Numeriska metoder för M, V och Z

Jacques Huitfeldt

Matematik
Chalmers Tekniska Högskola
2005

1 Introduktion.

Det här kompendiet handlar om hur man löser några olika typer av matematiska problem, som ofta uppstår i teknik och naturvetenskap, med hjälp av dator. Inom ämnesområdet numerisk analys utvecklar och analyserar man metoder för att lösa sådana problem. Det är ett specialiserat område som det krävs ganska god kunskap i matematik för att man skall kunna tränga in i.

För att få relativt lättillgängliga beräkningsverktyg har man samlat datorprogram, där man implementerat dessa numeriska metoder, i paket för olika beräkningsuppgifter, s.k. matematisk programvara.

Vi kommer använda matematisk programvara för de olika klasser av matematiska problem som vi skall behandla. Den typiske användaren av matematisk programvara är inte expert på numerisk analys utan vill bara kunna utföra beräkningar för att få svar på frågor inom sitt eget ämnesområde. Man måste ändå ha en del kunskap om de numeriska metoder som används i den matematiska programvaran för att kunna använda den på ett tillfredställande sätt. Därför kommer vi också att presentera en del av de numeriska metoder som programvaran bygger på.

Resten av det här avsnittet behandlar några grundläggande begrepp inom numerisk analys, något om hur datorer räknar och slutligen något om matematisk programvara.

1.1 Vägen till beräkningsresultat.

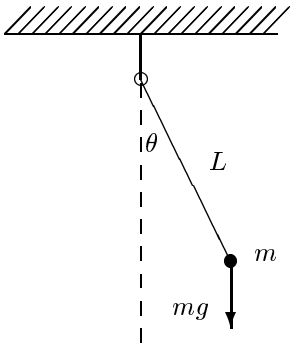
Givet en matematisk modell för ett experiment eller en konstruktion så leder de flesta frågeställningar fram till att man vill lösa ett matematiskt problem av något slag.

Med hjälp av *matematisk analys* kan man komma fram till resultat om existens och egenskaper hos lösningen till ett matematiskt problem. Om det matematiska problemet kommer från en matematisk modell inom t.ex. mekaniken så är det rimligt att vänta sig att det existerar en entydigt bestämd lösning till problemet som beror kontinuerligt på indata, annars vore modellen inte användbar. Ett matematiskt problem som har dessa egenskaper kallas *rättställt*.

Ibland kan man också skriva upp en formel för lösningen, dvs. ge den på sluten form. Vanligen måste man, för att man skall kunna få en lösning på sluten form, förenkla det matematiska problemet så kraftigt att det kanske inte alls beskriver verkligheten längre.

Med *numerisk analys* försöker man istället hitta en approximativ lösning direkt till det *ursprungliga* problemet.

Exempel 1.1. Låt oss betrakta en matematisk pendel. En masspunkt med massan m hänger i en viktlös smal stav av längden L.



Med beteckningarna i figuren och Newtons andra lag får vi rörelseekvationen

$$\theta'' + \frac{g}{L}\sin(\theta) = 0.$$

Man kan visa att det för givna begynnelsevärden på hastighet och läge finns en entydigt bestämd lösning som beror kontinuerligt på indata. Problemet är alltså rättställt. Det går dock inte att skriva upp en lösning på sluten form. Däremot kan man för små utslagsvinklar approximera $\sin(\theta)$ med θ och får då den linjära ekvationen

$$\theta'' + \frac{g}{L}\theta = 0$$

med lösningen

$$\theta(t) = A\sin\left(\sqrt{\frac{g}{L}}t\right) + B\cos\left(\sqrt{\frac{g}{L}}t\right),$$

där A och B är konstanter som bestäms entydigt av begynnelsevärdena.

För större utslagsvinklar ger den förenklade ekvationen helt andra lösningar än den ursprungliga ekvationen och blir därmed helt meningslös.

Däremot kan man med numeriska metoder, som vi skall ta fram senare i kursen, hitta en godtyckligt bra approximation av lösningen till den ursprungliga ekvationen. ■

Numeriskt problem och algoritm.

Ett matematiskt problem innehåller ofta oändligt mycket information. För att vi skall kunna behandla det med datorn måste vi approximera det matematiska problemet med ett *numeriskt problem* med bara ändligt mycket information som kan representeras i datorn, dvs. ett ändligt antal tal med ändligt antal siffror. Vi gör då en s.k. *diskretisering* av problemet. Exempel på diskretisering är då vi ersätter en oändlig serie med en partialsumma eller en differentialekvation med en differensekvation. Vid en sådan diskretisering uppstår ett s.k. *diskretiseringsfel*.

Exempel 1.2. Om vi inför framåtdifferenskvoten

$$D_+(h)y(x) = \frac{y(x+h) - y(x)}{h}$$

och centraldifferenskvoten

$$D_+D_-(h)y(x) = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2}$$

så är de approximationer av derivatorna $y'(x)$ respektive $y''(x)$. Senare i kursen kommer vi visa att [1]

$$y'(x) = D_+(h)y(x) + \mathcal{O}(h)$$

och

$$y''(x) = D_+D_-(h)y(x) + \mathcal{O}(h^2)$$

Här har vi mått på hur noggranna approximationerna är, dvs. hur stora diskretiseringsfelen är. [2]

Vi kommer använda differensapproximationer bland annat för att approximera lösningar till differentialekvationer (som i exempel 1.1). Hur noggrann denna approximation blir, beror bl.a. av diskretiseringsfelen i approximationen av derivatorna. ■

Med en *numerisk algoritm* för ett numeriskt problem avses en fullständig beskrivning av en följd av väldefinierade operationer, genom vilken varje tillåten uppsättning indata transformeras till en uppsättning utdata. Algoritmen brukar *implementeras* i form av ett *datorprogram* av ändlig längd och som är exekverbart på ändlig tid.

Förhoppningsvis är utdata från den numeriska algoritmen en god approximation av lösningen till det matematiska problemet. Att undersöka om så är fallet är en uppgift för den numeriska analysen.

Stabilitet.

Ett problem (matematiskt eller numeriskt) kallas *stabilt* eller *välkonditionerat* om små förändringar i indata till problemet leder till endast små förändringar i lösningen till problemet. Störningar i indata förekommer ofta. Indata kan vara mätdata av begränsad noggrannhet eller reella tal som måste approximeras med ett fixt antal siffror för att kunna lagras i datorn.

En algoritm kallas stabil eller välkonditionerad om den ger exakt lösning till ett problem som är nära det ursprungliga problemet, dvs. exakt lösning till ett stört problem. Genom s.k. bakåtfelanalys försöker man ge begränsningar på storleken av dessa störningar.

I figuren nedan till vänster ser vi hur en stabil algoritm beter sig för ett stabilt problem. Istället för lösningen x till problemet P så ger algoritmen lösningen x^* till problemet P^*. Eftersom algoritmen är stabil så ligger P^* nära P. Nu är problemet stabilt så x^* ligger nära x. Alltså har vi fått en bra approximation av den exakta lösningen till det givna problemet.

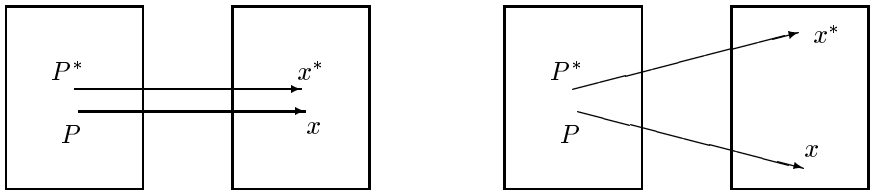
[1] I matematikboken beskrivs stort ordo på sid. 356.

[2] Nu skall vi se varför det blir så. Taylorutveckling av y runt x ger

$$y(x+h) = y(x) + y'(x)h + \frac{y''(x)}{2}h^2 + \frac{y^{(3)}(x)}{6}h^3 + \frac{y^{(4)}(x)}{24}h^4 + \cdots$$

$$y(x-h) = y(x) - y'(x)h + \frac{y''(x)}{2}h^2 - \frac{y^{(3)}(x)}{6}h^3 + \frac{y^{(4)}(x)}{24}h^4 - \cdots$$

Subtraktion resp. addition ger resultaten.



Till höger i figuren ser vi hur en stabil algoritm beter sig för ett instabilt problem. Istället för lösningen x till problemet P så ger algoritmen lösningen x^* till problemet P^*. Eftersom algoritmen är stabil så ligger P^* nära P. Men nu är problemet instabilt så x^* ligger inte nära x. Alltså har vi inte lyckats få en bra approximation av den sökta lösningen. Algoritmen kan givetvis inte trolla bort instabiliteten i problemet.

Eftersom numeriska algoritmer kan ha olika grad av stabilitet är det viktigt att välja algoritm med tanke på detta. Instabila algoritmer kommer vi överhuvudtaget inte använda oss av.

1.2 Noggrannhetsanalys.

Låt x beteckna ett exakt värde och $\tilde{x}$ en approximation av x (närmevärde till x). För det absoluta felet i $\tilde{x}$ inför vi beteckningen

$$e_x = \tilde{x} - x$$

Även δx kommer att användas för att beteckna det absoluta felet, speciellt då vi inte direkt ser det som ett fel utan snarare som en osäkerhet eller variation.

Oftast känner vi inte x utan bara $\tilde{x}$ och får då ge en gräns för storleken på det absoluta felet. Denna gräns betecknar vi med E_x, dvs.

$$|e_x| \leq E_x$$

Har man en approximation $\tilde{x}$ och en gräns för absoluta felet E_x så vet vi om x att

$$\tilde{x} - E_x \leq x \leq \tilde{x} + E_x$$

Detta brukar också skrivas $x = \tilde{x} \pm E_x$.

Det är rimligt att sätta felet i relation till hur stor en kvantitet är. Vi inför följande beteckning för det relativa felet i $\tilde{x}$

$$r_x = \frac{e_x}{x} = \frac{\tilde{x} - x}{x} \quad (\text{om } x \neq 0)$$

Med denna beteckning gäller

$$\tilde{x} = x(1 + r_x)$$

som är bra att utnyttja i vissa sammanhang. För gränsen på storleken av relativa felet använder vi beteckningen R_x, dvs.

$$|r_x| \leq R_x.$$

En approximation $\tilde{x}$ av x sägs ha t korrekta decimaler om

$$|e_x| \leq \frac{1}{2} \cdot 10^{-t}.$$

Om $\tilde{x}$ har k inledande nollor så säger vi att $\tilde{x}$ har $s = t - k$ signifikanta siffror. Detta gäller för $\tilde{x}$ med belopp mindre än 1. Om vi t.ex. har talet 7100 ± 50 så säger vi att approximationen har 2 signifikanta siffror.

Felfortplantning.

Vi känner en approximation $\tilde{x}$ av x och vill beräkna $f(x)$. Det vi kan göra är att beräkna $f(\tilde{x})$. Vad gäller då för

$$e_f = f(\tilde{x}) - f(x)$$

Vi tar och Taylorutvecklar f runt $\tilde{x}$

$$f(x) = f(\tilde{x} - (\tilde{x} - x)) = f(\tilde{x} - e_x) = f(\tilde{x}) - f'(\tilde{x})e_x + \frac{f''(\tilde{x})}{2}e_x^2 - \cdots$$

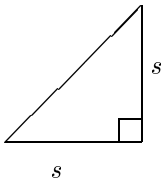
som ger

$$e_f = f(\tilde{x}) - f(x) = f'(\tilde{x})e_x - \frac{f''(\tilde{x})}{2}e_x^2 + \cdots$$

Vi är intresserade av storleksordningen på e_f, så om $f'(\tilde{x}) \neq 0$ och e_x litet kan vi bortse ifrån högre potenser av e_x och får den s.k. felfortplantningsformeln

$$e_f \approx f'(\tilde{x})e_x$$

Exempel 1.3. Betrakta triangeln



där vi inte känner sidan s exakt, utan $s = \tilde{s} \pm E_s$. Triangelns area är

$$A(s) = \frac{1}{2}s^2$$

och vi vill veta med vilken noggrannhet den kan bestämmas. Vi har $A'(s) = s$ så vi får att $e_A \approx A'(\tilde{s})e_s = \tilde{s}e_s$ ■

1.3 Talrepresentation i datorn och flyttalsaritmetik.

Om man vill kunna hantera både (till belopp) mycket små och mycket stora tal i datorn så är det inte lämpligt att arbeta med ett fixt antal decimaler, utan man arbetar istället med ett fixt antal siffror, s.k. flyttal.

Tal representeras då på formen

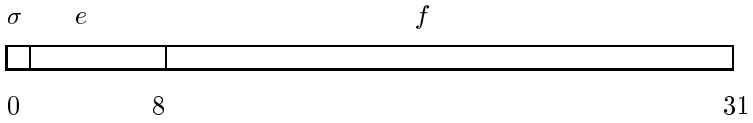
$$x = \sigma m \beta^e$$

med $1 \leq m < \beta$ (normalisering som gör representationen entydig), $\sigma = \pm 1$ och $L \leq e \leq U$ ett heltal. Här kallas β för basen (vanligen 2 eller 16), m för mantissan och e för exponenten. Om vi låter t vara antalet siffror i bråkdelen av m så kan ett allmänt flyttalssystem karakteriseras genom (β, t, L, U).

Om resultatet vid en beräkning ger ett flyttal med $e > U$ har man ett s.k. överspill (overflow). Motsvarande då $e < L$ kallas underspill (underflow).

Beräkningar kan givetvis inte göras exakta utan avrundningsfel uppstår eftersom datorn räknar med ett ändligt fixt antal siffror. Förlust av relativ noggrannhet p.g.a. kancellation av siffror (vid subtraktion av två nästan lika tal) och utskiftning (vid addition av tal med väsentligt olika storleksordning) är svårigheter som man ofta helt eller delvis kan ta sig förbi genom omskrivningar.

En standard för binär flyttalsaritmetik antogs 1985 av den amerikanska organisationen IEEE. Standarden definierar fyra olika format för flyttal, grundformat och utökat format bägge i enkel och dubbel precision. Grundformatet i enkel precision använder 1 ord = 32 bitar med 1 bit för tecken, 8 bitar för exponenten och 23 bitar för bråkdelen.



Det innebär att $L = -126$ och $U = 127$. För värdet v av ett lagrat tal gäller:

(1). Om $e = 255$ och $f \neq 0$ så är $v = NaN$ (Not-a-Number).

(2). Om $e = 255$ och $f = 0$ så är $v = (-1)^\sigma \infty$.

(3). Om $0 < e < 255$ så är $v = (-1)^\sigma (1.f) 2^{e-127}$.

(4). Om $e = 0$ och $f \neq 0$ så är $v = (-1)^\sigma (0.f) 2^{-126}$ (gradvis underspill).

(5). Om $e = 0$ och $f = 0$ så är $v = (-1)^\sigma 0$.

Normalfallet är (3). Normaliseringen gör att heltalssiffran alltid är en 1:a som därför inte behöver lagras. Vi ser att exponenten är lagrad förskjuten.

Fall (1) NaN kan t.ex. uppstå vid en operation $0/0$ eller kvadratroten ur ett negativt tal. Överspill representeras av (2). Vanligen sätts resultatet till noll vid underspill, i standarden föreskrivs dock gradvis underspill, fall (4).

Grundformatet i dubbel precision använder 2 ord = 64 bitar med 1 bit för tecken, 11 bitar för exponenten och 52 bitar för bråkdelen. Detta ger $L = -1022$ och $U = 1023$. Flyttalssystemen vid IEEE-standard är alltså (2, 23, -126, 127) vid enkel precision och (2, 52, -1022, 1023) vid dubbel precision.

Standarden föreskriver ett minsta antal bitar för det utökade formatet. Vanligen har man använt 80 bitar för både enkel och dubbel precision. De 80 bitarna räcker precis till för dubbel precision.

För flyttalsrepresentationen av x, ofta betecknad $fl(x)$, kan det relativa avrundningsfelet uppskattas med

$$\frac{|x - fl(x)|}{|x|} \leq \mu,$$

där $\mu = \frac{1}{2}\beta^{-t}$ kallas avrundningsenheten (t är antal siffror i bråkdelen). Denna relation kan också skrivas

$$fl(x) = x(1 + r) \quad \text{med} \quad |r| \leq \mu$$

Av detta ser vi att flyttalen ligger tätare nära det till belopp minsta talet än nära det till belopp största talet.

Implementeringar av standarden skall innehålla bl.a. de fyra räknesätten, kvadratrotsfunktion och konvertering binärt-decimalt. När alla operander är normaliserade, skall en operation utföras så att resultatet är lika med det avrundade resultatet av samma operation utförd med oändlig precision, dvs.

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + r) \quad |r| \leq \mu$$

Standarden specificerar att avrundning skall ske enligt de regler vi är vana vid.

Vid s.k. framåtanalys följer man upp alla beräkningar och avrundningar för att få en uppskattning av hur mycket det beräknade värdet skiljer sig från det exakta.

Tidigare nämnde vi den s.k. bakåtanalysen, som kan vara enklare att använda vid mer komplicerade algoritmer. Här uppfattades den erhållna approximationen som exakt lösning till ett stört problem, dvs. med avvikelse i indata, och alla beräknings- och avrundningsfel relateras till fel i indata.

1.4 Övningar.

1. Låt $x = 1.00 \pm 0.005$ och $y = 2.00 \pm 0.01$. Bestäm gränser för absoluta och relativa felen i produkten xy respektive kvoten $\frac{x}{y}$.

2. Linsformeln
$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b}$$
ger ett samband mellan föremålsavståndet a, bildavståndet b och brännvidden f. Vad gäller för f om vi har följande osäkerhet i indata, $a = 200 \pm 1$ och $b = 100 \pm 0.5$?

3. Antag att vi står framför ett torn och att avståndet till tornet är 30 ± 1 m. Vidare är vinkeln mellan markplanet och siktlinjen upp till tornets topp $19 \pm 1\,^\circ$. Hur högt är tornet?

4. Ge en felgräns för felet i $y = \cos(x)$ då $x = 0 \pm 0.001$. Tänk dig för innan du använder felfortplantningsformeln.

5. Vi vill beräkna $f = x^2 - y^2$ (x och y lagrade i datorn) med hjälp av flyttalsaritmetik och väljer mellan algoritmerna

   $Algoritm\ 1 : t_1 = x^2,\ t_2 = y^2,\ f_1 = t_1 - t_2$

   $Algoritm\ 2 : s_1 = x + y,\ s_2 = x - y,\ f_2 = s_1 s_2$

(a) Ange det absoluta felet i det beräknade värdet för de båda fallen.

(b) Kan man garantera att det relativa felet blir litet för någon av algoritmerna?

1.5 Lösningar.

1. Vi har $x = 1.00 \pm 0.005$ och $y = 2.00 \pm 0.01$. För multiplikationen får vi

$$xy \leq (\tilde{x} + E_x)(\tilde{y} + E_y) = 1.005 \cdot 2.01 = 2.02005$$

$$xy \geq (\tilde{x} - E_x)(\tilde{y} - E_y) = 0.995 \cdot 1.99 = 1.98005$$

dvs. $1.98005 \leq xy \leq 2.02005$. Som approximation av $z = xy$ tar vi

$$\tilde{z} = \frac{1.98005 + 2.02005}{2} = 2.00005$$

med

$$E_z = \frac{2.02005 - 1.98005}{2} = 0.02$$

och

$$R_z \approx 0.01, \text{ eftersom } |r_z| = \frac{|e_z|}{|z|} \leq \frac{E_z}{1.98005} = 0.01010075\cdots$$

För divisionen får vi

$$\frac{x}{y} \leq \frac{\tilde{x} + E_x}{\tilde{y} - E_y} = \frac{1.005}{1.99} = 0.5050251\cdots < 0.505026$$

$$\frac{x}{y} \geq \frac{\tilde{x} - E_x}{\tilde{y} + E_y} = \frac{0.995}{2.01} = 0.4950248\cdots > 0.495024$$

dvs. $0.495024 < \frac{x}{y} < 0.505026$, och som approximation av $z = \frac{x}{y}$ tar vi

$$\tilde{z} = \frac{0.495024 + 0.505026}{2} = 0.500025$$

med

$$E_z = \frac{0.505026 - 0.495024}{2} = 0.005001 \approx 0.005$$

och

$$R_z \approx 0.01, \text{ eftersom } |r_z| = \frac{|e_z|}{|z|} \leq \frac{E_z}{0.495024} = 0.0101025\cdots$$

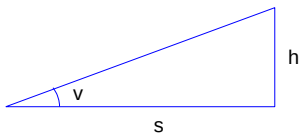
2. Vi skall beräkna f då $\frac{1}{f} = \frac{1}{a} + \frac{1}{b}$, med $a = 200 \pm 1$ och $b = 100 \pm 0.5$. Direkt räkning ger

$$\frac{1}{f} \leq \frac{1}{\tilde{a} - E_a} + \frac{1}{\tilde{b} - E_b} = \frac{1}{199} + \frac{1}{99.5} = \frac{1}{66.333\cdots}$$

$$\frac{1}{f} \geq \frac{1}{\tilde{a} + E_a} + \frac{1}{\tilde{b} + E_b} = \frac{1}{201} + \frac{1}{100.5} = \frac{1}{67}$$

varav $66.333\cdots \leq f \leq 67$. Så vi tar $\tilde{f} = 66.667$ med $|e_f| = |\tilde{f} - f| \leq 0.334$, dvs. $E_f = 0.334$, och $R_f \approx 0.005$.

3. Höjden ges av $h = s\ \tan(v)$



$\tan(v)$ växande på $-\pi/2 \leq v \leq \pi/2 \Rightarrow 29\ \tan(18°) \leq h \leq 31\ \tan(20°)$

Följer att $9.422... \leq h \leq 11.283...$ eller $h = 10.35 \pm 0.93$

4. Vi har $y(x) = \cos(x)$ och $\tilde{x} = 0$ med $|e_x| \leq 0.001$. Felfortplantningsformeln $e_y \approx y'(\tilde{x})e_x$ duger inte ty $y'(\tilde{x}) = y'(0) = -\sin(0) = 0$. Vi skulle få $e_y \approx 0$ men vi måste få fram storleksordningen. Taylorutveckling runt $\tilde{x}$ ger

$$y(x) = y(\tilde{x}) + y'(\tilde{x})(x - \tilde{x}) + \frac{1}{2}y''(\tilde{x})(x - \tilde{x})^2 + \cdots \approx$$

$$\approx y(\tilde{x}) - y'(\tilde{x})e_x + \frac{1}{2}y''(\tilde{x})e_x^2$$

Eftersom $y'(\tilde{x}) = 0$ får vi $e_y \approx -\frac{1}{2}y''(\tilde{x})e_x^2$. Nu är $y''(x) = -\cos(x)$ så vi får $|e_y| \lesssim \frac{1}{2}e_x^2 \leq \frac{1}{2} \cdot 10^{-6}$.

5.(a) För algoritm 1 får vi

$$fl(fl(x^2) - fl(y^2)) = (x^2(1 + r_1) - y^2(1 + r_2))(1 + r_3),$$

där $|r_1| \leq \mu$, $|r_2| \leq \mu$ och $|r_3| \leq \mu$. Om vi utvecklar högerledet får vi

$$x^2(1 + r_1 + r_3 + r_1r_3) - y^2(1 + r_2 + r_3 + r_2r_3) =$$

$$= x^2 - y^2 + r_3(x^2 - y^2) + r_1x^2 - r_2y^2 + x^2r_1r_3 - y^2r_2r_3$$

Så för det absoluta felet gäller

$$fl(x^2 - y^2) - (x^2 - y^2) \approx r_3(x^2 - y^2) + r_1x^2 - r_2y^2$$

där vi bortsett från termer som är av storleksordningen $\mathcal{O}(\mu^2)$.

För algoritm 2 får vi (med $|r_1| \leq \mu$, $|r_2| \leq \mu$ och $|r_3| \leq \mu$)

$$fl(fl(x + y) \cdot fl(x - y)) =$$

$$= ((x + y)(1 + r_1)(x - y)(1 + r_2))(1 + r_3) =$$

$$= (x^2 - y^2)(1 + r_1)(1 + r_2)(1 + r_3) =$$

$$= (x^2 - y^2)(1 + r_1 + r_2 + r_3 + r_1r_2 + r_1r_3 + r_2r_3 + r_1r_2r_3)$$

Så för det absoluta felet gäller

$$fl((x + y)(x - y)) - (x^2 - y^2) \approx (x^2 - y^2)(r_1 + r_2 + r_3)$$

där vi bortsett från termer som är $\mathcal{O}(\mu^2)$.

(b) För algoritm 1 får vi

$$\text{relativa felet } \approx r_3 + \frac{r_1x^2 - r_2y^2}{x^2 - y^2}$$

där den sista termen kan bli obegränsat stor då $|x| \approx |y|$, medan för algoritm 2 får vi

$$\text{relativa felet } \approx r_1 + r_2 + r_3$$

som begränsas av 3μ oberoende av x och y. Så algoritm 2 är bättre än algoritm 1.

2 Ickelinjära ekvationer.

Vi skall studera numerisk lösning av ickelinjära ekvationer,

$$f(x) = 0, \tag{2.1}$$

där f är en skalär funktion i en variabel, dvs. $f : \mathbb{R} \to \mathbb{R}$.

En rot x^* till ekvationen $f(x) = 0$ kallas en enkelrot om derivatan $f'(x^*) \neq 0$ annars kallas den en multipelrot. Om $f'(x^*) = \cdots = f^{(m-1)}(x^*) = 0$ men $f^{(m)}(x^*) \neq 0$ så säger man att x^* har multipliciteten m. För en multipelrot x^* gäller att tangenten till grafen av f i $x = x^*$ sammanfaller med x-axeln.

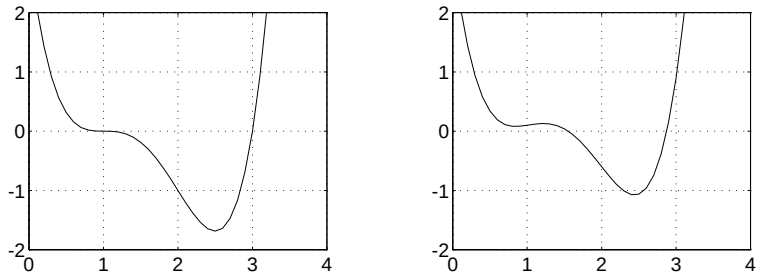
Exempel 2.1. Betrakta ekvationen

$$f(x) \equiv x^4 - 6x^3 + 12x^2 - 10x + 3 = 0.$$

I figuren till vänster ser vi rötterna till denna ekvation som skärningspunkterna mellan funktionens graf och x-axeln. Derivering ger

$$f'(x) = 4x^3 - 18x^2 + 24x - 10$$
$$f''(x) = 12x^2 - 36x + 24$$
$$f'''(x) = 24x - 36$$

Eftersom $f(1) = f'(1) = f''(1) = 0$ men $f'''(1) = -12 \neq 0$ så är $x = 1$ en trippelrot till ekvationen och $x = 3$ är en enkelrot då $f(3) = 0$ men $f'(3) = 8 \neq 0$.



Till höger i figuren har vi ändrat koefficienten framför x^2-termen med knappt 1% till 12.1. Vi får då två (reella) rötter $x = 1.5488$ och $x = 2.8746$, en förändring med 55% resp. 4%. ■

Av det här exemplet ser vi att det är tveksamt om man överhuvud taget skall försöka bestämma multipelrötter.

När vi vill bestämma rötterna till en ekvation $f(x) = 0$, ritar vi först upp grafen till funktionen f och gör en grov lokalisering av dem, för att sedan bestämma dem noggrant. Om vi har en enkel rot så får vi en teckenväxling hos funktionen f i ett intervall runt roten. Vi kan då använda den s.k. intervallhalveringsmetoden för att få en noggrann bestämning av roten. Man delar då intervallet i mitten och behåller det av delintervallen där funktionen växlar tecken. Sedan upprepar man detta förfarande tills det kvarvarande intervallet är tillräckligt litet.

Intervallhalveringsmetoden kan formuleras;

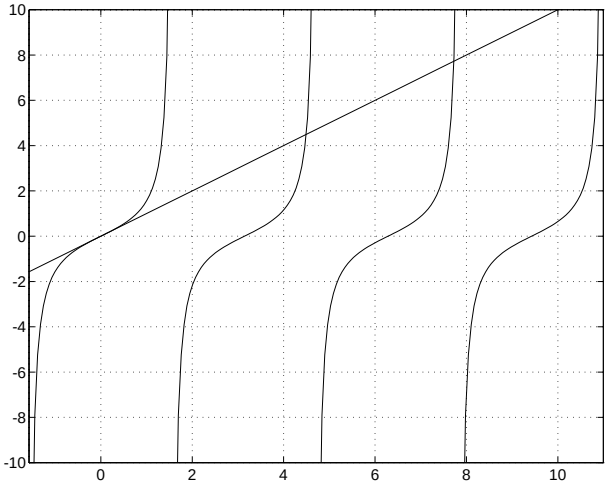
Givet ett intervall $I^{(0)} = [a, b]$ med teckenväxling, dvs. $f(a) \cdot f(b) < 0$ bestäm en följd av intervall $\{I^{(k)}\}$ enligt,

$$I^{(k)} = \begin{cases} [a, \frac{a+b}{2}], & \text{om } f(a) \cdot f(\frac{a+b}{2}) < 0 \\ [\frac{a+b}{2}, b], & \text{annars} \end{cases}, \; k = 1, 2, \cdots$$

Exempel 2.2. Låt oss se på ekvationen

$$f(x) \equiv \tan(x) - x = 0$$

Vi ritar linjen $y = x$ och kurvan $y = \tan(x)$ och finner rötterna som x-koordinaterna för skärningspunkterna mellan linjen och kurvan. Från grafen kan vi kanske se att vi har den minsta positiva roten i intervallet $4.4 \leq x \leq 4.6$ och den näst minsta i $7.6 \leq x \leq 7.8$.



Vi använder intervallhalveringsmetoden utgående ifrån dessa intervall. I MATLAB skulle det kunna se ut så här (för den minsta positiva roten)

```
>> a=4.4; fa=tan(a)-a;
>> b=4.6; fb=tan(b)-b;
>> fa*fb

ans =
   -5.5539
```

Vi ser att vi har en teckenväxling och fortsätter med

```
>> for k=1:20
      m=(a+b)/2; fm=tan(m)-m; disp([m fm])
      if fa*fm<0
         b=m; fb=fm;      % Teckenväxling mellan a och m, [a,m] nytt intervall.
      else
         a=m; fa=fm;      % Teckenväxling mellan m och b, [m,b] nytt intervall.
      end
   end
```

Resultatet ser vi i tabellen; medelpunkten i de reducerade intervallen samt storleken på f i medelpunkten.

k	$x^{(k)}$	$f(x^{(k)})$
0	4.50000000000000	0.13733205455118
1	4.45000000000000	-0.72673142705204
2	4.47500000000000	-0.34193309600374
3	4.48750000000000	-0.11607846046461
4	4.49375000000000	0.00688685252299
5	4.49062500000000	-0.05549125562777
6	4.49218750000000	-0.02453083095561
7	4.49296875000000	-0.00887975722717
8	4.49335937500000	-0.00101097164019
9	4.49355468750000	0.00293430089835
10	4.49345703125000	0.00096075596117
11	4.49340820312500	-0.00002533485450
12	4.49343261718750	0.00046765378061
13	4.49342041015625	0.00022114527223
14	4.49341430664063	0.00009790166145
15	4.49341125488281	0.00003628251667
16	4.49340972900391	0.00000547360940
17	4.49340896606445	-0.00000993067798
18	4.49340934753418	-0.00000222854815
19	4.49340953826904	0.00000162252716
20	4.49340944290161	-0.00000030301137

■

Från vårt exempel ser vi att intervallhalvering konvergerar mycket långsamt. Om vi tar $x^{(k)}$ som de successiva intervallens mittpunkter så gäller att

$$|x^{(k)} - x^*| \leq \frac{1}{2^k}|x^{(0)} - x^*|$$

För att få en ny korrekt decimal i approximationen av x^* måste vi i snitt göra 3.32 halveringar ($2^k = 10 \Rightarrow k \approx 3.32$). Detta är en alltför långsam konvergens för praktiska tillämpningar. Vi beräknar funktionen f vid varje delning, men vi använder bara funktionsvärdets tecken. Vi behöver mer effektiva metoder som inte ödslar beräkningskraft på detta sätt. Sådana skall vi se på nu.

2.1 Några iterationsmetoder.

En iterationsmetod är en metod som utgående ifrån en första approximation $x^{(0)}$ av en rot x^* till ekvationen $f(x) = 0$ genererar en talföljd $\{x^{(k)}\}_{k=0}^{\infty}$ som förhoppningsvis konvergerar mot x^*, dvs. sådan att $\lim_{k\to\infty} x^{(k)} = x^*$.

Om $\{x^{(k)}\}_{k=0}^{\infty}$ är en talföljd som konvergerar mot x^*, så är konvergensordningen det största positiva talet q, sådant att

$$\lim_{k\to\infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|^q} = C < \infty.$$

C kallas den asymptotiska felkonstanten. Om $q = 1$ så sägs konvergensen vara linjär, om $1 < q < 2$ så sägs konvergensen vara superlinjär och om $q = 2$ så sägs konvergensen vara kvadratisk.

Newtons metod.

Antag att $x^{(0)}$ är en approximation av en rot till ekvationen $f(x) = 0$.
Taylorutvecklingen runt $x^{(0)}$ ger;

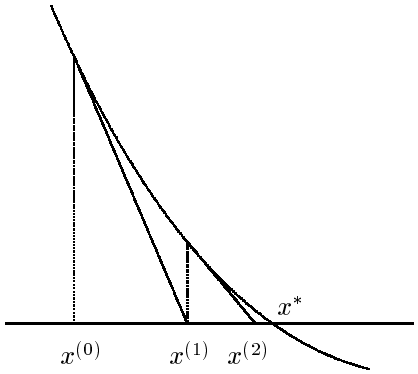
$$f(x) \approx f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}). \tag{2.2}$$

Vi får en lokal linjär modell av ekvationen genom att sätta högerledet i (2.2) till 0;

$$f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}) = 0$$

och lösa ut x som vi tar som nästa approximation $x^{(1)}$ av roten.

I figuren nedan ser vi hur vi utgående ifrån startapproximationen $x^{(0)}$ räknar fram $x^{(1)}$ genom att lokalt approximera f med tangentlinjen och följa den ned till x-axeln.



Allmänt kan Newtons metod formuleras;

Givet en approximativ lösning $x^{(0)}$ av $f(x) = 0$, bestäm en följd av approximationer $\{x^{(k)}\}$ enligt,

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, k = 0, 1, \cdots \tag{2.3}$$

Om startapproximationen ligger tillräckligt nära en rot så konvergerar iterationerna mot denna rot. En olämplig startapproximation leder till att Newtons metod divergerar.

Exempel 2.3. Återigen ser vi på ekvationen i exempel 2.2. Vi har $f(x) = \tan(x) - x$ med derivatan $f'(x) = \tan^2(x)$. Om vi tar $x^{(0)} = 4.5$ som startapproximation får vi med MATLAB

```
>> x=4.5; f=tan(x)-x; disp([x f])
>> for k=1:5
      fprim=tan(x)^2;
      x=x-f/fprim;
      f=tan(x)-x; disp([x f])
   end
```

Som ger följande resultat

k	$x^{(k)}$	$f(x^{(k)})$
0	4.50000000000000	0.13733205455118
1	4.49361390274320	0.00413187378926
2	4.49340965501325	0.00000397968077
3	4.49340945790925	0.00000000000369
4	4.49340945790906	0.00000000000000
5	4.49340945790906	0.00000000000000

Vi ser att vi får mycket snabb konvergens, antal korrekta decimaler fördubblas ungefär i varje steg. ■

Om x^* är roten så ger Taylorutveckling runt $x^{(k)}$;

$$0 = f(x^*) = f(x^{(k)}) + f'(x^{(k)})(x^* - x^{(k)}) + \frac{1}{2}f''(\xi^{(k)})(x^* - x^{(k)})^2,$$

där $\xi^{(k)}$ ligger mellan x^* och $x^{(k)}$. Från (2.3) har vi

$$f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)}).$$

Addition av dessa uttryck och en omflyttning ger

$$f'(x^{(k)})(x^{(k+1)} - x^*) = \frac{1}{2}f''(\xi^{(k)})(x^* - x^{(k)})^2.$$

Så om $f'(x^{(k)}) \neq 0$ så gäller

$$|x^{(k+1)} - x^*| = \frac{|f''(\xi^{(k)})|}{|2f'(x^{(k)})|}|x^{(k)} - x^*|^2,$$

dvs. om Newtons metod konvergerar mot en enkelrot[3] så är konvergensen kvadratisk och den asymptotiska felkonstanten blir

$$C = \frac{|f''(x^*)|}{|2f'(x^*)|}.$$

Kvadratisk konvergens innebär att nära roten får vi ungefär en fördubbling av antalet korrekta decimaler i varje iteration.

Sekantmetoden.

För att få en metod som inte använder derivator kan man ersätta $f'(x^{(k)})$ i Newtons metod med en differensapproximation

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}},$$

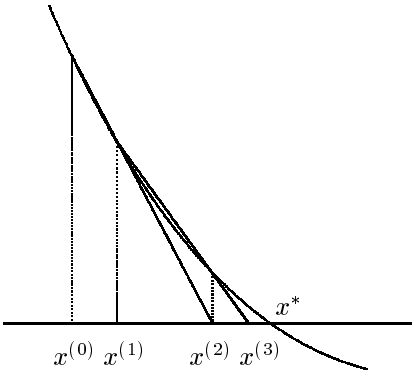
och får då sekantmetoden;

Givet två approximationer $x^{(0)}$ och $x^{(1)}$ av lösningen till $f(x) = 0$, bestäm en följd av approximationer $\{x^{(k)}\}$ enligt,

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}, k = 1, 2, \cdots$$

Geometriskt kan vi tolka metoden som att vi drar sekanten mellan två punkter på grafen och följer den tills den skär x-axeln.

[3] Om vi har en multipelrot så blir konvergensen endast linjär för Newtons metod och asymptotiska felkonstanten blir $C = \frac{m-1}{m}$, där m är rotens multiplicitet.



Exempel 2.4. Samma ekvation som förut, dvs. $f(x) \equiv \tan(x) - x = 0$. Vi startar med $x^{(0)} = 4.4$ och $x^{(1)} = 4.6$. Sekantmetoden kan då skrivas så här i MATLAB

```
>> x0=4.4 ; x1=4.6;
>> f0=tan(x0)-x0; f1=tan(x1)-x1;
>> disp([x0 f0]), disp([x1 f1])
>> for k=2:10
      x=x1-f1*(x1-x0)/(f1-f0);
      f=tan(x)-x;
      disp([x f])
      x0=x1; x1=x; f0=f1; f1=f;
   end
```

Vi får följande resultat

k	$x^{(k)}$	$f(x^{(k)})$
0	4.40000000000000	-1.30367621935025
1	4.60000000000000	4.26017489564805
2	4.44686236897447	-0.76968920869564
3	4.47029608002518	-0.42066508206646
4	4.49853981227796	0.10615193525585
5	4.49284879103358	-0.01129041742933
6	4.49339590205426	-0.00027368508835
7	4.49340949375490	0.00000072375364
8	4.49340945790677	-0.00000000004627
9	4.49340945790906	0.00000000000000
10	4.49340945790906	0.00000000000000

■

Man kan visa att konvergensordningen för sekantmetoden, vid bestämning av enkelrötter, är $q \approx 1.618$, dvs. konvergensen är då superlinjär. Detta innebär att nära roten får vi ungefär 1.6 gånger så många korrekta decimaler för varje iteration. Sekantmetoden är, trots den lägre konvergensordningen, ofta mer effektiv än Newtons metod eftersom sekantmetoden bara gör en funktionsberäkning per iteration medan Newtons metod dessutom gör en derivataberäkning.

Hybridmetoder.

Både Newtons metod och sekantmetoden är lokalt konvergenta metoder, medan intervallhalveringsmetoden är en globalt konvergent. Genom att använda en kombination av intervallhalveringsmetoden och lokala metoder kan man förena de goda egenskaperna hos respektive metod och undvika de dåliga. Man får då s.k. hybridmetoder. Det finns mycket effektiva sådana metoder i olika programbibliotek. I MATLAB finner vi **fzero** som bygger på denna teknik.

2.2 Lösningsnoggrannhet.

I praktiken kommer man att avbryta iterationsmetoderna efter ändligt många iterationer. Detta samt avrundningsfel under beräkningen av funktionen f gör att man tvingas acceptera approximativa lösningar $\bar{x}$ med $f(\bar{x}) \neq 0$.

Låt x^* vara den exakta lösningen till ekvationen och låt $\delta x = \bar{x} - x^*$ vara felet i vår approximativa lösning.

Taylorutveckling runt x^* ger

$$f(\bar{x}) = f(x^* + \delta x) = f(x^*) + f'(x^*)\delta x + \mathcal{O}(\delta x^2) =$$

$$= f'(x^*)\delta x + \mathcal{O}(\delta x^2).$$

För små δx gäller

$$f'(x^*)\delta x \approx f(\bar{x}),$$

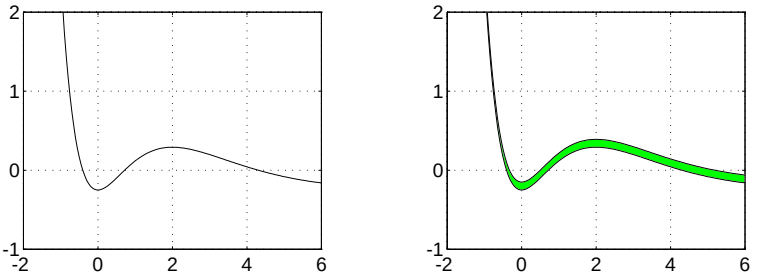
så om $f'(x^*) \neq 0$, dvs. om roten är enkel[4] , har vi

$$\delta x \approx \frac{f(\bar{x})}{f'(x^*)} \approx \frac{f(\bar{x})}{f'(\bar{x})}.$$

Exempel 2.5. Låt oss betrakta funktionen

$$f(x) = x^2 \mathrm{e}^{-x} - 0.2$$

Ekvationen $f(x) = 0$ har tre enkelrötter som vi ser i vänster figur nedan. I höger figur har vi ritat grafen av $f + \varepsilon$ för $|\varepsilon| \leq 0.05$. Vi ser hur derivatan kommer in. Den minsta roten x_{min} är välbestämd (välkonditionerad) eftersom derivatan är stor i dess omgivning. Däremot är den största roten x_{max} dåligt bestämd (illakonditionerad), derivatan är ju liten där. Närmare bestämt får vi $x_{min} = -0.37 \pm 0.04$ och $x_{max} = 4.75 \pm 0.45$.



■

Ett alternativ är följande: Antag att $\bar{x}$ är vår approximativa rot och vi tar $E = \frac{1}{2} \cdot 10^{-t}$ för något positivt heltal t. Om vi kan visa att funktionen f växlar tecken på intervallet $\bar{x} - E \leq x \leq \bar{x} + E$, dvs. $f(\bar{x} - E) \cdot f(\bar{x} + E) < 0$, så blir f noll någonstans i intervallet (om f är kontinuerlig), och därmed har $\bar{x}$ t korrekta decimaler.

2.3 Användning av fzero.

Antag vi skall bestämma den näst minsta positiva roten till ekvationen i exempel 2.2. Vi börjar med att rita grafen till $f(x) = \tan(x) - x$.

Från figuren i exemplet ser vi att aktuella roten bör ligga mellen 7.4 och 7.8. Vi börjar med att bilda en vektor, eller en lista, av säg 200 tal jämnt fördelade mellan 7.4 och 7.8 med:

```
>> x=linspace(7.4,7.8,200);
```

Vi undvek utskrift av talen genom att sätta ett semikolon (**;**) efter uttrycket. Nu definierar vi funktionen f som en inline-funktion (vi kan även göra en **function**-fil)

```
>> f=inline('tan(x)-x','x')
f =
     Inline function:
     f(x) = tan(x)-x
```

[4] Om $f'(x^*) = 0$, dvs. om vi har en multipelrot, tar vi med fler termer i Taylorutvecklingen och får,

$$f(\bar{x}) = \frac{f^{(m)}(x^*)}{m!}\delta x^m + \mathcal{O}(\delta x^{m+1}).$$

För små δx gäller då

$$|\delta x| \approx \sqrt[m]{m!\frac{|f(\bar{x})|}{|f^{(m)}(x^*)|}} \approx \sqrt[m]{m!\frac{|f(\bar{x})|}{|f^{(m)}(\bar{x})|}},$$

där m är rotens multiplicitet.

MATLAB bekräftar definitionen, undvik utskrift med **;** efter uttrycket. Nu är det dags att rita grafen

```
>> plot(x,f(x))
```

Vi lägger på ett rutnät, för att lättare se var roten är, med

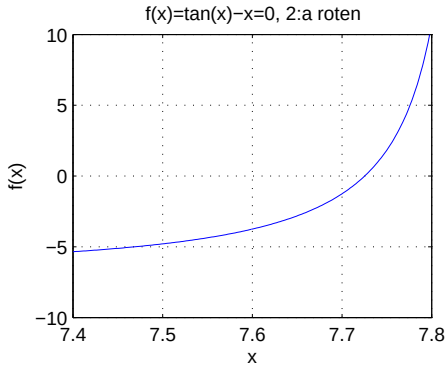
```
>> grid on
```

För att få en snyggare figur väljer vi lämplig skala på koordinataxlarna med

```
>> axis([7.4 7.8 -10 10])
```

och sätter rubrik och axeltexter med

```
>> title('f(x)=tan(x)-x=0, 2:a roten')
>> xlabel('x'), ylabel('f(x)')
```



Nu läser vi in en startapproximation $x^{(0)}$ med hjälp av

```
>> [x0,y0]=ginput(1);
```

Hårkorset placeras över skärningspunkten mellan grafen och y-axeln. Musknapp trycks ned och punktens koordinater hamnar i `x0` respektive `y0`.

Vi ser på $x^{(0)}$ och $f(x^{(0)})$ med **disp** som ger en kompakt utskrift

```
>> disp([x0,f(x0)])
   7.7355e+00   6.6645e-01
```

Vi tar nu `x0` som startapproximation till **fzero** för att få en noggrannare approximation

```
>> x=fzero(f,x0); disp([x,f(x)])
   7.7253e+00  -2.3093e-14
```

Om vi vill ha statistik utskriven från **fzero** så får vi det med `x=fzero(f,x0,optimset('display','iter'))`. Beror vår ekvation av en parameter p kan värden på den vidarebefodras via **fzero** till funktionen som definierar ekvationen enligt

```
>> x=fzero(f,x0,[],p)
```

Här har vi valt att sätta en platshållare (tomma mängden) istället för att använda **optimset**, vi får då default-värdet på `'display'` som är `'off'`.

Om funktionen istället är beskriven på en textfil, en s.k. **function**-fil, med namnet `f_ex2_2.m` och innehållet

```
function f=f_ex2_2(x)
f=tan(x)-x;
```

så finner vi roten med kommandot `>> fzero(@f_ex2_2,x0)`.

2.4 Övningar.

1. Visa att ekvationen $x^2 - \cos(x) = 0$ har precis en positiv rot. Beräkna sedan denna rot med fyra korrekta decimaler.

2. För att studera egenfrekvensen vid torisionssvängning vill man beräkna några av de minsta positiva rötterna till ekvationen

$$x\tan(x) = a,\ a > 0$$

(a) Hur många positiva rötter finns det och ungefär var ligger de.

(b) Beräkna den minsta positiva roten, för $a = 2$, med fyra korrekta decimaler.

3. Man kan beräkna $\sqrt{c}$, med enbart upprepade additioner och divisioner, genom att lösa ekvationen

$$x^2 - c = 0,$$

med t.ex. Newtons metod.

(a) Hur ser iterationsformeln ut om vi använder Newtons metod? För vilka startapproximationer konvergerar metoden?

(b) Visa att konvergensen är kvadratisk för den här ekvationen då $c \neq 0$. Vad gäller då $c = 0$. Bestäm den asymptotiska felkonstanten i båda fallen.

4. Ljudnivån L (i decibel) på avståndet r meter från en ljudkälla ges av formeln

$$L = L_0 - 20\,{}^{10}\log(r) - \beta r$$

där L_0 är ljudnivån 1 meter från ljudkällan och β är en parameter som anger hur ljudet försvagas då det går genom luften. (β beror av luftens viskositet, temperatur och fuktighet.) Vi vill för $L_0 = 80$ dB bestämma det avstånd där ljudnivån är 20 dB då $\beta = 1.15 \cdot 10^{-3}$. Rita en lämplig graf, med **plot**, och läs av en startapproximation av avståndet, med **ginput**. Bestäm sedan en noggrannare approximation med **fzero**.

5. Rita den kurva som implicit definieras av

$$x^2 + y^2 = 1 + a\sin(xy)$$

för $a = 0.8, 1.6, 2.4$ och 3.2. Ledning: Ansätt polära koordinater och lös ut radien, som funktion av vinkeln, för olika vinklar med **fzero**.

6. Betrakta polynomet

$$p(x) = (x-1)(x-2)(x-3)\cdots(x-7) =$$

$$= x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13132x^2 + 13068x - 5040.$$

Representera polynomet med en vektor som innehåller dess koefficienter. Detta kan göras i MATLAB med **poly**.

7. En lång stång upphettas momentant vid tiden $t = 0$ i mittpunkten. Med denna punkt i $x = 0$ och x-axeln utmed stången bestäms temperaturen $T(x,t)$ för $t > 0$ av uttrycket

$$T(x,t) = Ce^{-x^2/kt}/\sqrt{t}$$

där parametrarna $k = 5.17$ och $C = 28.3$ beror av värmeledningsförmågan respektive den tillförda värmen.

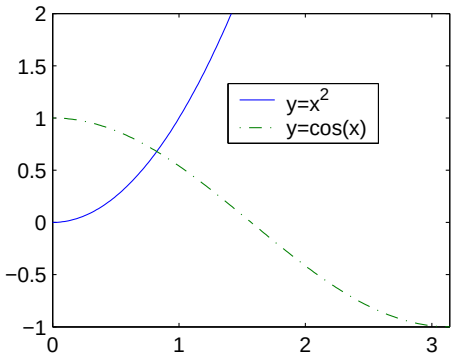
Beräkna under hur lång tid som temperaturen vid $x = 1$ överstiger 20°. Använd **fzero**.

2.5 Lösningar.

1. Vi ritar upp $y = x^2$ och $y = \cos(x)$ för att se var kurvorna skär varandra.

   Med MATLAB görs detta så här

```
>> x=linspace(0,pi,200);
>> plot(x,x.^2,x,cos(x),'-.')
>> axis([0 pi -1 2])
>> legend('y=x^2','y=cos(x)')
```



   Låt $f(x) = x^2 - \cos(x)$. Vi vill alltså lösa $f(x) = 0$.

$$\left.\begin{array}{l} x^2 \text{ växer monotont på } x \geq 0 \\ \cos(x) \text{ avtar monotont i } 0 \leq x \leq \pi \end{array}\right\} \Rightarrow \text{ högst en rot i } 0 \leq x \leq \pi$$

$$\left.\begin{array}{l} f(0) = -1 < 0 \\ f(\frac{\pi}{2}) = (\frac{\pi}{2})^2 > 0 \end{array}\right\} \text{ (teckenväxling) } \Rightarrow \text{ minst en rot i } 0 \leq x \leq \frac{\pi}{2}$$

   Så precis en rot i $0 \leq x \leq \frac{\pi}{2}$ och ingen i $\frac{\pi}{2} < x \leq \pi$.

   Eftersom $x^2 > \pi^2 > 1 = \max_x \cos(x)$ då $x > \pi$ så finns det ingen rot i $\pi < x < \infty$.

   Vi tar $x^{(0)} = 1$ som startapproximation i Newtons metod ($f'(x) = 2x + \sin(x)$) och får:

```
>> x=1; disp(x)                          1
>> f=inline('x^2-cos(x)','x');           0.83821840990471
>> fprim=inline('2*x+sin(x)','x');       0.82424186822587
>> for k=1:5                             0.82413231905093
      x=x-f(x)/fprim(x); disp(x)         0.82413231230252
   end                                   0.82413231230252
```

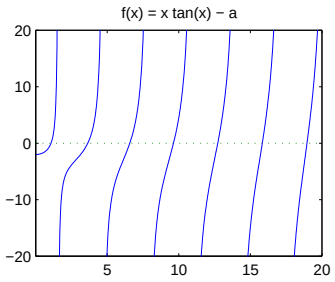
   Så vi tar $\tilde{x} = 0.8241$ som approximation av roten med felgräns $E = \frac{1}{2} \cdot 10^{-4}$.

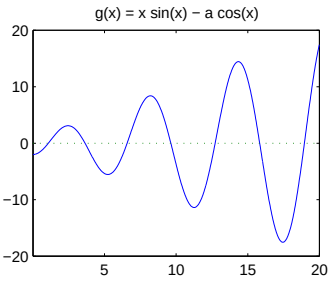
```
>> xapprox=round(1e4*x)/1e4              0.82410000000000
>> E=0.5e-4;
>> f(x-E)                               -1.191078187362526e-04
>> f(x+E)                                1.191145167211971e-04
```

   Eftersom $f(\tilde{x} - E)$ och $f(\tilde{x} + E)$ har olika tecken är saken klar.

2. Vi skall lösa $f(x) = x \tan(x) - a = 0$, för $a = 2$. Börjar med att rita graf med MATLAB

```
>> a=2; xmin=0; xmax=20; fmin=-200; fmax=200;
>> x=linspace(xmin,xmax,500); nx=[xmin xmax]; ny=[0 0];
>> f=x.*tan(x)-a;
>> f(f<fmin)=-Inf;          % Dessa två rader behövs inte men ger snyggare graf.
>> f(f>fmax)=Inf;           % De lodräta asymptoterna kommer annars med.
>> subplot(2,2,1), plot(x,f,nx,ny,':')          % Graf med nollnivån prickad.
>> axis([xmin xmax -20 20]), title('f(x) = x tan(x) - a')
```

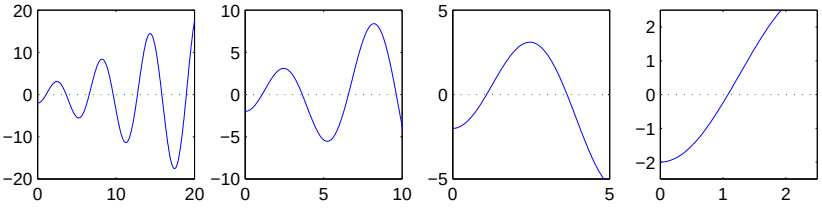




Bättre att formulera om: $x\ \tan(x) = a \Rightarrow x\ \sin(x) = a\ \cos(x)$. Så vi löser istället $g(x) = x\ \sin(x) - a\ \cos(x) = 0$

```
>> g=inline('x.*sin(x)-a*cos(x)','x','a');   % Vi låter a vara en parameter.
>> subplot(2,2,2), plot(x,g(x,a),nx,ny,':')
>> axis([xmin xmax -20 20]), title('g(x) = x sin(x) - a cos(x)')
```

Vi använder **zoom** för att lokalisera aktuellt nollställe



Får noggrann approximation med **fzero**

```
>> [x0,y]=ginput(1);
>> x=fzero(g,x0,[],a)                          % Observera hur a kommer med.
x =
    1.0769
```

3.(a) $f(x) = x^2 - c, f'(x) = 2x$

$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} = \frac{1}{2}(x^{(k)} + \frac{c}{x^{(k)}}),\ k = 0, 1, \cdots$

Konvergens för $x^{(0)} > 0$ (och då blir alltid $x^{(1)} > \sqrt{c}$, sedan monoton konvergens).

(b) $x^{(k+1)} - \sqrt{c} = \frac{1}{2}(x^{(k)} + \frac{c}{x^{(k)}}) - \sqrt{c} =$

$$= \frac{1}{2x^{(k)}}((x^{(k)})^2 + c - 2x^{(k)}\sqrt{c}) = \frac{1}{2x^{(k)}}(x^{(k)} - \sqrt{c})^2$$

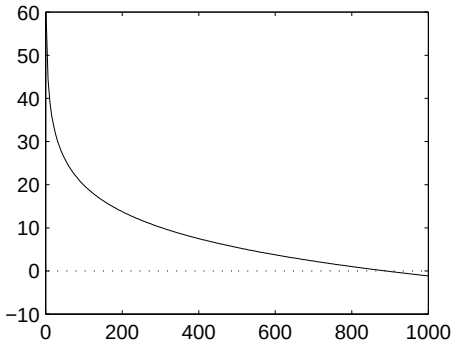
$$\frac{x^{(k+1)} - \sqrt{c}}{(x^{(k)} - \sqrt{c})^2} = \frac{1}{2x^{(k)}} \to \frac{1}{2\sqrt{c}} \text{ då } k \to \infty$$

Alltså kvadratisk konvergens med asymptotiska felkonstanten $\frac{1}{2\sqrt{c}}$.

Om $c = 0$ (dubbelrot) har vi $x^{(k+1)} = \frac{1}{2}x^{(k)}$, dvs. vi får linjär konvergens med asymptotiska felkonstanten $\frac{1}{2}$.

4. Vi skall lösa ekvationen $f(x) = 60 - 20^{10}\log(x) - 1.15 \cdot 10^{-3}x = 0$. Vi ritar grafen, läser av startapproximation med **ginput** och beräknar lösningen med **fzero**.

```
>> ljud=inline('60-20*log10(x)-1.15e-3*x','x')
ljud =
     Inline function:
     ljud(x) = 60-20*log10(x)-1.15e-3*x
>> x=linspace(1,1000,200);
>> plot(x,ljud(x),x,0*x,':')
>> [avst_0,y]=ginput(1);
>> avst=fzero(ljud,avst_0)
avst =
  888.9647
```



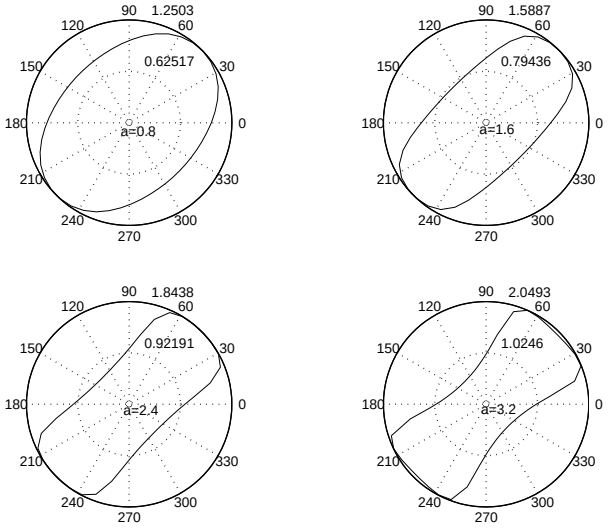
5. Låt $x = r(\theta)\cos(\theta), y = r(\theta)\sin(\theta)$. Insättning ger

$$x^2 + y^2 = r(\theta)^2,\ 1 + a\sin(xy) = 1 + a\sin(r(\theta)^2\frac{1}{2}\sin(2\theta))$$

Vi skall alltså, givet θ, lösa ekvationen $f(r) = 1 + a\sin(r^2\frac{1}{2}\sin(2\theta)) - r^2 = 0$.

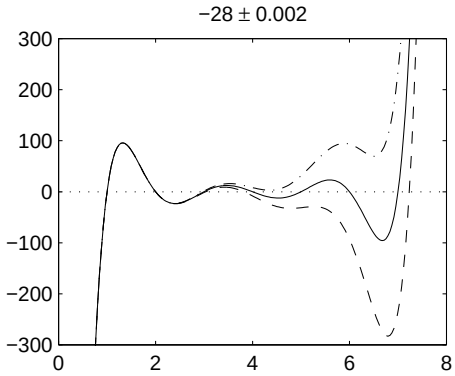
```
>> bloeja=inline('1+a*sin(r.^2*sin(2*theta)/2)-r.^2','r','theta','a');
>> a=0.8;
>> theta=linspace(0,2*pi,50); r=zeros(size(theta));
>> r(1)=fzero(bloeja,1,[],theta(1),a);
>> for k=2:length(theta)
       r(k)=fzero(bloeja,r(k-1),[],theta(k),a);
   end
>> subplot(2,2,1), polar(theta,r), text(-0.1,-0.1,'a=0.8')
osv.
```

Observera att vi tar $r(\theta_{k-1})$ som startapproximation av $r(\theta_k)$.



6. Vi ritar upp grafen av polynomet och dess störningar

```
>> x=linspace(0,8,200);
>> c=poly(1:7)
>> c1=c; c1(2)=c1(2)-0.002;
>> c2=c; c2(2)=c2(2)+0.002;
>> plot(x,polyval(c,x), ...
        x,polyval(c1,x),'--', ...
        x,polyval(c2,x),'-.',x,0*x,':')
>> axis([0 8 -300 300])
>> title('-28 \pm 0.002')
```



Vi ser att det händer rätt mycket med grafen i intervallet $3.5 \leq x \leq 7.5$ då vi stör x^6-koefficienten.

```
>> [roots(c) roots(c1) roots(c2)]
ans =
   7.0000            7.2330            6.6444 + 0.3908i
   6.0000            5.4587 + 0.5401i  6.6444 - 0.3908i
   5.0000            5.4587 - 0.5401i  4.3682 + 0.2244i
   4.0000            3.8196            4.3682 - 0.2244i
   3.0000            3.0331            2.9717
   2.0000            1.9989            2.0011
   1.0000            1.0000            1.0000
```

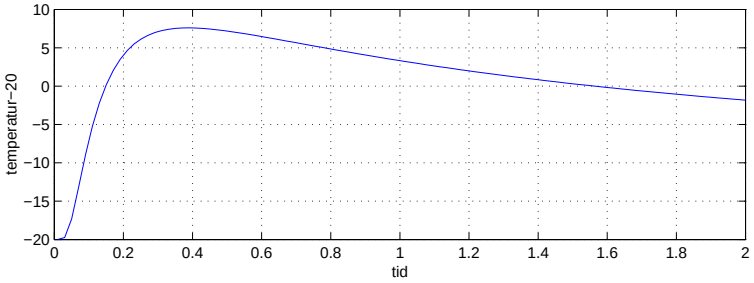
Vi ser att då x^6-koefficienten är -28.002 så har två reella nollställen gått ut i komplexa talplanet och då den är -27.998 så är det fyra som försvunnit.

7. Vi skapar en fil med beskrivning av vår funktion

```
function f=varm(t,k,C,x)
f=C*exp(-x^2/k./t)./sqrt(t)-20;
```

och ritar graf med

```
>> k=5.17; C=28.3; x=1;
>> t=linspace(0.01,2,100); plot(t,varm(t,k,C,x)), grid on
>> xlabel('tid'), ylabel('temperatur-20')
```



Sedan läser vi in startapproximationer samt finjusterar dessa med

```
>> [t0,T0]=ginput(1); t0=fzero(@varm,t0,[],k,C,x);
>> [t1,T1]=ginput(1); t1=fzero(@varm,t1,[],k,C,x);
>> tiden=t1-t0
tiden =
   1.4145e+00
```

3 Numerisk derivation och integration.

I det här avsnittet skall vi se på lite olika differensapproximationer av derivator. Vi skall också se på hur man kan approximera bestämda integraler.

3.1 Differensapproximation av derivator.

Om vi inför framåtdifferenskvoten

$$D_+y(x) = \frac{y(x+h) - y(x)}{h}$$

bakåtdifferenskvoten

$$D_-y(x) = \frac{y(x) - y(x-h)}{h}$$

och centraldifferenskvoten

$$D_0y(x) = \frac{y(x+h) - y(x-h)}{2h}$$

så är de approximationer av derivatan $y'(x)$.

Om vi inför centraldifferenskvoten

$$D_+D_-y(x) = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2}$$

så är den en approximation av andra derivatan $y''(x)$.

Det gäller att

$$y'(x) = D_+y(x) + \mathcal{O}(h) \tag{3.1}$$

$$y'(x) = D_-y(x) + \mathcal{O}(h) \tag{3.2}$$

$$y'(x) = D_0y(x) + \mathcal{O}(h^2) \tag{3.3}$$

och

$$y''(x) = D_+D_-y(x) + \mathcal{O}(h^2) \tag{3.4}$$

Framåt- och bakåtdifferenskvoten D_+ och D_- är första ordningens approximationer av $y'(x)$, medan centraldifferenskvoten D_0 är av andra ordningen. Centraldifferensen D_+D_- ger en andra ordningens approximation av $y''(x)$.

Nu skall vi se varför det blir så. Taylorutveckling av y runt x ger

$$y(x+h) = y(x) + y'(x)h + \frac{y''(x)}{2}h^2 + \frac{y^{(3)}(x)}{6}h^3 + \frac{y^{(4)}(x)}{24}h^4 + \cdots \tag{3.5}$$

och

$$y(x-h) = y(x) - y'(x)h + \frac{y''(x)}{2}h^2 - \frac{y^{(3)}(x)}{6}h^3 + \frac{y^{(4)}(x)}{24}h^4 - \cdots \tag{3.6}$$

Från (3.5) respektive (3.6) följer direkt relationerna (3.1) och (3.2). Subtraktion av (3.5) med (3.6) ger (3.3) och addition ger (3.4).

Exempel 3.1. Låt oss för $y(x) = \exp(x)$ approximera $y'(0.4)$ med framåtdifferensen D_+ och centraldifferensen D_0 för att jämföra approximationernas olika noggrannhetsordning. Vi får för en följd av allt mindre h-värden tabellen (exakt gäller $y'(0.4) = \exp(0.4) = 1.49182469764127\cdots$)

h	$D_+(h)y(x)$	$D_+(h)y(x) - y'(x)$
10^{-1}	1.56896573058858	0.07714103294731
10^{-2}	1.49930874715833	0.00748404951706
10^{-3}	1.49257085868970	0.00074616104843
10^{-4}	1.49189929136373	0.00007459372246
10^{-5}	1.49183215678583	0.00000745914456

för framåtdifferensen. Vi ser att felet är proportionellt mot h. För centraldifferensen får vi

h	$D_0(h)y(x)$	$D_0(h)y(x) - y'(x)$
10^{-1}	1.49431231562063	0.00248761797935
10^{-2}	1.49184956151056	0.00002486386929
10^{-3}	1.49182494627875	0.00000024863748
10^{-4}	1.49182470012832	0.00000000248705
10^{-5}	1.49182469766362	0.00000000002235

Här är felet proportionellt mot h^2. ■

3.2 Differensapproximation av hög noggrannhetsordning.

Om vi tar h mycket litet för att få ett litet diskretiseringsfel får vi kancellation vid beräkning av differenskvoten (funktionsvärdena är nästan lika och vi subtraherar). Behöver vi noggrannare approximation av derivator får vi istället använda differensapproximationer av högre noggrannhetsordning.

Exempel 3.2. Vi tar samma funktion som i exempel 3.1 och fortsätter att approximera med D_0 för ännu mindre h-värden, dvs. vi försöker låta $h \to 0$. Så här ser det ut

h	$D_0(h)y(x)$	$D_0(h)y(x) - y'(x)$
10^{-6}	1.49182469766362	0.00000000002235
10^{-7}	1.49182469821874	0.00000000057747
10^{-8}	1.49182469710851	-0.00000000053276
10^{-9}	1.49182488584643	0.00000018820516
10^{-10}	1.49182444175722	-0.00000025588405
10^{-11}	1.49182888264932	0.00000418500804
10^{-12}	1.49169565588636	-0.00012904175491

Felet börjar tydligen växa istället för att avtaga ytterligare. ■

Vi skall ta och se efter vad som händer med centraldifferenskvoten

$$D_0 y(x) = \frac{y(x+h) - y(x-h)}{2h}$$

om vi använder approximationer $\tilde{y}$ av y, med $|\tilde{y} - y| \leq E_y$.

Dessa approximationer använder vi i $D_0 y(x)$ och beräknar $\widetilde{D_0 y}(x)$ med

$$|\widetilde{D_0 y}(x) - D_0 y(x)| \leq \frac{2E_y}{2h} = \frac{E_y}{h}$$

Detta fel växer då h minskar. Om vi tar h mycket litet får vi ju kancellation vid beräkning av differenskvoten.

Diskretiseringsfelet

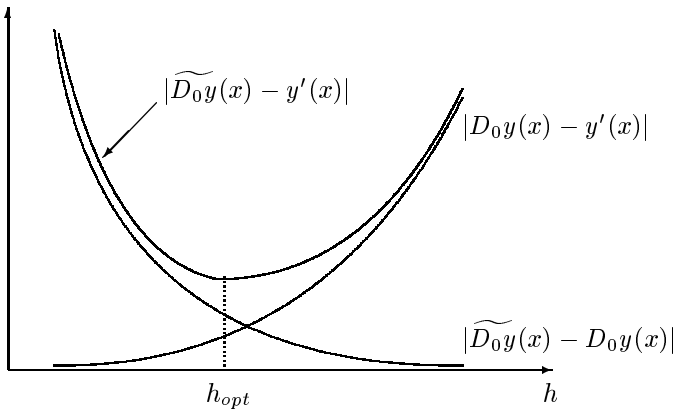
$$|D_0 y(x) - y'(x)| \lesssim \frac{h^2}{6}|y^{(3)}(x)|$$

minskar då h minskar.

Vi ritar en figur där vi ser att det finns ett optimalt h-värde, för vilket totala felet

$$|\widetilde{D_0 y}(x) - y'(x)|$$

blir så litet som möjligt.



Antag att vi approximerar $y'(x)$ med en centraldifferenskvot med steget h, låt $D_0(h)y(x)$ beteckna denna approximation. Detta är en andra ordningens approximation, men vi vill få en ännu noggrannare approximation. Om vi dessutom har använt steget $2h$, dvs. beräknat approximationen $D_0(2h)y(x)$ av $y'(x)$, så gäller

$$D_0(h)y(x) = \frac{y(x+h) - y(x-h)}{2h} = y'(x) + \frac{h^2}{6}y^{(3)}(x) + \mathcal{O}(h^4)$$

$$D_0(2h)y(x) = \frac{y(x+2h) - y(x-2h)}{4h} = y'(x) + 4\frac{h^2}{6}y^{(3)}(x) + \mathcal{O}(h^4)$$

Från dessa relationer får vi att

$$4D_0(h)y(x) - D_0(2h)y(x) = 3y'(x) + \mathcal{O}(h^4)$$

eller

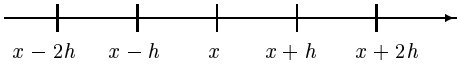
$$\left(D_0(h) + \frac{D_0(h) - D_0(2h)}{3}\right) y(x) = y'(x) + \mathcal{O}(h^4) \tag{3.7}$$

Vi har fått en 4:e ordningens approximation av derivatan. I vänsterledet i (3.7) kan vi uppfatta $\frac{D_0(h)-D_0(2h)}{3}$ som en korrektion av $D_0(h)$. Denna teknik att få noggrannare approximationer kallas Richardsonextrapolation.

Ser vi efter i detalj så blir differenskvoten i (3.7)

$$\frac{-y(x+2h) + 8y(x+h) - 8y(x-h) + y(x-2h)}{12h}$$

och de punkter som används är



Exempel 3.3. Vi använder nu det extrapolerade värdena från (3.7) på samma funktion som i exempel 3.1.

h	$D_{xtrp}(h)y(x)$	$D_{xtrp}(h)y(x) - y'(x)$
10^{-1}	1.49181971896888	-0.00000497867239
10^{-2}	1.49182469714400	-0.00000000049728
10^{-3}	1.49182469764133	0.00000000000006

Felet är proportionellt mot h^4. Givetvis kommer felet även för de extrapolerade värdena att börja växa då vi tar h alltför litet. Poängen är att vi får hög noggrannhet utan att ta h speciellt litet. ■

3.3 Numerisk integrering.

Ofta kan man inte analytiskt bestämma integraler

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

utan man får nöja sig med att beräkna approximationer. Till exempel integralen

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

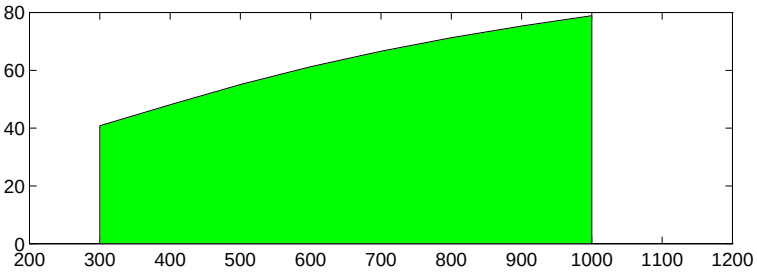
kan inte beräknas exakt, eftersom det inte finns något slutet uttryck för den primitiva funktionen. Men det kan också vara så att integranden bara är känd i vissa punkter, som i följande exempel.

Exempel 3.4. Man kan beräkna hur mycket energi det behövs för att upphetta en mol metylklorid (CH_3Cl) från 300 K till 1000 K genom att beräkna integralen

$$q_p = \int_{300}^{1000} C_p(T)dT$$

där $C_p(T)$ är värmekapaciteten per mol för metylklorid enligt tabellen

T (K)	300	400	500	600	700	800	900	1000
C_p (J/mol K)	40.82	48.10	55.09	61.25	66.60	71.26	75.33	78.90



■

Trapetsregeln och Simpsons regel.

En typ av metoder får man genom att ersätta integranden med ett interpolationspolynom och integrera detta (vi får s.k. Newton-Cotes formler).

Använder vi ett förstagrads polynom $P_1(x)$, som interpolerar f i a och b, får vi den s.k. trapetsregeln

$$\int_a^b f(x)dx \approx \int_a^b P_1(x)dx = \frac{b-a}{2}\{f(a)+f(b)\}.$$

Då $f(x) = P_1(x) + \frac{f''(\xi(x))}{2}(x-a)(x-b)$ (se appendix A) så blir felet vid approximationen (om f'' är kontinuerlig)

$$\int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)dx = -\frac{f''(\zeta)}{12}(b-a)^3,$$

där $a \leq \zeta \leq b$. Trapetsregeln är exakt för linjära polynom.

Om vi använder ett andragrads polynom $P_2(x)$, som interpolerar f i a, $\frac{a+b}{2}$ och b, får vi Simpsons regel

$$\int_a^b f(x)dx \approx \int_a^b P_2(x)dx = \frac{b-a}{6}\{f(a)+4f(\frac{a+b}{2})+f(b)\}.$$

Man kan visa att felet är (om $f^{(4)}$ är kontinuerlig)

$$-\frac{f^{(4)}(\zeta)}{2880}(b-a)^5,$$

där $a \leq \zeta \leq b$, dvs. Simpsons regeln är exakt för kubiska polynom.

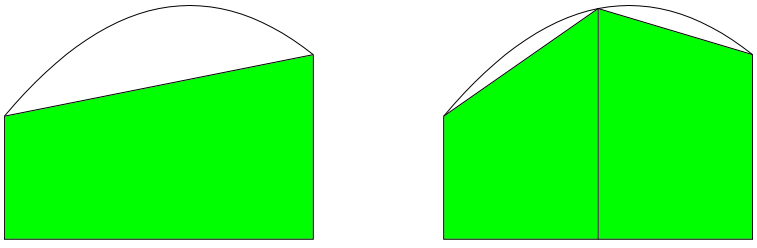
Trapetsmetoden och Simpsons metod.

För att få bättre noggrannhet i approximationen är det inte lämpligt att ta interpolationspolynom av allt högre gradtal, som vi förstår av interpolationsexemplet. Vad man kan göra är t.ex. att dela upp intervallet $a \leq x \leq b$ i n delintervall av längden $h = \frac{b-a}{n}$ och använda trapetsregeln på varje delintervall. Vi får då trapetsmetoden

$$T_n = h\{\frac{f(x_0)}{2} + f(x_1) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\},$$

där $x_i = a + ih$.

Man brukar successivt beräkna $T_1, T_2, T_4, \cdots$ så att man hela tiden utnyttjar redan beräknade funktionsvärden.



Vi har att $T_n \to \int_a^b f(x)dx$ då $n \to \infty$, dvs. trapetsmetoden är konvergent.

För T_n får vi ett fel för varje delintervall, det totala felet blir

$$-h^3 \sum_{i=0}^{n-1} \frac{f''(\zeta_i)}{12} = -\frac{b-a}{12} h^2 \frac{\sum_{i=0}^{n-1} f''(\zeta_i)}{n}$$

där $x_i \leq \zeta_i \leq x_{i+1}$. Nu kan vi skriva felet så här;

$$-\frac{b-a}{12} f''(\zeta) h^2,$$

där $a \leq \zeta \leq b$.

Med samma intervallindelning får vi Simpsons metod

$$S_n = \frac{h}{3}\{f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\}$$

med felet

$$-\frac{b-a}{180} f^{(4)}(\zeta) h^4$$

Adaptiva metoder.

Det finns en mängd av andra typer av metoder, t.ex. adaptiva metoder där man eftersträvar att utvärdera integranden f där det är störst variation i den. I MATLAB finns de adaptiva funktionerna **quad** och **quadl**.

3.4 Användning av quad och quadl.

Om vi exempelvis vill beräkna den bestämda integralen $\int_0^1 \mathrm{e}^{x^2}\, dx$ så kan den approximeras med

```
>> quad(inline('exp(x.^2)'),0,1)
```

Funtionen som beskriver integranden skall skrivas så att den beräknas för en vektor med x-värden. Vill vi ha resultatet med feltoleransen 10^{-8} (default är 10^{-6}) gör vi

```
>> quad(inline('exp(x.^2)'),0,1,1e-8)
```

Vid krav på hög noggrannhet bör man använda **quadl**.

Om integranden beror av en parameter och vi vill utföra integrationen för olika värden på parametern, t.ex. om

$$\int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - x^2 \sin^2(\theta)}}$$

skall beräknas för olika värden på x, så gör vi det enligt

```
>> quad(inline('1./sqrt(1-x^2*sin(theta).^2)','theta','x'),0,pi/2,[],[],x)
```

där x måste givits ett värde. Här har vi satt tomma mängden [ ] som platshållare för feltoleransen och får då default-toleransen, givetvis kan vi även sätta in önskad tolerans. Den andra platshållaren gäller en parameter som styr om det skall ske utskrift av delresultat under integrationen (default är ingen utskrift).

Har vi definierat integranden med en funktionsfil, t.ex.

```
function f=mitt_ex(x)
f=exp(x.^2);
```

så beräknas integralen (över $0 \leq x \leq 1$) med

```
>> quad(@mitt_ex,0,1)
```

3.5 Övningar.

1. Följande tabell över $f(x)$ är given

x	1.2	1.4	1.6
$f(x)$	3.6	3.8	4.6

   Uppskatta $f'(1.4)$ och $f''(1.4)$ med centraldifferenskvoter.

2. För centraldifferenskvoten

$$D_0(h)f(x) = \frac{f(x+h) - f(x-h)}{2h}$$

   gäller

$$f'(x) - D_0(h)f(x) = a_1h^2 + a_2h^4 + \cdots$$

   där koefficienterna $a_1, a_2, \cdots$ är oberoende av h.

   Antag att vi har följande tabell över entalpin H för 1 kg vatten vid 1 atm ($1.01325 \cdot 10^5$ Pa) tryck

T (°C)	H (J)
16	67.2632
17	71.4476
18	75.6312
19	79.8141
20	84.9963
21	88.1778
22	92.3589
23	96.5395
24	100.7196

   Beräkna en approximation av värmekapaciteten $C_p(T) = H'(T)$ vid temperaturen $T = 20°\mathrm{C}$ med hjälp av differensapproximation och Richardsonextrapolation. Gör en ordentlig feluppskattning. Tabellvärdena är korrekt avrundade.

3. Beräkna integralen i exempel 3.4 med trapetsmetoden. Hur stort kan beloppet av trunkeringsfelet högst bli om vi vet att andra derivatan av C_p till belopp är mindre än $8.3 \cdot 10^{-5}$.

   Hur mycket inverkar felen i tabellen på det beräknade värdet av q_p (tabellvärdena har två korrekta decimaler).

4. Beräkna en approximation av $\int_0^1 \tan(\sqrt{x})dx$. Integrandens derivata är obegränsad i $x = 0$ så en variabelsubstitution är lämplig innan man använder någon numerisk metod.

5. Vad gör man med $\int_0^1 \frac{\sin(x)}{\sqrt{1-x^2}}dx$? Integranden är ju singulär.

6. Rita upp cosinus- och sinusfunktionerna över intervallet $0 \le x \le 5$. Fyll det slutna området mellan graferna. Använd **fill**. Beräkna även arean av området med **quad**.

7. I samband med studiet av ljusdiffraktion stöter man på de s.k. Fresnelintegralerna

$$C(x) = \int_0^x \cos\left(\frac{\pi t^2}{2}\right) dt, \qquad S(x) = \int_0^x \sin\left(\frac{\pi t^2}{2}\right) dt$$

   Rita upp dessa funktioners grafer över intervallet $0 \le x \le 5$. Använd **quad** på ett förståndigt sätt, även repetitionssatsen **for** behövs.

3.6 Lösningar.

1. Vi har

$$f'(1.4) \approx \frac{f(1.6) - f(1.2)}{2 \cdot 0.2} = 2.5, \quad \text{resp.} \quad f'(1.4) \approx \frac{f(1.6) - 2f(1.4) + f(1.2)}{0.2^2} = 15$$

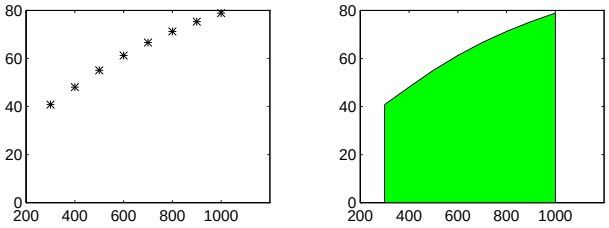
2. Richardson-extrapolation ger

h	$D_0(h)H(20)$	$\Delta/3$	$D_1(h)H(20)$
4	4.182050		
2	4.181925	-4.166667·10^{-5}	4.181883
1	4.181850	-2.500000·10^{-5}	4.181825

   Vi tar $D_1(1)H(20)$ som approximation av $H'(20)$.
   $E_{D_0(h)H} = \frac{2E_H}{2h} = \frac{E_H}{h} < \frac{1}{2} \cdot 10^{-4} \Rightarrow E_{D_1(h)H} \le \frac{5}{3} \cdot \frac{1}{2} \cdot 10^{-4}$.
   $|D_1(1)H(20) - H'(20)| \lesssim |D_1(1)H(20) - D_1(2)H(20)| = 5.8 \cdot 10^{-5}$.
   Vi får $H'(20) = 4.181825 \pm 1.5 \cdot 10^{-4}$

3. Med MATLAB beräknar vi q_p och skattar absoluta felet

```
>> T=300:100:1000;  h=100;
>> Cp=[40.82 48.10 55.09 61.25 66.60 71.26 75.33 78.90];
>> plot(T,Cp,'*'), axis([200 1200 0 80])
>> fill([T(end) T(end) T(1) T],[Cp(end) 0 0 Cp],'g')
>> axis([200 1200 0 80])
```

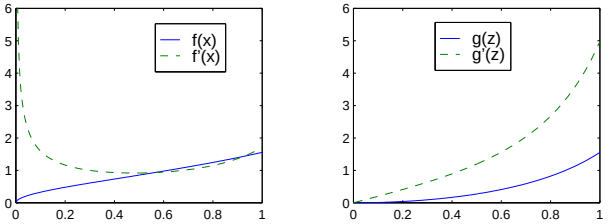



```
>> qp=h*(Cp(1)/2+sum(Cp(2:end-1))+Cp(end)/2)
qp =
       43749
>> Absfel=(T(end)-T(1))/12*max(abs(diff(Cp,2)))
Absfel =
   48.4167
```

Trapetsmetoden ger $\tilde{q}_p = 43749$ och trunkeringsfelet blir högst 48.4. Mätfel inverkar med en osäkerhet på 3.5 som i detta fall är närmast försumbart.

4. Skall beräkna $I = \int_0^1 \tan(\sqrt{x})\,dx$. Integranden $f(x) = \tan(\sqrt{x})$ har derivatan $f'(x) = \frac{1}{2\sqrt{x}}(1 + \tan^2(\sqrt{x}))$ som är singulär i $x = 0$.

Variabelsubstitution $\sqrt{x} = z \quad (dx = 2z\,dz)$ ger $I = 2\int_0^1 z\tan(z)\,dz$. Integranden $g(z) = z\tan(z)$ har derivatan $g'(z) = \tan(z) + z(1 + \tan^2(z))$ som är reguljär. Denna integral är mer lätthanterlig!



Vi använder trapetsmetoden på $\int_0^1 f(x)\,dx$

```
>> n=1; a=0; b=1; T0=0; itmax=20; tol=1e-5;
>> for i=1:itmax
       h=(b-a)/n; x=a+h*[0:n]'; f=tan(sqrt(x));
       T=h*(f(1)/2+sum(f(2:n))+f(n+1)/2);
       disp([sprintf('%7.0f',n),'  ',sprintf('%2.7f',T)])
       if abs(T-T0)<tol, break, end
       T0=T; n=2*n;
   end

      1  0.7787039
      2  0.8166071
      4  0.8388319
      8  0.8491718
     16  0.8534776
     32  0.8551661
osv.
   2048  0.8561744
```

Med 2048 funktionsberäkningar får vi resultatet. Om vi nu använder trapetsmetoden på $\int_0^1 g(z)\,dz$ får vi lika bra resultat med 512 funktionsberäkningar.

Behovet av variabelsubstitution finns även då vi använder färdig programvara, t.ex. **quad** eller **quadl** i MATLAB (feltoleransen sätts till 10^{-5})

```
>> f=inline('tan(sqrt(x))');
>> [q,fcnt]=quadl(f,0,1,1e-5)
q =
      0.85618
fcnt =
    78
```

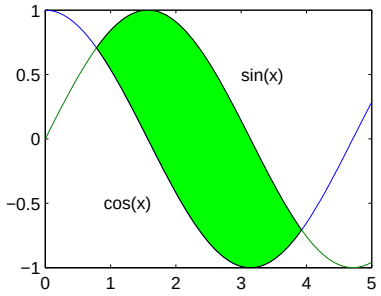
```
>> g=inline('2*z.*tan(z)');
>> [q,gcnt]=quadl(g,0,1,1e-5)
q =
      0.85618
gcnt =
    18
```

Algoritmerna är adaptiva, den gör funktionsberäkningar där de behövs som bäst. Det blir ganska stor skillnad i antal funktionsberäkningar även här.

5. Om vi låter $x = \sin(t)$ övergår integralen i $\int_0^{\pi/2} \sin(\sin(t))\,dt$ som beräknas till 1.7865 med t.ex. trapetsmetoden.

6. I figuren ser vi att vi skall fylla i från $x_a = \frac{\pi}{4}$ till $x_b = \frac{5\pi}{4}$, dessa x-värden uppfyller $\sin(x) = \cos(x)$.

```
>> x=linspace(0,5,250);
>> y=cos(x); z=sin(x);
>> plot(x,y,x,z)
>> text(0.9,-0.5,'cos(x)'), text(3,0.5,'sin(x)')
>> xa=pi/4; xb=5*pi/4;
>> xa_xb=linspace(xa,xb,200); xb_xa=fliplr(xa_xb);
>> hold on
>> fill([xa_xb xb_xa], ...
        [sin(xa_xb) cos(xb_xa)],'g')
>> hold off
>> arean=quad('sin(x)-cos(x)',xa,xb)
arean =
    2.8284
```

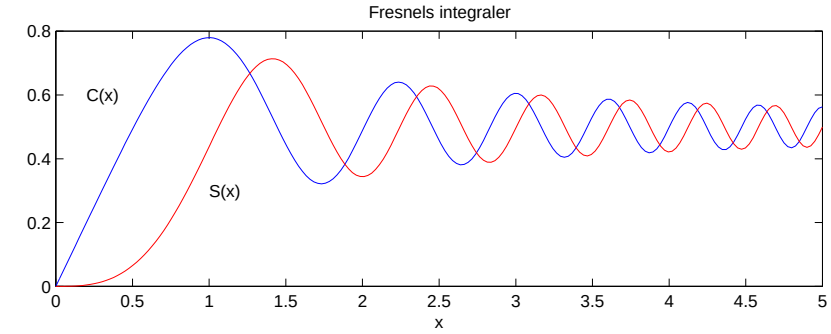


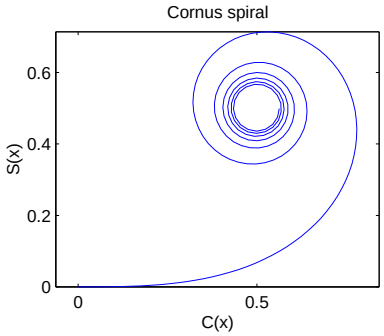
7. Vi skall rita upp Fresnelintegralerna

$$C(x) = \int_0^x \cos\left(\frac{\pi t^2}{2}\right) dt, \qquad S(x) = \int_0^x \sin\left(\frac{\pi t^2}{2}\right) dt$$

över intervallet $0 \le x \le 5$.

```
>> n=200;
>> x=linspace(0,5,n);
>> c=inline('cos(pi/2*t.^2)'); s=inline('sin(pi/2*t.^2)');
>> C=zeros(size(x)); S=zeros(size(x));
>> for k=2:n
      C(k)=C(k-1)+quad(c,x(k-1),x(k)); S(k)=S(k-1)+quad(s,x(k-1),x(k));
   end
>> subplot(2,1,1), plot(x,C,x,S,'r'), title('Fresnels integraler')
>> text(0.2,0.6,'C(x)'), text(1,0.3,'S(x)'), xlabel('x')
>> subplot(2,2,3), plot(C,S), axis('equal'), title('Cornus spiral')
>> xlabel('C(x)'), ylabel('S(x)')
```


Fresnels integraler
C(x)
S(x)
x
0
0.2
0.4
0.6
0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Cornus spiral
S(x)
C(x)
0
0.2
0.4
0.6
0
0.5

4 Begynnelsevärdesproblem.

I det här kapitlet skall vi studera numerisk lösning av begynnelsevärdesproblem för första ordningens differentialekvationer

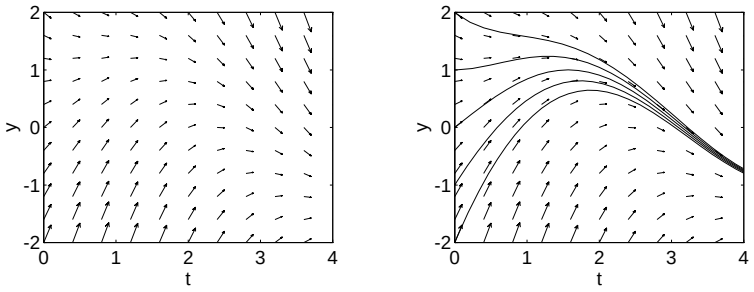
$$\begin{cases} y' = f(t,y), a \le t \le b, \\ y(a) = c, \end{cases} \tag{4.1}$$

där f är en känd funktion och c är en känd konstant. Den exakta lösningen till problemet är en funktion $t \mapsto y(t)$, vid tiden $t = a$ är funktionens värde känt och för $a < t < b$ är dess derivata känd.

Exempel 4.1. I den vänstra figuren nedan har vi ritat riktningsfältet till den skalära ekvationen,

$$y' = -y(t) + \sin(t) + \cos(t), 0 \le t \le 4,$$

och i den högra lösningskurvorna för några olika begynnelsevärden på $y(0)$.



Vi ser hur lösningskurvorna följer riktningsfältet.

Med MATLAB kan vi få fram en numerisk lösning[5] genom att använda funktionen **ode45**. Vi beskriver högerledet i differentialekvationen och beräknar sedan lösningen för t.ex. $y(0) = 1$

```
>> myrslok=inline('-y+sin(t)+cos(t)','t','y')

myrslok =
Inline function:
myrslok(t,y) = -y+sin(t)+cos(t)

>> y0=1; [t,y]=ode45(myrslok,0:0.1:4,y0);
>> plot(t,y)
```

Riktningsfältet kan vi rita med[6]

```
>> s=0.1;
>> for t=0:0.4:4
      for y=-2:0.4:2
         dt=1; dy=myrslok(t,y);              % I punkten (t,y) ritar vi ett
         plot([t,t+s*dt],[y,y+s*dy])         % litet streck i riktningen (1,y').
         hold on
      end
   end
```

■

Exempel 4.2. I detta exempel visar vi hur ett begynnelsevärdesproblem för en andra ordningens differentialekvation kan skrivas som ett system av första ordningens differentialekvationer, dvs. på formen (4.1). Tekniken kan direkt generaliseras till begynnelsevärdesproblem för högre ordningens differentialekvationer.

En bladfjäder är fastspänd i sin ena ända. Om man knäpper på den fria ändan kommer den att vibrera. Svängningen är inte harmonisk, den återställande kraften är $-kx^3$, där x är avvikelsen från jämviktsläget. På grund av luftmotståndet blir svängningen dämpad. Man tar hänsyn till detta genom en motståndsterm proportionell mot hastigheten. Detta ger rörelseekvationen

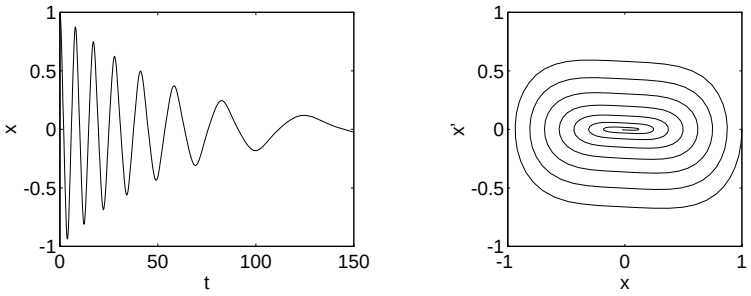
$$x'' = -\frac{k}{m}x^3 - \frac{\lambda}{m}x'.$$

[5] Till det här enkla problemet finns en formel för den exakta lösningen, $y(t) = c\exp(-t) + \sin(t)$, där $c = y(0)$.
[6] När vi lärt oss lite linjär algebra kan vi använda funktionen **quiver** för att enklare rita riktningsfält.

Genom att låta $x' = v$ kan vi skriva ekvationen som ett första ordningens system

$$\begin{cases} x' = v \\ v' = -\frac{k}{m}x^3 - \frac{\lambda}{m}v \end{cases}$$

För begynnelsevärdena $x(0) = 1$ och $v(0) = x'(0) = 0$ har vi i figuren nedan till vänster ritat x som funktion av t. I den högra figuren ser vi fasporträttet, dvs. vi ser kurvan $t \mapsto (x(t), x'(t))$, som man ofta ritar upp för sådana här problem. Vi har tagit $\frac{k}{m} = 1.0$ och $\frac{\lambda}{m} = 0.05$. Trots att ekvationen ser mycket enkel ut kan man inte få fram den exakta lösningen på sluten form utan vi har fått använda en numerisk metod för att approximera den.



■

Från matematiken vet vi att om f är kontinuerlig och uppfyller ett s.k. Lipschitz-villkor,

$$|f(t,u) - f(t,v)| \leq L|u - v|,$$

i strimman $a \leq t \leq b$, där L kallas Lipschitz-konstanten, så har problemet (4.1) en entydigt bestämd lösning $y(t)$ som är definierad på hela intervallet $a \leq t \leq b$. Vi skall se att Lipschitz-konstanten även har betydelse för de numeriska metoder som används för att lösa begynnelsevärdesproblem.

4.1 Några differensmetoder.

Program för lösning av problemet (4.1) stegar sig fram, utgående från $y(a) = c$ i $t = a$, genom att bestämma approximationer y_n av $y(t_n)$ på ett nät $a = t_0 < t_1 < \cdots < t_N = b$. I varje steg från t_n bestäms en steglängd h_n så att approximationen y_{n+1} i $t_{n+1} = t_n + h_n$ uppfyller vissa noggrannhetskrav.

Vi skall se hur man kan approximera differentialekvationen (4.1) med en differensekvation. Det som ligger närmast till hands är kanske att approximera derivatan $y'(t)$ med en differenskvot, t.ex. framåtdifferensen

$$y'(t) \approx D_+ y(t) = \frac{y(t+h) - y(t)}{h}$$

bakåtdifferensen

$$y'(t) \approx D_- y(t) = \frac{y(t) - y(t-h)}{h}$$

eller centraldifferensen

$$y'(t) \approx D_0 y(t) = \frac{y(t+h) - y(t-h)}{2h}$$

Låter vi y_n vara approximationen av den exakta lösningen $y(t_n)$ så leder dessa differensapproximationer till Euler framåtmetoden

$$y_{n+1} = y_n + hf(t_n, y_n),$$

Euler bakåtmetoden

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}),$$

respektive mittpunktsmetoden

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n).$$

Euler framåtmetoden kan vi tolka geometriskt; utgående från begynnelsevärdet följer metoden riktningsfältet med korta steg.

Ett annat sätt att göra en differensmetod som ligger nära till hands är att utgå från integralformuleringen av (4.1) och approximera integralen på något lämpligt sätt. Integration av differentialekvationen i (4.1) från t_n till $t_{n+1} = t_n + h$ ger

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt.$$

Om vi approximerar integralen med trapetsregeln så får vi den s.k. trapetsmetoden,

$$y_{n+1} = y_n + \frac{h}{2}\{f(t_n, y_n) + f(t_{n+1}, y_{n+1})\}.$$

Euler framåtmetoden, Euler bakåtmetoden och trapetsmetoden är exempel på enstegsmetoder, man behöver bara känna y_n för att beräkna y_{n+1}. Mittpunktsmetoden är en flerstegsmetod, man behöver känna y_{n-1} och y_n för att kunna beräkna y_{n+1}.

Euler framåtmetoden och mittpunktsmetoden är exempel på explicita metoder, vi stoppar bara in y_n respektive y_{n-1} och y_n i högerledet för att få fram y_{n+1}. Euler bakåtmetoden och trapetsmetoden kallas för implicita metoder, i varje steg måste vi i allmänhet lösa en ickelinjär ekvation i y_{n+1}.

En implicit metod kräver mer beräkningsarbete per steg än en explicit metod. För vissa typer av begynnelsevärdesproblem, s.k. styva problem, är implicita metoder lämpliga att använda, eftersom man då kan ta större steglängd än för explicita metoder. Totalt sett blir då en implicit metod mer effektiv än en explicit.

En differensmetod sägs vara konvergent om

$$y_n \to y(t) \text{ då } h \to 0,$$

där $n = \frac{t-a}{h}$, för alla $a < t < b$.

Metoderna ovan är alla konvergenta. Vi kommer i senare avsnitt se att det krävs mer än konvergens för att ge önskad noggrannhet till en rimlig beräkningskostnad.

Exempel 4.3 Vi visar nu att Euler framåtmetoden är konvergent: Låt $e_n = y(t_n) - y_n$. Taylorutveckling ger

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\tau_n) =$$

$$= y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\tau_n)$$

Om vi subtraherar $y_{n+1} = y_n + hf(t_n, y_n)$ får vi

$$e_{n+1} = e_n + h\{f(t_n, y(t_n)) - f(t_n, y_n)\} + \frac{h^2}{2}y''(\tau_n)$$

eller

$$|e_{n+1}| \leq (1 + hL)|e_n| + \frac{h^2}{2} \max_{a \leq \tau \leq b} |y''(\tau)|$$

där L är Lipschitz-konstanten. Med induktion får vi

$$|e_n| \leq (1 + hL)^n |e_0| + \left(\sum_{k=0}^{n-1} (1 + hL)^k \right) \frac{h^2}{2} \max_{a \leq \tau \leq b} |y''(\tau)|$$

Men nu är $|e_0| = 0$ och

$$\sum_{k=0}^{n-1} (1 + hL)^k = \frac{(1 + hL)^n - 1}{(1 + hL) - 1} \qquad \text{(geometriska seriens summa)}$$

Eftersom $(1 + hL)^n \leq \mathrm{e}^{Lnh} = \mathrm{e}^{L(t_n - a)}$ så har vi

$$|e_n| \leq \frac{h}{2L}(\mathrm{e}^{L(t_n - a)} - 1) \max_{a \leq \tau \leq b} |y''(\tau)|$$

och vi ser att $e_n \to 0$ då $h \to 0$. ■

Störningsresultat.

När t.ex. Eulers metod tar ett steg så gör den ett litet fel så att den inte ligger kvar på lösningskurvan vi vill följa utan hamnar på en lösningskurva tillhörande ett annat begynnelsevärde. Det bästa metoden kan göra i nästa steg är att försöka följa denna nyalösningskurva så bra som möjligt ett litet stycke.

Hur bra det kommer att gå beror både på metodens egenskaper och på problemets egenskaper. Om närliggande lösningskurvor dras samman med tiden så är problemet stabilt (eller välkonditionerat) och om de sprids ut kraftigt med tiden så är problemet instabilt (eller illakonditionerat).

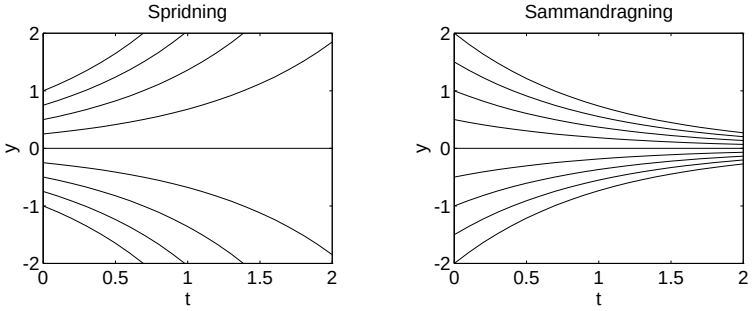
Vi behöver störningsresultat för att beskriva problemets egenskaper.

Om $y(t)$ och $u(t)$ är två lösningar till ekvationen i (4.1) med olika begynnelsevärden och om L är Lipschitz-konstanten så gäller att

$$|y(t) - u(t)| \leq e^{L(t-t_n)}|y(t_n) - u(t_n)| \tag{4.2}$$

för $a \leq t_n \leq t \leq b$.

Om $L(b-a)$ inte alltför stort visar (4.2) att problemet inte är alltför instabilt, och de klassiska metoder vi presenterar senare i kursen kommer fungera bra. Då $L(b-a)$ stor är antingen problemet kraftigt instabilt, dvs. kraftig spridning hos lösningskurvorna, och då kan vi inte göra mycket åt det, eller också är det kraftigt sammandragande, ett s.k. styvt problem, och då blir de klassiska metoderna ineffektiva och vi behöver speciella metoder för att kunna lösa dessa styva problem effektivt. Den klassiska Lipschitz-konstanten kan inte skilja de två senare fallen åt vilket den skalära s.k. testekvationen $y' = \lambda y$ visar.



I uppskattningen (4.2) gäller då likhet för $\lambda \geq 0$ (spridning), men för $\lambda < 0$ (sammandragning) får vi samma uppskattning som för $|\lambda|$. Anledningen är att i bägge fallen är Lipschitz-konstanten $L = |\lambda|$. Denna oförmåga hos Lipschitz-konstanten att skilja dessa fall åt har gjort att man fått söka andra begrepp, men vi går inte in på det i denna kursen.

4.2 Felspridning.

Program för lösning av begynnelsevärdesproblem använder inte konstant steglängd h. Man vill uppskatta felet i varje steg och justera h, steglängden för nästa steg, utifrån denna uppskattning. Ett för kort steg leder till onödiga beräkningar och därmed ineffektivitet, medan ett för långt steg leder till att man inte klarar av uppsatta krav på noggrannhet. Det fel som vanligen uppskattas av programmen är det så kallade lokala felet och inte det verkliga felet. Det är ju det senare felet en användare egentligen är intresserad av.

Globala och lokala fel.

Låt $y(t)$ beteckna den entydiga lösningen till problemet

$$\begin{cases} y' = f(t,y), a \leq t \leq b, \\ y(a) = c. \end{cases}$$

Med det verkliga eller globala felet i $t = t_n$ avser man

$$g_n = y(t_n) - y_n.$$

Tyvärr är det relativt svårt och kostsamt att uppskatta detta fel.

Antag att man har en approximation y_n av lösningen vid tiden $t = t_n$. Det man kan önska är att kunna noggrant följa den lösningskurva som passerar punkten (t_n, y_n) fram till tiden $t = t_{n+1}$.

Låt $u_n(t)$ vara lösningen till

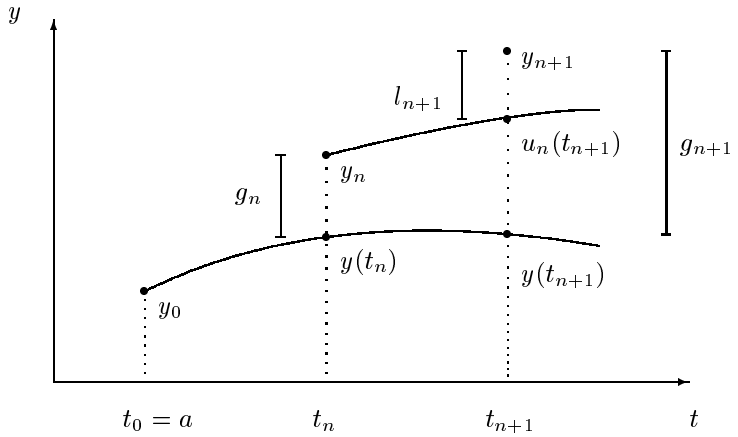
$$\begin{cases} u_n' = f(t, u_n), \\ u_n(t_n) = y_n. \end{cases} \tag{4.3}$$

Man brukar kalla $u_n(t)$ den lokala lösningen.

Med det lokala felet i $t = t_{n+1}$ avser man

$$l_{n+1} = u_n(t_{n+1}) - y_{n+1},$$

dvs. felet som blir då man försöker följa lösningen som utgår ifrån (t_n, y_n) genom att ta ett steg med metoden. Dessa fel illustreras i figuren.



Det är ett rimligt önskemål att den numeriska metoden håller de lokala felen små, vilken effekt dessa fel har på det globala felet beror på differentialekvationen. Det gäller att

$$g_{n+1} = y(t_{n+1}) - y_{n+1} = \{y(t_{n+1}) - u_n(t_{n+1})\} + \{u_n(t_{n+1}) - y_{n+1}\},$$

dvs.

$$g_{n+1} = \{y(t_{n+1}) - u_n(t_{n+1})\} + l_{n+1}. \tag{4.4}$$

Eftersom $y(t_n) - u_n(t_n) = g_n$ så är $y(t_{n+1}) - u_n(t_{n+1})$ det av differentialekvationen fortplantade globala felet från $t = t_n$, se figuren igen.

Det globala felet g_{n+1} är alltså summan av det från det tidigare steget fortplantade globala felet och det lokala felet l_{n+1}.

Exempel 4.4. Låt oss betrakta testekvationen $y' = \lambda y$, för en komplex konstant λ . Det gäller att

$$\begin{aligned} y(t) &= y(t_n)e^{\lambda(t-t_n)}, \\ u_n(t) &= y_n e^{\lambda(t-t_n)}, \end{aligned}$$

så

$$y(t_{n+1}) - u_n(t_{n+1}) = \{y(t_n) - y_n\}e^{\lambda(t_{n+1}-t_n)} = g_n e^{\lambda h_n}.$$

Från (4.4) får vi då

$$g_{n+1} = g_n e^{\lambda h_n} + l_{n+1}. \tag{4.5}$$

Om $Re(\lambda) > 0$ divergerar lösningskurvorna och ju större $Re(\lambda)$ är ju större spridning får vi. Från (4.5) ser vi att små lokala fel i varje steg inte medför små globala fel. Å andra sidan om $Re(\lambda) < 0$ så konvergerar lösningskurvorna och (4.5) visar att kontroll av de lokala felen ger kontroll av det globala felet.

Vi får en uppskattning av det globala felet m.h.a. de lokala felen genom upprepad användning av (4.5);

$$g_{n+1} = \sum_{j=1}^{n+1} e^{\lambda(t_{n+1}-t_j)} l_j.$$

■

En metod sägs ha noggrannhetsordning p om det lokala felet är $\mathcal{O}(h^{p+1})$, dvs. om det lokala felet är av ordning $p + 1$.

Exempel 4.5. För Euler framåtmetoden har vi

$$l_{n+1} = u_n(t_{n+1}) - y_{n+1} = u_n(t_{n+1}) - \{y_n + hf(t_n, y_n)\}.$$

Taylorutveckling ger

$$u_n(t_{n+1}) = u_n(t_n) + hu_n'(t_n) + \frac{h^2}{2}u_n''(\tau_n) =$$

$$= y_n + hf(t_n, y_n) + \frac{h^2}{2}u_n''(\tau_n),$$

så vi har

$$l_{n+1} = \frac{h^2}{2}u_n''(\tau_n),$$

dvs. Euler framåtmetoden är en första ordningens metod. Även Euler bakåt är en första ordningens metod, medan mittpunktsmetoden och trapetsmetoden är av andra ordningen. ■

Det lokala felet för en flerstegsmetod kommer att bero på hur metoden sprider tidigare lokala fel. Man måste kräva att metoden är stabil, dvs. att små störningar i begynnelsevärden och högerled i differensekvationen ger endast små störningar i dess lösning.

4.3 Stabilitet.

Trots att konvergens är nödvändigt för en metod skall vara användbar, så är det långt ifrån tillräckligt. Det är av avgörande betydelse hur approximationerna $\{y_n\}$ beter sig för steglängder som inte går mot noll. En acceptabel situation är att de beräknade y_n-värdena, under inverkan av lokala fel och avrundningsfel, har samma kvalitativa beteende som den exakta lösningen $y(t)$; dvs. att felen inte växer allt för mycket och förstör approximationen. Den verkliga noggrannheten hos $\{y_n\}$ bestäms av steglängden.

Exempel 4.6. Låt oss studera hur mittpunktsmetoden fungerar på testproblemet

$$\begin{cases} y' = \lambda y, t \geq 0 \\ y(0) = c. \end{cases}$$

Vi får

$$y_0 = c, \quad y_1 = ce^{h\lambda},$$
$$y_{n+1} = y_{n-1} + 2h\lambda y_n,$$

där y_0 och y_1 har givits den exakta lösningens värden.
Man kan visa att för lösningen till denna differensekvation gäller

$$y_n = \beta_1 e^{\lambda t_n}(1 + \mathcal{O}(h^2)) + \beta_2(-1)^n e^{-\lambda t_n}(1 + \mathcal{O}(h^2)),$$

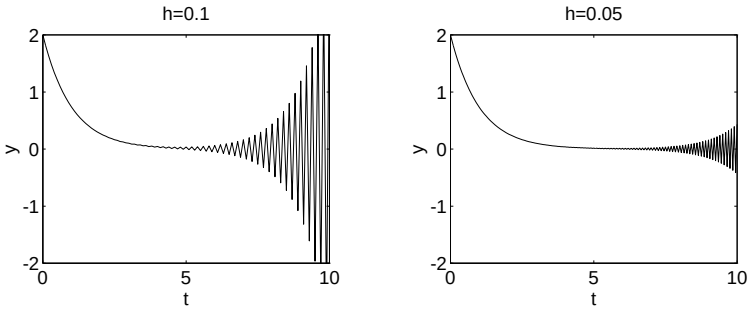
där

$$\beta_1 = c + \mathcal{O}(h^2\lambda^2),$$
$$\beta_2 = \mathcal{O}(h^3\lambda^3).$$

Vi ser att lösningen till differensekvationen konvergerar mot lösningen till differentialekvationen som $\mathcal{O}(h^2)$.

Det finns två parametrar β_1 och β_2 i lösningen till det diskreta problemet men i det kontinuerliga problemets lösning $y(t) = ce^{\lambda t}$ finns det bara en parameter. Den första termen i y_n approximerar den sökta lösningen $y(t)$ medan den andra termen är en s.k. parasitlösning, den har inget med det ursprungliga problemet att göra utan har skapats av diskretiseringen. Då $\lambda > 0$ är detta inget problem för parasiten dör bort med tiden, men då $\lambda < 0$ blir det problem för då växer parasiten exponentiellt med tiden. Teckenväxlingen i parasiten gör att den diskreta lösningen kommer att oscillera.

Figuren nedan visar hur det ser ut med $\lambda = -1$ och $c = 2$. Till vänster tog vi steglängden $h = 0.1$ och till höger $h = 0.05$. Förr eller senare slår parasiten igenom och förstör approximationen, men vi kan flytta den tidpunkten framåt genom att välja h allt mindre.



Mittpunktsmetoden är visserligen konvergent, men vi ser att det inte räcker för att få ett tillfredsställande uppförande. ■

Vi behöver analysera hur metoderna beter sig för fixa steglängder, som inte går mot noll. I allmännhet kan denna analys bara utföras för väldigt små klasser av ekvationer, den enklaste är just testekvationen

$$y' = \lambda y, t \geq 0. \tag{4.6}$$

För $Re(\lambda) \leq 0$ är alla lösningar till (4.6) begränsade så det vore naturligt att kräva att följden $\{y_n\}$ också skulle vara begränsad. Analysen förenklas om man bara betraktar konstanta steglängder h.

En metod kallas absolut stabil om den för givna h och λ ger begränsade approximationer då den används på (4.6). Mängden i komplexa planet av alla $h\lambda$ för vilka detta gäller kallas metodens absoluta stabilitetsområde.

Exempel 4.7. Om vi använder Euler framåtmetoden på (4.6) får vi

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n$$

och vi ser att $\{y_n\}$ är begränsad om och endast om $|1 + h\lambda| \leq 1$, dvs. om och endast om $h\lambda$ ligger på cirkelskivan med radien 1 och centrum i $z = -1$.

Med Euler bakåtmetoden på (4.6) får vi

$$y_{n+1} = y_n + h\lambda y_{n+1},$$

dvs.

$$y_{n+1} = \frac{1}{1 - h\lambda} y_n,$$

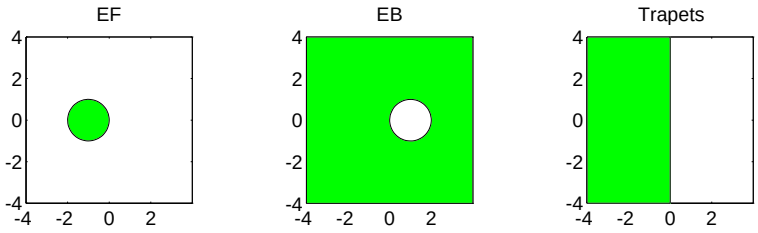
och $\{y_n\}$ är begränsad om och endast om $|1-h\lambda| = 1$, dvs. om och endast om $h\lambda$ ligger utanför cirkelskivan med radien 1 och centrum i $z = 1$.

Mittpunktsmetodens stabilitetsområde är endast ett litet intervall på imaginära axeln ($|Im(h\lambda)| \leq 1$). För trapetsmetoden får vi

$$y_{n+1} = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} y_n,$$

och stabilitetsområdet är vänstra halvplanet, $Re(h\lambda) \leq 0$.

I figuren ser vi stabilitetsområdena (gråtonade) för Euler framåt, Euler bakåt och trapets metoderna.



■

En metod vars stabilitetsområde innehåller $h\lambda = 0$ kallas nollstabil eller bara stabil och om stabilitetsområdet innehåller vänstra halvplanet, $Re(h\lambda) \leq 0$, så kallas den A-stabil . (En sådan metod har samma stabilitetsegenskaper som det kontinuerliga problemet i fallet $y' = \lambda y$.)

4.4 Användning av ode45.

För att få effektiva metoder måste man variera steglängden så att den anpassar sig efter bl.a. variation i lösningen. Funktionen **ode45** i MATLAB är ett exempel på sådana metoder. Den bygger på ett s.k. Runge-Kutta-Fehlberg par, en 4:e ordningens metod och en 5:e ordningens. Man använder skillnaden mellan metoderna för att skatta det lokala felet.

Vi skall se hur man får fram lösningen till ekvationen i exempel 4.2. Först lagrar vi en beskrivning av högerledet i differentialekvationen (efter omskrivning till system) på en fil `blad.m`:

```
function f=blad(t,y)
f=[y(2);-1*y(1)^3-0.05*y(2)];
```

I MATLAB tar vi och sätter begynnelsetid $t_0 = 0$, sluttid $t_{slut} = 150$ och begynnelsevärdet $y_0 = [1, 0]^T$.

```
>> t0=0; tslut=150;
>> y0=[1;0];
```

Sedan tar vi och låter **ode45** räkna fram lösningen med kommandot

```
>> [t,y]=ode45(@blad,t0,tslut,y0);
```

Nu är `t` en vektor i MATLAB med de tidpunkter (mellan t_0 och t_{slut}) där approximationer har räknats fram och `y` är en matris med två kolonner, den första kolonnen innehåller x-värdena (läget vid de olika tidpunkterna) och den andra innehåller x'-värdena (hastigheten vid de olika tidpunkterna). Vi ritar upp lösningen $t \mapsto (t, x(t))$ med

```
>> subplot(2,2,1)
>> plot(t,y(:,1))
>> axis([0 150 -1 1])
>> xlabel('t'), ylabel('x')
```

och fasporträttet $t \mapsto (x(t), x'(t))$ med

```
>> subplot(2,2,2)
>> plot(y(:,1),y(:,2))
>> axis([-1 1 -1 1]), axis('square')          % Kvadratiskt koordinat system.
>> xlabel('x'), ylabel('x'''')
```

Så fick vi alltså fram bilderna i exempel 4.2.

4.5 Övningar.

1. Räkna fram approximationer av $y(0.8)$, där $y(t)$ är lösningen till problemet

$$\begin{cases} y' = 1 + t\,y^2 \\ y(0) = 0 \end{cases}$$

   Använd Eulers metod med stegen 0.2 och 0.1.

   Vilken approximationsordning tycks metoden ha?

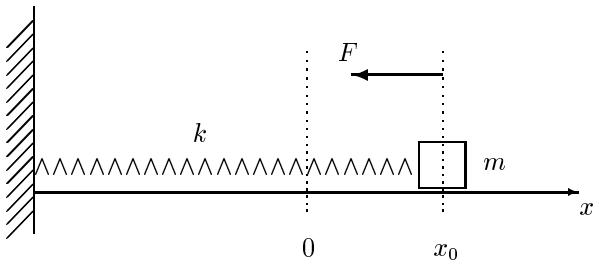
2. En klotformig vattentank med radien R töms på vatten genom att ett hål med radien r öppnas i botten. I toppen på tanken finns ett öppet lufthål. Vattenståndet H ges som lösningen till differentialekvationen

$$H' = \frac{C}{(2R - H)\sqrt{H}},\ C = -r^2\sqrt{2g},$$

   med begynnelsevillkoret $H(0) = H_0$. Låt $H_0 = 3$ m, $R = 3$ m, $r = 0.4$ m, $g = 9.81$ m/s^2.

   Bestäm $H(t)$ numeriskt med **ode45**.

3. Antag att vi har en partikel med massan $m = 0.1$ kg som är fastsatt i en fjäder med fjäderkonstanten $k = 0.12$ N/m. Detta innebär att kraften då fjädern sträckts en sträcka x blir $-kx$. Fjäderns andra ända är fastsatt i en fix punkt och partikeln kan röra sig utan friktion längs en horisontell linje.



Vid tiden $t = 0$ släpps partikeln från vila, på avståndet $x_0 = 0.1$ m från jämviktspunkten, och man vill förutsäga den fortsatta rörelsen.

Inför ett koordinatsystem enligt figuren ovan med origo där partikeln har sitt jämviktsläge. Då x betecknar partikelns läge får vi rörelseekvationen $x'' = -\frac{k}{m}x$.

Vi kan skriva om ekvationen som ett begynnelsevärdesproblem för ett första ordningens system

$$\begin{cases} x' = v \\ v' = -\frac{k}{m}x \end{cases}$$

med begynnelsevillkoren $x(0) = x_0$, $v(0) = 0$.

Lös problemet numeriskt. Rita en bild av lösningen under 30 sekunder.

4.6 Lösningar.

1. Vi vill beräkna en approximation av $y(0.8)$. (Exakt gäller $y(0.8) = 0.919965461\cdots$) Med $f(t, y) = 1 + ty^2$ blir Euler framåtmetoden;

$$y_{n+1} = y_n + hf(t_n, y_n) = y_n + h(1 + t_n y_n^2)$$

$$h = 0.2 \left| \begin{array}{ll} t_0 = 0 & y_0 = 0 \\ t_1 = 0.2 & y_1 = y_0 + h(1 + t_0 y_0^2) = 0 + 0.2 \cdot (1 + 0 \cdot 0^2) = 0.2 \\ t_2 = 0.4 & y_2 = y_1 + h(1 + t_1 y_1^2) = 0.2 + 0.2 \cdot (1 + 0.2 \cdot 0.2^2) = 0.4016 \\ t_3 = 0.6 & y_3 = y_2 + h(1 + t_2 y_2^2) = 0.4016 + 0.2 \cdot (1 + 0.4 \cdot 0.4016^2) = 0.6145 \\ t_4 = 0.8 & y_4 = y_3 + h(1 + t_3 y_3^2) = 0.6145 + 0.2 \cdot (1 + 0.6 \cdot 0.6145^2) = 0.8598 \end{array} \right.$$

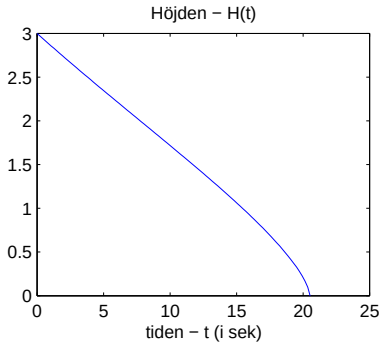
$y(0.8; 0.2) = 0.8598$ med felet $0.0601\cdots$

$$h = 0.1 \left| \begin{array}{ll} t_0 = 0 & y_0 = 0 \\ t_1 = 0.1 & y_1 = y_0 + h(1 + t_0 y_0^2) = 0 + 0.1 \cdot (1 + 0 \cdot 0^2) = 0.1 \\ t_2 = 0.2 & y_2 = y_1 + h(1 + t_1 y_1^2) = 0.1 + 0.1 \cdot (1 + 0.1 \cdot 0.1^2) = 0.2001 \\ t_3 = 0.3 & y_3 = y_2 + h(1 + t_2 y_2^2) = 0.2001 + 0.1 \cdot (1 + 0.2 \cdot 0.2001^2) = 0.3009 \\ \quad \vdots & \quad \vdots \\ t_8 = 0.8 & y_8 = y_7 + h(1 + t_7 y_7^2) = \cdots = 0.8854 \end{array} \right.$$

$y(0.8; 0.1) = 0.8854$ med felet $0.0345\cdots$

Eftersom felet ungefär halveras då vi halverar steglängden så bekrätas att Euler framåtmetoden är av 1:a ordningen.

2.
```
>> f=inline('C/(2*R-H)/sqrt(H)','t','H', ...
            'dummy','C','R')
f =
     Inline function:
     f(t,H,dummy,C,R) = C/(2*R-H)/sqrt(H)
>> H0=3; R=3; r=0.4; g=9.81; C=-r^2*sqrt(2*g);
>> tmax=21;
>> tidsint=linspace(0,tmax,100);
>> [t,H]=ode45(f,tidsint,H0,[],C,R);
>> plot(t,H)
```



3. Vi inför ett koordinatsystem med origo där partikeln har sitt jämviktsläge. Då x betecknar partikelns läge får vi rörelseekvationen $x'' = -\frac{k}{m}x$ med $x(0) = x_0$ och $x'(0) = 0$.

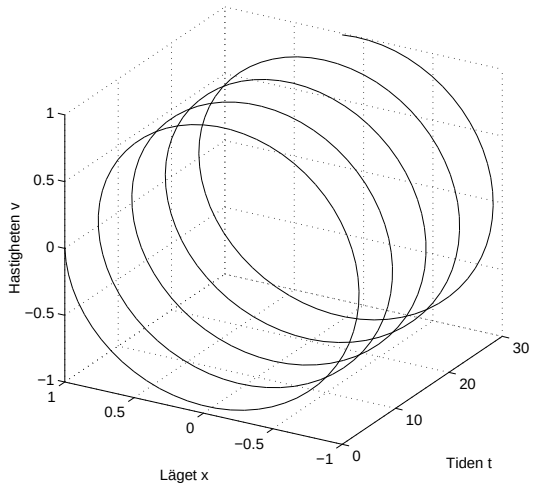
   Vi skriver om till system

$$\begin{cases} x' = v, \ x(0) = x_0 \\ v' = -\frac{k}{m}x, \ v(0) = 0 \end{cases}$$

   och löser med **ode45**.

```
>> dudt=inline('[u(2);-k*u(1)/m]','t','u', ...
                'dummy','k','m');
>> k=0.12; m=0.1; tmax=30; u0=[0.1;0];
>> tidsint=linspace(0,tmax,100);
>> [t,u]=ode45(dudt,tidsint,u0,[],k,m);

>> plot3(t,u(:,1),u(:,2)), axis('square')
```



Appendix A - Interpolation.

Låt f vara en skalär funktion i en variabel, dvs. $f : \mathbb{R} \to \mathbb{R}$. Antag att vi har $n+1$ olika punkter $x_0, x_1, \cdots, x_n$ med motsvarande funktionsvärden $f_i = f(x_i)$, $i = 0, 1, \cdots, n$, och vill bestämma en lämplig funktion P som interpolerar dessa värden, dvs. som uppfyller

$$P(x_i) = f_i, i = 0, 1, \cdots, n.$$

Vanligt är att använda ett polynom av grad n eller styckvisa polynom, s.k. spline-interpolation.

Polynominterpolation.

Interpolationspolynomet P_n av grad n är entydigt bestämt, men man kan ansätta polynomet på olika sätt. En ofta praktisk ansats är Newtons ansats

$$P_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots$$

$$+c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}).$$

Interpolationskraven $P_n(x_i) = f_i, i = 0, 1, \cdots, n$ ger

$$\begin{gathered} P_n(x_0) = c_0 = f_0, \\ P_n(x_1) = c_0 + c_1(x_1 - x_0) = f_1, \\ \vdots \\ P_n(x_n) = c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \cdots \\ +c_n(x_n - x_0)(x_n - x_1) \cdots (x_n - x_{n-1}) = f_n. \end{gathered}$$

Detta är ett nedåt triangulärt linjärt ekvationssystem med koefficienterna $c_0, c_1, \cdots, c_n$ som obekanta. Man löser lätt detta system genom framåtsubstitution.

En annan ansats är Lagranges ansats

$$P_n(x) = \sum_{i=1}^{n} f_i \, l_i(x), \quad l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

Vi observerar att $l_i(x_j) = 1$, om $i = j$, annars 0. För $n = 1$ får vi

$$P_1(x) = f_0 \, \frac{x - x_1}{x_0 - x_1} + f_1 \, \frac{x - x_0}{x_1 - x_0}$$

och för $n = 2$ får vi

$$P_2(x) = f_0 \, \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + f_1 \, \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f_2 \, \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Nu skall vi se hur väl $f(x)$ approximeras av $P_n(x)$.

Om $f(x)$ har $n+1$ kontinuerliga derivator så gäller för interpolationsfelet;

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n)$$

med $\xi(x)$ någonstans mellan $x_0, x_1, \cdots, x_n$ och x.

Förklaring: Låt $e(x) = f(x) - P_n(x)$ och $g(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$ och bilda $w(t) = e(x)g(t) - e(t)g(x)$.

Då gäller $w(x) = 0$ och $w(x_i) = 0, i = 0, 1, \cdots, n$ eftersom $g(x_i) = e(x_i) = 0$.
Alltså har w de $n+2$ nollställena $x_0, x_1, \cdots, x_n, x$ och upprepad användning av Rolles sats ger att $w^{(n+1)}(\xi(x)) = 0$ för något $\xi(x)$ mellan $x_0, x_1, \cdots, x_n$ och x.

Nu har vi $w^{(n+1)}(t) = e(x)g^{(n+1)}(t) - e^{(n+1)}(t)g(x)$, men $e^{(n+1)}(t) = f^{(n+1)}(t)$ och $g^{(n+1)}(t) = (n+1)!$ för alla t så

$$0 = w^{(n+1)}(\xi(x)) = e(x)g^{(n+1)}(\xi(x)) - e^{(n+1)}(\xi(x))g(x) =$$

$$= (n+1)!e(x) - f^{(n+1)}(\xi(x))g(x)$$

och resultatet följer. ■

Interpolation av vissa funktioner med polynom av högt gradtal kan uppföra sig illa. Felet kan bli mycket stort vilket man kan se om man interpolerar funktionen

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

på intervallet $-1 \leq x \leq 1$, med polynom av högt gradtal och ekvidistanta interpolationspunkter. I figuren nedan ser vi resultatet för $n = 3, 4, 5, 8$, ökar man gradtalet så blir resultatet bara ännu sämre. Det som inträffar brukar kallas Runges fenomen.