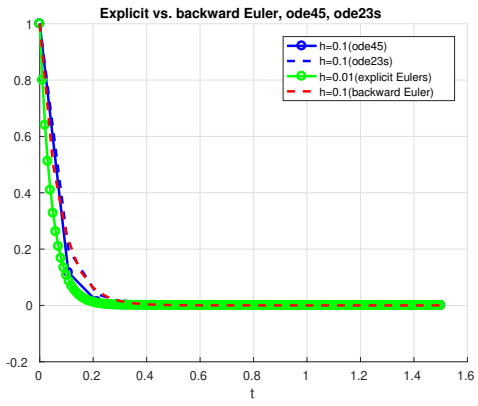
ODE (Styva problem)

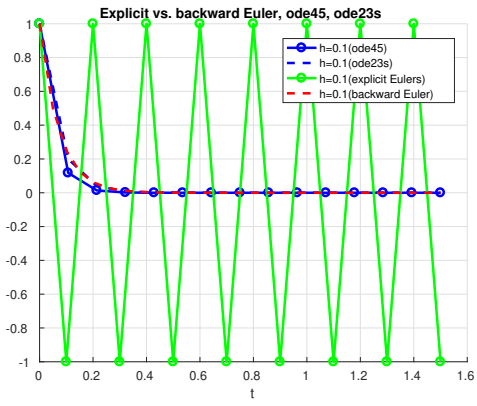
Exempel: vi löser $y' = \lambda y,\ y(0) = 1,\ t \in [0, 5],\ \lambda = -20000$ med Matlabs ode23 samt ode23s (s för stiff).

ode23 kräver 119476 funktionsberäkningar och ger ett maxfel av $1.2 \cdot 10^{-6}$.

ode23s kräver 230 funktionsberäkningar och ger ett maxfel av $3.4 \cdot 10^{-13}$.

ODE (Styva problem). Exempel: $y' = -20y, y(0) = 1$.





Taylor series metoder

Vi har redan sett att vi kan få explicit Euler's metod via Taylor's utveckling. Om vi kan 4 termer i Taylorsutveklingen

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y^{(3)}(t) + ...$$

då vi ska få andra ordningens metod:

$$y_{k+1} = y_k + h_k y'_k + \frac{h_k^2}{2}y''_k.$$

Eftersom den metod kräver beräkning en av y''_k, då vi användr följande formula för detta:

$$y'' = f_t(t,y)t' + f_y(t,y)y' = f_t(t,y) + f_y(t,y)f(t,y),$$

var $f(t,y)$ är högerleden i ODE

$$y'(t) = f(t,y).$$

Taylor series metoder

Andra ordningens enstegsmetod är nu:

$$y_{k+1} = y_k + h_k y'_k + \frac{h_k^2}{2}y''_k. \tag{58}$$

med

$$y'_k(t) = f(t_k, y_k). \tag{59}$$

för approximation av första derivata $y'_k(t)$, och med

$$y''_k = f_t(t_k, y_k) + f_y(t_k, y_k)f(t_k, y_k), \tag{60}$$

för approximation av andra derivata $y''_k(t)$.

Taylor series metoder

Exempel

Vi vill visa hur kan vi använda andra ordningens enstegsmetod (58) för att lösa problem

$$y'(t) = f(t,y) = -2ty^2, \quad y(0) = 1. \tag{61}$$

Man kan kolla att exakt lösning för problemet är $y(t) = \frac{1}{1+t^2}$.
För att använda (58), vi ska beräkna y'' i (60):

$$f_t = (-2ty^2)'_t = -2y^2, \quad f_y = (-2ty^2)'_y = -4ty.$$

Därför (60) skrivs som

$$y''_k = f_t(t_k,y_k)+f_y(t_k,y_k)f(t_k,y_k) = -2y_k^2+4t_ky_k\cdot 2t_ky_k^2 = 2y_k^2(-1+4t_k^2y_k).$$

Då flerstegsmetod (58) för problemet (61) skrivs som:

$$y_{k+1} = y_k + h_ky'_k + \frac{h_k^2}{2}y''_k = y_k - 2h_kt_ky_k^2 + \frac{h_k^2}{2}(2y_k^2(-1+4t_k^2y_k)).$$

ODE: Runge-Kutta metoder

Runge-Kutta metoder är enstegsmetoder, som använder finite difference approximationer av funktioner $f(t,y(t))$ i punkter, som ligger på $[t_k, t_{k+1}]$.
För att derivera Runge-Kutta metoder, vi ska använda

$$y'' = f_t(t,y)t' + f_y(t,y)y' = f_t(t,y) + f_y(t,y)f(t,y), \tag{62}$$

var $f(t,y)$ är högerleden i ODE

$$y'(t) = f(t,y).$$

Nästa steg är att vi ska approximera höger leden i (62) med hjälp av Taylors formel i två variabel:

$$f(t+h, y+hf) = f + hf_t + hf_yf + O(h^2) = f + h(f_t + f_yf) + O(h^2) \tag{63}$$

så at vi kan få

$$f_t + f_yf = \frac{f(t+h,y+hf) - f(t,y)}{h} + O(h). \tag{64}$$

ODE: Runge-Kutta metoder

I så fall andra ordningens Taylors metod (58) ska skrivas som

$$y_{k+1} = y_k + h_k y'_k + \frac{h_k^2}{2} y''_k = y_k + h_k f(t_k, y_k) + \frac{h_k^2}{2} \frac{f(t_k + h_k, y_k + h_k f(t_k, y_k)) - f(t_k, y_k)}{h_k}$$
$$= y_k + \frac{h_k}{2}(f(t_k, y_k) + f(t_k + h_k, y_k + h_k f(t_k, y_k))). \tag{65}$$

som är också känd som Heuns metod, som vi redan har diskuterat och deriverat på annat sätt. Heuns metod är enstegsmetod som kan skrivas också

$$\begin{aligned} \tilde{y}_k &= y_k + h_k f(t_k, y_k), \\ y_{k+1} &= y_k + \frac{h_k}{2}(f(t_k, y_k) + f(t_k + h_k, \tilde{y}_k)). \end{aligned} \tag{66}$$

ODE: Runge-Kutta metoder

Mest känd är följande klassisk Runge-Kutta metod av 4 ordningen:

$$\begin{aligned} y_{k+1} &= y_k + \frac{h_k}{6}(k_1 + k_2 + k_3 + k_4), \\ k_1 &= f(t_k, y_k), \\ k_2 &= f(t_k + h_k/2, y_k + (h_k/2)k_1), \\ k_3 &= f(t_k + h_k/2, y_k + (h_k/2)k_2), \\ k_4 &= f(t_k + h_k, y_k + h_k k_3). \end{aligned} \tag{67}$$

Runge-Kutta metod av 4 ordningen är Simpson's metod när $f = f(t)$:

$$\begin{aligned} y_{k+1} &= y_k + \frac{h_k}{6}(k_1 + k_2 + k_3 + k_4), \\ k_1 &= f(t_k), \\ k_2 &= f(t_k + h_k/2), \\ k_3 &= f(t_k + h_k/2), \\ k_4 &= f(t_k + h_k). \end{aligned} \tag{68}$$

ODE: Runge-Kutta metoder

Fördel med Runge-Kutta metoder är att man behöver inte känna till lösningen på tidigare iterationer $k-1, \ldots$ för att beräkna nästa iteration, $k+1$. De tillåter ändra h under beräkningstiden.
Vi har tittat på explicita Runge-Kutta metoder. De duger inte för lösning av styva problem, eller icke-stabila problem. För lösning av styva problem används implicita Runge-Kutta metoder, som har gemensamt form

$$\begin{aligned} y_{k+1} &= y_k + h\sum_{i=1}^{s} b_i k_i, \\ k_i &= f(t_k + c_i h, y_k + h\sum_{j=1}^{s} a_{ij} k_j), \quad i = 1, \ldots, s. \end{aligned} \tag{69}$$

Enklaste implicit Runge-Kutta metod följer from (69) med för $s = 1, b_1 = 1, c_1 = 1, a_{11} = 1$

$$y_{k+1} = y_k + hk_1, \quad k_1 = f(t_k + h, y_k + hk_1), \tag{70}$$

som kan skrivas om för att få implicit Euler's metod:

$$y_{k+1} = y_k + hf(t_k + h, y_k + h) = y_k + hf(t_{k+1}, y_{k+1}). \tag{71}$$

ODE: multistep-metoder

Linjära multistep-metoder har formen

$$y_{k+1} = \sum_{i=1}^{m} \alpha_i y_{k+1-i} + h\sum_{i=0}^{m} \beta_i f(t_{k+1-i}, y_{k+1-i}). \tag{72}$$

Om $\beta_0 = 0$ i (72), då metoden är explicit, annars metoden är implicit.
Parametrar α_i, β_i är bestämda genom polynominterpolation.
Vi kan visa hur kan man derivera explicit två-steg metod på formen

$$y_{k+1} = \alpha_1 y_k + h(\beta_1 y'_k + \beta_2 y'_{k-1}) \tag{73}$$

Här parametrar $\alpha_1, \beta_1, \beta_2$ ska determineras. Med hjälp av metoden för underbestämda koefficienter vi kommer att tvinga formeln (77) att vara exakt för de tre första monomialerna.

ODE: multistep-metoder

Vi ska använda följande 3 villkor:

- $$y(t) = 1 \to y'(t) = 0 \to 1 = \alpha_1 \cdot 1 + h(\beta_1 \cdot 0 + \beta_2 \cdot 0). \quad (74)$$
- $$y(t) = t \to y'(t) = 1 \to t_{k+1} = \alpha_1 \cdot t_k + h(\beta_1 \cdot 1 + \beta_2 \cdot 1). \quad (75)$$
- $$y(t) = t^2 \to y'(t) = 2t \to t_{k+1}^2 = \alpha_1 \cdot t_k^2 + h(\beta_1 \cdot 2t_k + \beta_2 \cdot 2t_{k-1}). \quad (76)$$

Nu väljer vi $t_{k-1} = 0, h = 1$ och i så fall $t_k = 1, t_{k+1} = 2$ och löser följande system av linjära ekvationer för $\alpha_1, \beta_1, \beta_2$:

$$\begin{aligned} \alpha_1 &= 1, \\ \alpha_1 + \beta_1 + \beta_2 &= 2, \\ \alpha_1 + 2\beta_1 &= 4. \end{aligned}$$

ODE: multistep-metoder

Från lösningen av system

$$\begin{aligned} \alpha_1 &= 1, \\ \alpha_1 + \beta_1 + \beta_2 &= 2, \\ \alpha_1 + 2\beta_1 &= 4. \end{aligned}$$

vi hittar $\alpha_1 = 1, \beta_1 = 3/2, \beta_2 = -1/2$ och metoden

$$y_{k+1} = \alpha_1 y_k + h(\beta_1 y'_k + \beta_2 y'_{k-1})$$

kan skrivas som

$$y_{k+1} = y_k + \frac{h}{2}(3y'_k - y'_{k-1}), \quad (77)$$

och genom konstruktion är av ordningen två.

ODE (Några andra ODE-problem)

- Tvåpunkts randvärdesproblem:

  $$y'' = f(t, y, y'),\ \alpha_1 y(a) + \beta_1 y'(a) = \gamma_1,\ \alpha_2 y(b) + \beta_2 y'(b) = \gamma_2$$

- 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

  $$y'' + \frac{\lambda y}{\sqrt{y^2 + t^2}} = 0, \text{ samt randvillkor}$$

ODE (Några andra ODE-problem)

- Tidsfördröjningsproblemet (delay equations)

  $$y'(t) = y(t) - y(t - T) + \ldots$$

  Inkubationstid; ändlig utbredningshastighet...
- Differentialalgebraiska problem; differentialekvation med algebraiska "bivillkor". Specialfall, implicita problem: $g(t, y)y' = f(t, y)$.

Övning

Skriv om följande system ekvationer som ett första ordningens system:

$$\begin{cases} u'' = 2u'v' + v^2 + t \\ v''' = u + v + v''u \end{cases}, \quad \begin{cases} u(0) = 1,\ u'(0) = -1 \\ v(0) = 2,\ v'(0) = 3,\ v''(0) = -4 \end{cases}$$

Övning

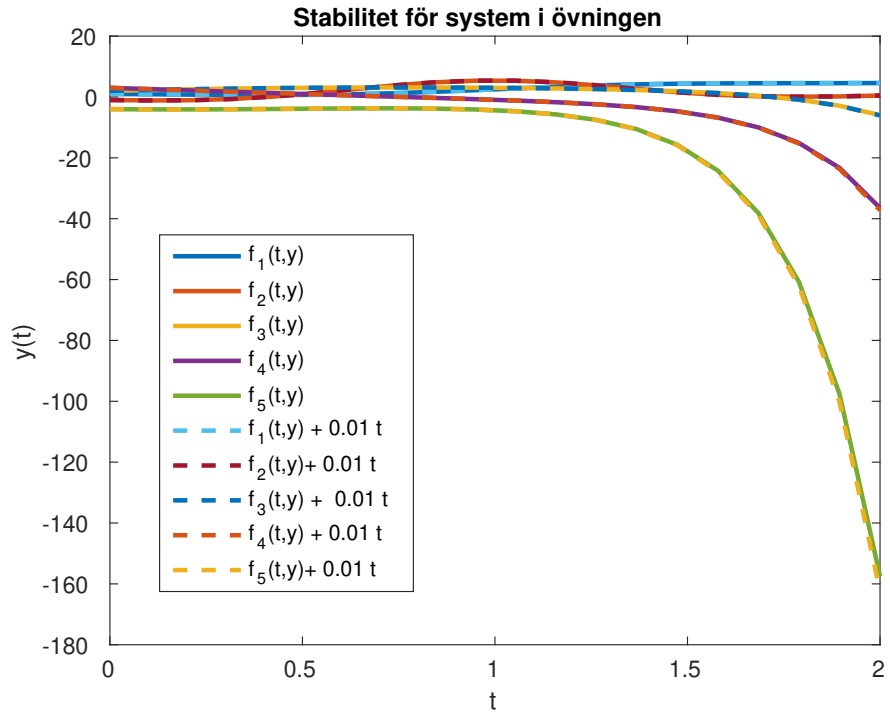
Skriv om följande system ekvationer som ett första ordningens system:

$$\begin{cases} u'' = 2u'v' + v^2 + t \\ v''' = u + v + v''u \end{cases}, \quad \begin{cases} u(0) = 1,\ u'(0) = -1 \\ v(0) = 2,\ v'(0) = 3,\ v''(0) = -4 \end{cases}$$

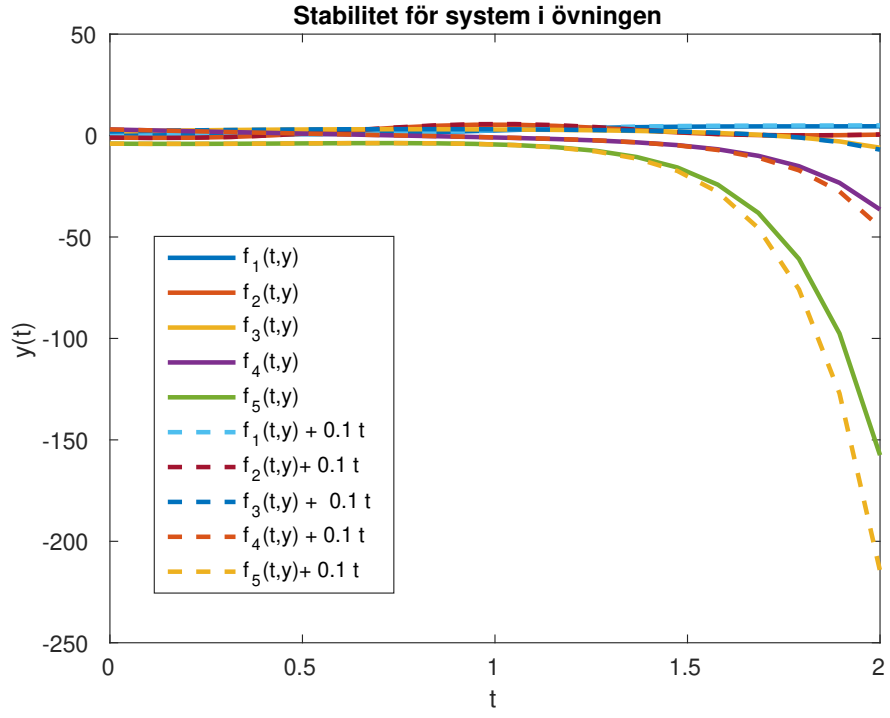
Svar:
Inför $y_1 = u$, $y_2 = u' = y_1'$, $y_3 = v$, $y_4 = v' = y_3'$ och $y_5 = v'' = y_4'$. Systemet blir

$$\begin{cases} y_1' = y_2 \\ y_2' = 2y_2y_4 + y_3^2 + t \\ y_3' = y_4 \\ y_4' = y_5 \\ y_5' = y_1 + y_3 + y_5y_1 \end{cases}, \quad \begin{cases} y_1(0) = 1 \\ y_2(0) = -1 \\ y_3(0) = 2 \\ y_4(0) = 3 \\ y_5(0) = -4 \end{cases}$$

Stabilitet



Stabilitet



Övning

Skriv om följande ekvationer som första ordningens system:

- a) $y'' = t + y + y', y(0) = 1, y'(0) = -1$
- b) $y''' = y'' + ty, y(0) = 1, y'(0) = -1, y''(0) = 3,$
- c) $y''' = y'' - 2y' + y - t + 1, y(0) = 1, y'(0) = -1, y''(0) = 3.$

Övning

Skriv om följande ekvationer som första ordningens system:

- a) $y'' = t + y + y', y(0) = 1, y'(0) = -1$
  Svar:
  Sätt $u_1 = y, u_2 = u_1' = y'$. Vi får systemet:
  $$\begin{cases} u_1' = u_2, \\ u_2' = t + u_1 + u_2, \\ u_1(0) = 1, u_2(0) = -1. \end{cases}$$
- b) $y''' = y'' + ty, y(0) = 1, y'(0) = -1, y''(0) = 3$, Svar:
  Sätt $y = u_1, y' = u_1' = u_2, y'' = u_2' = u_3$. Vi får systemet:
  $$\begin{cases} u_1' = u_2, \\ u_2' = u_3, \\ u_3' = u_3 + tu_1 \\ u_1(0) = 1, u_2(0) = -1, u_3(0) = 3. \end{cases}$$

Övning

Skriv om följande ekvation som första ordningens system:

- c) $y''' = y'' - 2y' + y - t + 1, y(0) = 1, y'(0) = -1, y''(0) = 3.$
  Svar: sätt $y = u_1, y' = u_1' = u_2, y'' = u_2' = u_3$. Vi får

  systemet: $\begin{cases} u_1' = u_2, \\ u_2' = u_3, \\ u_3' = u_3 - 2u_2 + u_1 - t + 1, \\ u_1(0) = 1, u_2(0) = -1, u_3(0) = 3. \end{cases}$

Övning

Sätt upp bakåt-Euler för problemet

$$y' = -y^2, y(0) = 1.$$

Formulera den ickelinjära ekvation som uppkommer för att beräkna y_{k+1} samt ställ upp Newtons metod för denna ekvation.

Övning

Sätt upp bakåt-Euler för problemet

$$y' = -y^2, y(0) = 1.$$

Formulera den ickelinjära ekvation som uppkommer för att beräkna y_{k+1} samt ställ upp Newtons metod för denna ekvation.

Övning

Svar:
Bakåt Euler:

$$\frac{y^{k+1} - y^k}{h} = -(y^{k+1})^2$$

eller

$$y^{k+1} + h(y^{k+1})^2 = y^k.$$

För att lösa den ekvation vi använder Newtons metod: vi inför ny variabel $z = y^{k+1}$ och skriver om bakåt Eulers metod som:

$$z + hz^2 = y^k.$$

Newtons' metod blir:

$$z^{j+1} = z^j - \frac{h(z^j)^2 + z^j - y^k}{2hz^j + 1}$$

Här, j är iteration i Newton's metod.

Övning

Vilka lösningar har följande problem?

$$y' = 3/2y^{1/3}, y(0) = 0$$

Övning

Vilka lösningar har följande problem?

$$y' = 3/2y^{1/3}, y(0) = 0$$

Svar: Ekvationen är separabel. Löser vi på den på ett av de vanliga sätten, får vi:

$$\int \frac{dy}{y^{1/3}} = 3/2 \int dt$$

och

$$3/2y^{2/3} = 3/2t + const,$$

eller $y(t) = (t + 2/3 \cdot const)^{3/2}$.
Begynnelsevärdet ger $const = 0$ och då $y(t) = t^{3/2}$. Lösningen är inte entydig eftersom även $y(t) = 0$ är en lösning.

Övning

Eulers metod kan härledas på följande sätt:

$$y(t+h) = y(t)+hy'(t)+\frac{h^2}{2}y''(t)+\ldots \approx y(t)+hy'(t) = y(t)+hf(t,y(t)).$$

vilket ger framåt Eulers metoden

$$y_{k+1} = y_k + hf(t_k, y_k).$$

Härled en högre ordningens metod genom att ta med nästa term i Taylorutvecklingen.

Övning

Approximera

$$y''(t) \approx \frac{y'(t) - y'(t-h)}{h}$$

så att

$$\begin{aligned} y(t+h) &\approx y(t) + hy'(t) + \frac{h^2}{2}y''(t) && (78)\\ &= y(t) + hy'(t) + \frac{h^2}{2}\frac{y'(t) - y'(t-h)}{h} && (79)\\ &= y(t) + hf(t,y) + \frac{h}{2}(f(t,y) - f(t-h, y-h)) && (80)\\ &= y(t) + \frac{h}{2}(3f(t,y) - f(t-h, y-h)). && (81) \end{aligned}$$

Detta leder till metoden:

$$y_{k+1} = y_k + \frac{h}{2}(3f(t_k, y_k) - f(t_{k-1}, y_{k-1}))$$

vilket var den andra ordningens flerstegsmetod vi såg under föreläsningen.

Övning

En annan tänkbar approximation är t.ex.

$$y''(t) \approx \frac{y'(t+h) - y'(t)}{h}$$

så att

$$\begin{aligned} y(t+h) &\approx y(t) + hy'(t) + \frac{h^2}{2} y''(t) && (82)\\ &= y(t) + hy'(t) + \frac{h^2}{2} \frac{y'(t+h) - y'(t)}{h} && (83)\\ &= y(t) + hf(t,y) + \frac{h}{2}(f(t+h, y+h) - f(t,y)) && (84)\\ &= y(t) + \frac{h}{2}(f(t+h, y+h) + f(t,y)). && (85) \end{aligned}$$

Detta leder till implicita flerstegsmetoden:

$$y_{k+1} = y_k + \frac{h}{2}(f(t_{k+1}, y_{k+1}) + f(t_k, y_k)).$$