

# Tillämpningar i matematik med MATLAB

## 1 Inledning

Vi skall se på några tillämpningar av matematik som kräver beräkningsmetoder och dator. För detta är MATLAB rätt verktyg. I kursen ”Linjär algebra och numerisk analys” i läsperiod 4 kommer ni lära er mer om de beräkningsmetoder som MATLAB använder.

## 2 Nollställen

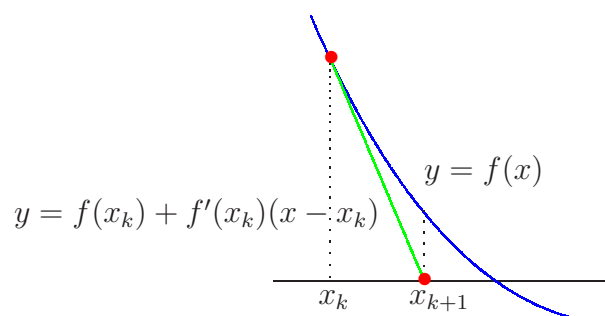
En lösning  $x^*$  till en ekvation  $f(x) = 0$  kallas ett nollställe eller en rot. Om vi inte kan få fram någon formel för att lösa en ekvation så kan vi beräkna en approximation med t.ex. **Newtons metod**: Antag att  $x_k$  är en approximation av ett nollställe till ekvationen  $f(x) = 0$ . Följ tangenten i punkten  $(x_k, f(x_k))$ , dvs.

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

ned till  $x$ -axeln ( $y = 0$ ) och tag skärningspunktens  $x$ -koordinat

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

som en ny approximation av nollstället.



Läs gärna mer om Newtons metod (eller Newton-Raphsons metod) i Persson-Böiers avsnitt 4.5.

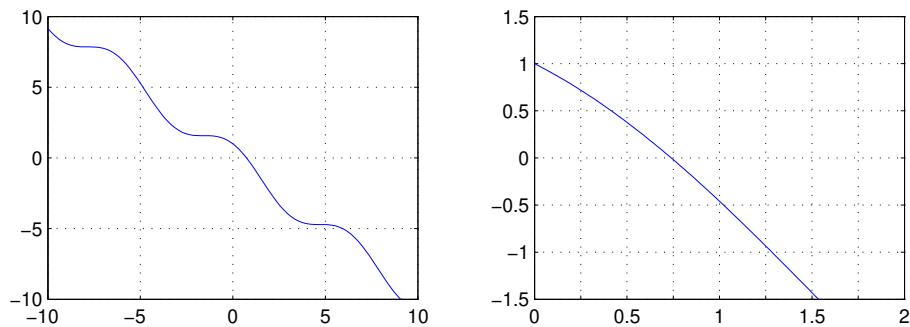
**Exempel 1.** Vi skall använda Newtons metod för att beräkna nollställena till funktionen

$$f(x) = \cos(x) - x$$

Vi börjar med att rita en graf av funktionen så att vi ser hur många nollställen vi har och ungefär var de ligger.

Vi beskriver funktionen och ritat grafen med

```
>> f=@(x)cos(x)-x;  
>> x=linspace(-10,10);  
>> plot(x,f(x))
```



Från grafen till höger, som är över ett lite mindre intervall, ser vi att vi har ett nollställe nära  $x_0 = 0.75$  som vi tar som startapproximation för Newtons metod.

```
>> Df=@(x)-sin(x)-1;  
>> x=0.75;  
>> kmax=10; tol=0.5e-8;  
>> for k=1:kmax  
    h=-f(x)/Df(x);  
    x=x+h;  
    disp([x h])  
    if abs(h)<tol, break, end  
end  
  
0.739111138752579 -0.010888861247421  
0.739085133364485 -0.000026005388094  
0.739085133215161 -0.000000000149324
```

I kolumnen till vänster ser vi  $x_k$ -värdena och i den till höger ser vi motsvarande  $f(x_k)$ -värden. Vi ser att vi får snabb konvergens, dvs. vi får snabbt ett noggrant resultat. Iterationen avbryts eftersom vi har mer än åtta korrekta decimaler (felet mindre än  $\frac{1}{2} \times 10^{-8}$ ).

**Uppgift 1.** Låt  $f(x) = x^3 - \cos(4x)$ . Lös ekvationen  $f(x) = 0$ . Rita upp grafen till  $f$  för att se var ungefär lösningarna (skärningspunkterna) ligger. Hur många lösningar finns det? Läs av i grafiken en första approximation av en lösning för att sedan förbättra denna med Newtons metod. Rita ut lösningen med en liten ring. Upprepa tills du beräknat alla lösningar till ekvationen.

## Färdiga program i MATLAB

Verktyslådan OPTIMIZATION TOOLBOX i MATLAB har en funktion `fzero` som finner nollställe till en given funktion och används enligt något av alternativen

```
x=fzero(fun,x0)      x=fzero(fun,x0,opts)
```

där `fun` beskriver funktionen vi skall finna nollstället till, `x0` är en första approximation av nollstället eller ett intervall med teckenväxling som omsluter nollstället vi söker. I senare fallet kommer `fzero` garanterat finna en approximation av ett nollställe.

Alternativet med `opts` använder vi då vi t.ex. vill ange hur noggrant lösningen skall beräknas. Man skapar vektorn `opts` med funktionen `optimset`, se hjälptexten.

Om vi skulle använda `fzero` för att beräkna ett av nollställena i uppgift 1 skulle det kunna se ut så här

```
>> f=@(x)x.^3-cos(4*x);
>> x0=-1;
>> x=fzero(f,x0)
```

**Exempel 2.** Kastbana utan luftmotstånd ("Mer om funktioner och grafik i MATLAB" exempel 4) beskrivs av

$$y(x) = y_0 - \frac{g}{2v_0^2 \cos^2(\theta)} \left( x - \frac{v_0^2 \sin(2\theta)}{2g} \right)^2 + \frac{v_0^2 \sin^2(\theta)}{2g}$$

där  $y_0$  är utkasthöjden,  $v_0$  är utkastfarten och  $\theta$  är utkastvinkeln.

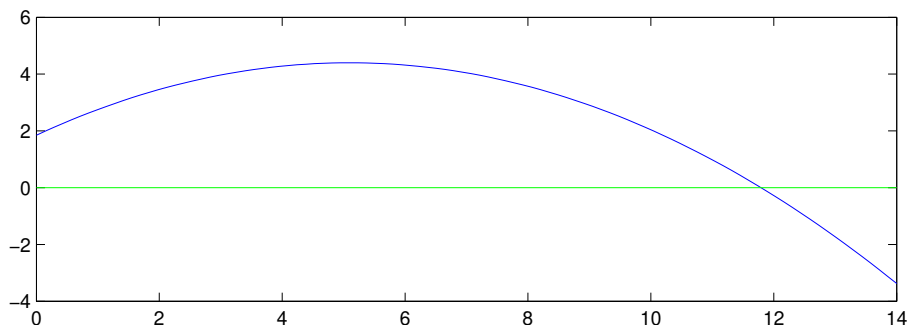
Hur långt når kastet om vi tar  $y_0 = 1.85$ ,  $v_0 = 10$  m/s och  $\theta = 45^\circ$ ?

Vi beskriver kastbanan med en funktion

```
function y=kast(x,y0,v,t)
    g=9.81;
    a=g/(2*v^2*cos(t)^2); b=v^2*sin(2*t)/(2*g); c=v^2*sin(t)^2/(2*g);
    y=y0-a*(x-b).^2+c;
```

ritar graf med

```
>> y0=1.85; v0=10; t=45*pi/180;
>> x=linspace(0,14);
>> plot(x,kast(x,y0,v0,t),[x(1) x(end)],[0 0], 'g')
```



och löser  $y(x) = 0$  med

```
>> x=fzero(@(x)kast(x,y0,v0,t), [11, 12])
x =
    11.7928
```

Lägg märke till hur vi använder ett anonymt funktionshandtag för att få med parametervärden. Det är bara `x` som `fzero` skall arbeta med.

**Uppgift 2.** Rita den slutna kurva som i polära koordinater ges av

$$r(\theta) = \frac{2 + \sin(3\theta)}{\sqrt{1 + \exp(\cos(\theta))}}$$

För vilka vinklar skär kurvan enhetscirkeln?

Ledning: Använd polära koordinater. Rita cirkeln och kurvan med `plot`. Läs in startvärden på vinklar med `ginput` och beräkna sedan vinklarna noggrant med `fzero`.

### 3 Optimering

Vill vi minimera en funktion  $f(x)$  som är deriverbar så söker vi lokala minpunkter, t.ex. genom att lösa ekvationen  $g(x) = f'(x) = 0$ . Sedan får vi kontrollera om dessa är lokala eller globala minimum. Läs gärna Persson-Böiers avsnitt 4.2. Vill vi maximera funktionen så finner vi maxpunkt genom att minimera  $h(x) = -f(x)$ .

Det finns också andra metoder som söker lokalt minimum till en funktionen. Vi beskriver inte dessa nu utan nöjer oss med att se vad MATLAB har för färdiga program.

#### Färdiga program i MATLAB

Verktyslådan OPTIMIZATION TOOLBOX i MATLAB har en funktion `fminbnd` som finner lokal minpunkt till en given funktion och används enligt något av alternativen

```
x=fminbnd(fun,x1,x2)      x=fminbnd(fun,x1,x2,opts)
```

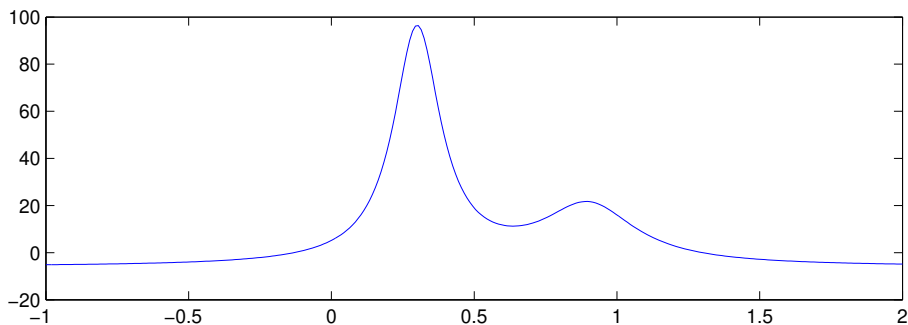
där `fun` beskriver funktionen vi skall minimera, `x1` och `x2` ger ett intervall som omsluter minpunkten vi söker. Alternativet med `opts` använder vi då vi t.ex. vill ange hur noggrant lösningen skall beräknas. Man skapar vektorn `opts` med funktionen `optimset`, se hjälptexten.

**Exempel 3.** Vi söker lokala min- och maxpunkter till funktionen

$$f(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

Vi börjar med att beskriva funktionen och rita dess graf med (lägg märke till de elementvisa operationerna)

```
>> f=@(x)1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;  
>> x=linspace(-1,2,300);  
>> plot(x,f(x))
```



Vi bestämmer den lokala minpunkten enligt

```
>> x=fminbnd(f,0.5,0.8)  
x =  
    0.6370
```

Vi bestämmer sedan de lokala maxpunkterna, genom att minimera  $h(x) = -f(x)$ , med

```
>> h=@(x)-f(x);  
>> x=fminbnd(h,0,0.5)  
x =  
    0.3004
```

På motsvarande sätt bestämmer vi den högra maximipunkten.

**Uppgift 3.** I samband med studiet av diffraktion vill man söka lokala maximum till funktionen

$$y(u) = \frac{\sin^2(u)}{u^2}$$

Hur många lokala maximipunkter finns det? Bestäm den minsta (positiva) lokala maximipunkten och motsvarande lokala maximivärde.

## 4 Integraler

Ibland kan man inte bestämma integraler exakt utan man får nöja sig med att beräkna approximationer. T.ex.  $\int_0^1 \exp(x^2) dx$  kan inte beräknas exakt, eftersom det inte finns någon användbar primitiv funktion. Det kan också vara så att integranden bara är känd i vissa punkter, t.ex. vi har en serie med mätdata.

Den geometriska tolkningen av integralen  $\int_a^b f(x) dx$  är arean av ytan mellan grafen av integranden  $y = f(x)$  och  $x$ -axeln, dvs.  $y = 0$ , mellan  $x = a$  och  $x = b$ .

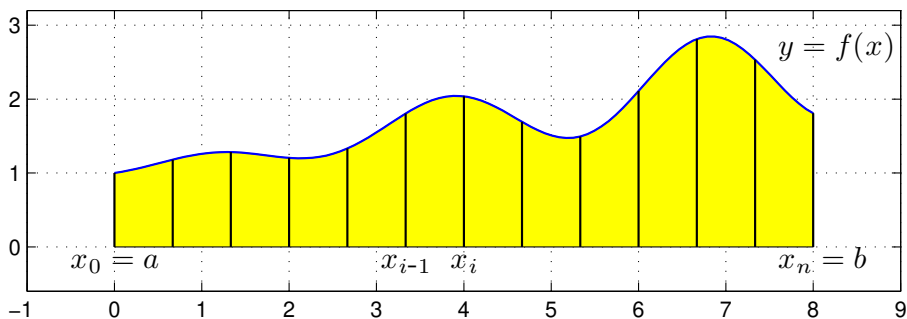
Vi gör en likformig indelning av intervallet  $a \leq x \leq b$

$$a = x_0 < x_1 < x_2 < \dots < x_{i-1} < x_i < \dots < x_{n-1} < x_n = b$$

så att vi får  $n$  lika långa delintervall  $x_{i-1} \leq x \leq x_i$  med bredden  $h = \frac{b-a}{n}$ .

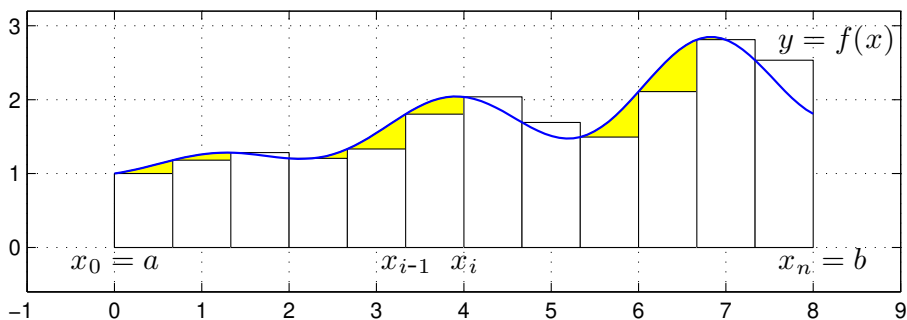
Sedan delar vi upp integralen i en summa av delintegraler över varje delintervall

$$\int_a^b f(x) dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} f(x) dx$$



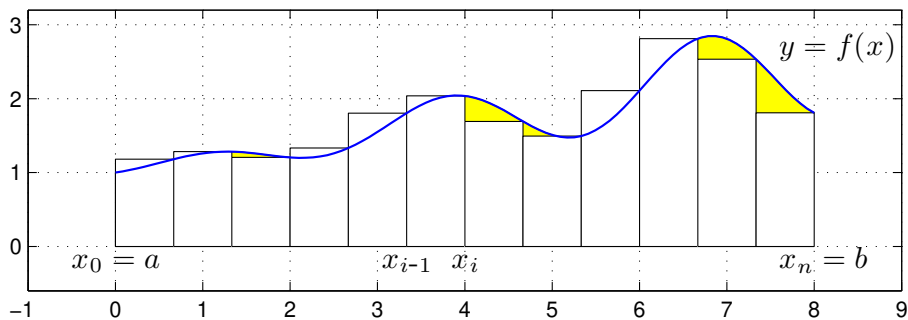
Om vi approximerar  $f(x)$  med  $f(x_{i-1})$  i intervallen  $x_{i-1} \leq x \leq x_i$  får vi **vänster rektangelregel**

$$\int_a^b f(x) dx \approx L_n = \sum_{i=1}^n h f(x_{i-1})$$



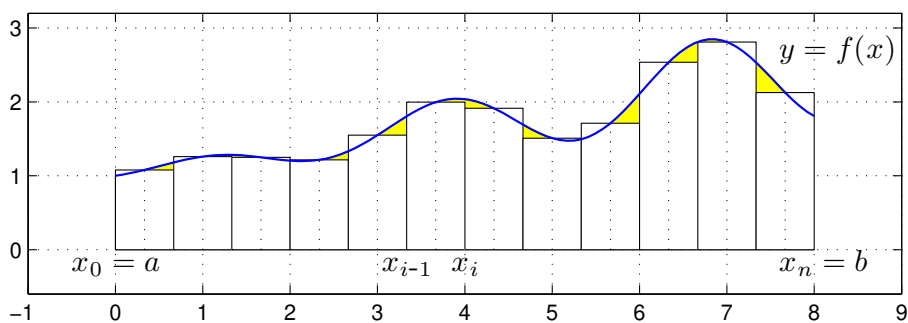
Om vi approximerar  $f(x)$  med  $f(x_i)$  i intervallen  $x_{i-1} \leq x \leq x_i$  får vi **höger rektangelregel**

$$\int_a^b f(x) dx \approx R_n = \sum_{i=1}^n h f(x_i)$$



Om vi approximerar  $f(x)$  med  $f(m_i)$  i intervallen  $x_{i-1} \leq x \leq x_i$ , där  $m_i$  är mittpunkterna i intervallen, får vi **mittpunktsmetoden**

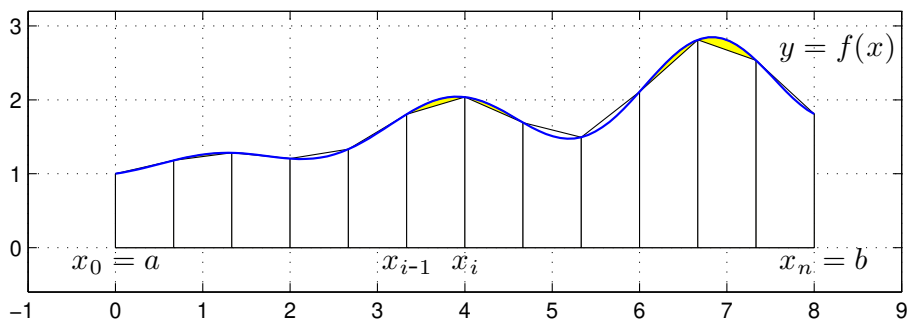
$$\int_a^b f(x) dx \approx M_n = \sum_{i=1}^n h f\left(\frac{x_{i-1} + x_i}{2}\right)$$



Dessa olika varianter är Riemannsummor (med  $\xi_i = x_{i-1}$ ,  $\xi_i = x_i$  respektive  $\xi_i = m_i$ ), se Persson-Böiers avsnitt 6.2.

Vi kan också approximera integralen med medelvärdet av vänster och höger rektangelregel och får då **trapetsmetoden**

$$\int_a^b f(x) dx \approx T_n = \sum_{i=1}^n \frac{h}{2} (f(x_{i-1}) + f(x_i))$$



Namnet på denna metod kommer från de parallelltrapetser som approximerar i varje delintervall.

Antag att vi vill beräkna  $\int_0^1 x \sin(x) dx$  med vänster rektangelregel med  $n = 100$ .

Vi skulle kunna göra så här

```
>> n=100;
>> a=0; b=1; f=@(x)x.*sin(x);
>> h=(b-a)/n
>> q=0;
>> for i=0:n-1
        x=a+i*h;
        q=q+h*f(x);
    end
>> q
```

Att använda en for-sats är oftast inte effektivt i MATLAB. Vi genererar hellre en vektor av alla funktionsvärden  $f(x_i)$  och sedan summerar dessa enligt

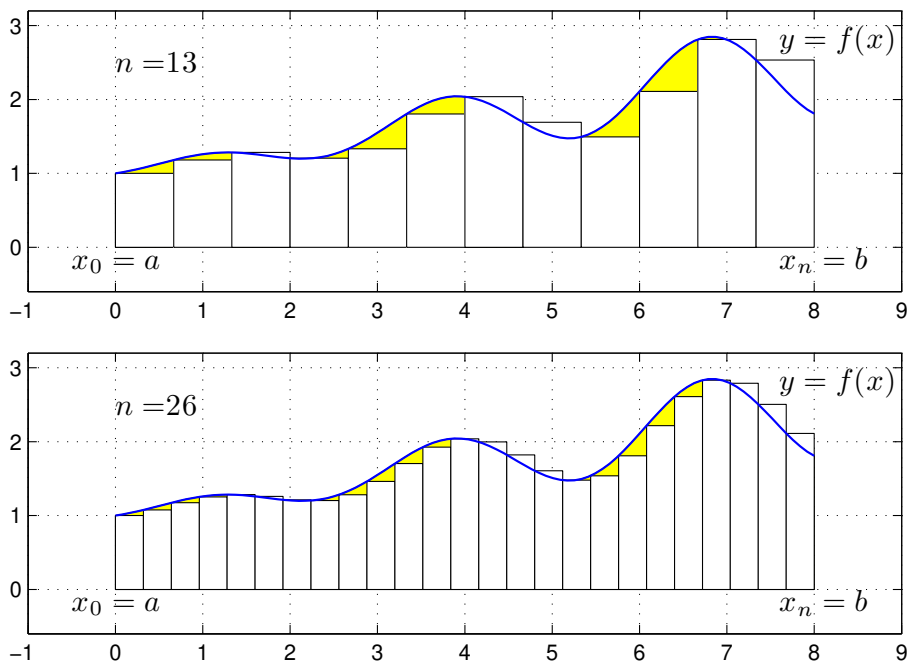
```
>> n=100;
>> a=0; b=1; f=@(x)x.*sin(x);
>> x=linspace(a,b,n+1);
>> h=(b-a)/n;
>> q=sum(h*f(x(1:n)))
```

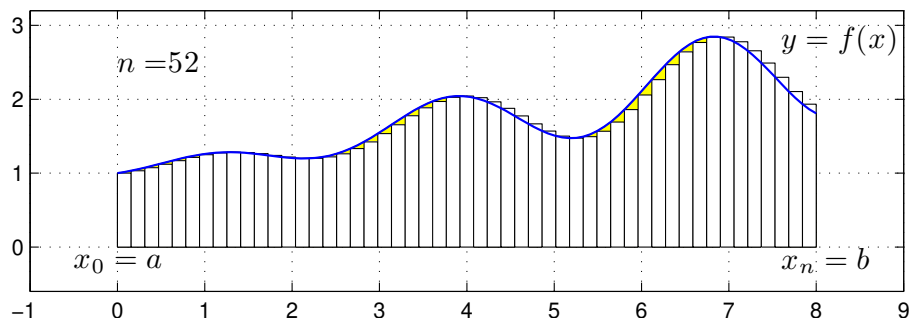
Detta sätt att organisera en beräkning kallas att vektorisera den, dvs. man genererar först en eller flera vektorer och utför sedan den önskade beräkningen på dem. De elementvisa operationerna `.*` `./` `.^` är exempel på vektoriserade operationer. Med funktionen `sum` summeras vektorn.

**Uppgift 4.** Beräkna en approximation av integralen  $\int_0^1 x \sin(x) dx$  med vänster och höger rektangelregel samt mittpunkts- och trapetsreglerna. Använd `sum`.

För metoderna ovan gäller att samtliga är konvergenta, dvs. låter vi antal delintervall  $n$  gå mot oändligheten så går approximationerna mot integralens värde.

Vi ser på några bilder för vänster rektangelregel där  $n$  blir allt större





Vi ser att vi allt bättre täcker upp ytan under grafen med allt fler och smalare staplar.

Nu räcker det i praktiken inte med konvergens. Vi måste få en bra approximation på en kort tid, dvs. inte behöva ta  $n$  alltför stort.

För vänster och höger rektangelregel gäller att om vi fördubblar antal delintervall så halveras felet i approximationen av integralen. För mittpunkts- och trapetsmetoderna gäller, vid samma fördubbling av antal delintervall, att felet delas med fyra, dvs. mycket bättre utdelning.

## Färdiga program i MATLAB

Det finns funktioner i MATLAB för att beräkna bestämda integraler

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

både effektivt och noggrant, t.ex. finner vi `integral` som används enligt

```
q=integral(fun,a,b)      q=integral(fun,a,b,name,value)
```

där `fun` är en funktion som beskriver integranden, `a` och `b` ger integrationsintervallet.

Om vi vill använda `integral` för att beräkna integralen  $\int_0^1 x \sin(x) dx$  så skulle det se ut så här

```
>> a=0; b=1; f=@(x)x.*sin(x);
>> q=integral(f,a,b)
```

Genom att ge `name` som `'AbsTol'` eller `'RelTol'` kan vi ange önskad absolut respektive relativ noggrannhet i beräkningen och med `value` ger vi värdet på noggrannheten.

**Uppgift 5.** Beräkna arean av det slutna området mellan graferna till funktionerna

$$g(x) = \exp\left(-\frac{x^2}{2}\right) \quad \text{och} \quad h(x) = x^2 - 3x + 2.$$

Rita en bild av området. Använd `fzero` för att beräkna skärningspunkterna mellan graferna och `integral` för att beräkna en approximation till integralen.

## 5 Differentialekvationer

Låt  $u(x)$  vara en funktion. En ekvation som beskriver ett samband mellan  $x$ ,  $u$  och derivator av  $u$  kallas, som säkert bekant, en differentialekvation. Differentialekvationens s.k. ordning är ordningen på högsta derivatan som ingår.



Den allmänna differentialekvationen av 1:a, 2:a,  $\dots$  ordningen har formerna

$$u' = f(x, u), \quad u'' = f(x, u, u'), \quad \dots$$

En differentialekvation av ordning  $n$  kan alltid skrivas om som ett system av  $n$  kopplade ekvationer av 1:a ordningen.

Vi skall se på s.k. begynnelsevärdesproblem för ekvationer av 1:a ordningen

$$\begin{cases} u' = f(t, u), & 0 \leq t \leq T \\ u(0) = u_0 \end{cases}$$

Här känner vi värdet av  $u$  vid  $t = 0$ . Vi använder  $t$  som variabel, eftersom  $t$  är ofta tiden och  $u$  tillståndet i ett mekaniskt system eller dylikt.

## Färdiga program i MATLAB

I MATLAB finns flera olika funktioner för lösning av begynnelsevärdesproblem av olika karaktär, t.ex. finner vi `ode45` som används enligt

$$[t, U] = \text{ode45}(\text{fun}, \text{tspan}, u_0) \quad [t, U] = \text{ode45}(\text{fun}, \text{tspan}, u_0, \text{opts})$$

där `fun` är en funktion som beskriver differentialekvationens högerled, `tspan` är en vektor som anger tidsintervallet och `u0` ger begynnelsevärdet för lösningen vid tiden `tspan(1)`. Funktionen `ode45` producerar som utdata dels `t`, en vektor som innehåller tidpunkter och dels `U`, en vektor eller matris som innehåller lösningen för de olika tidpunkterna.

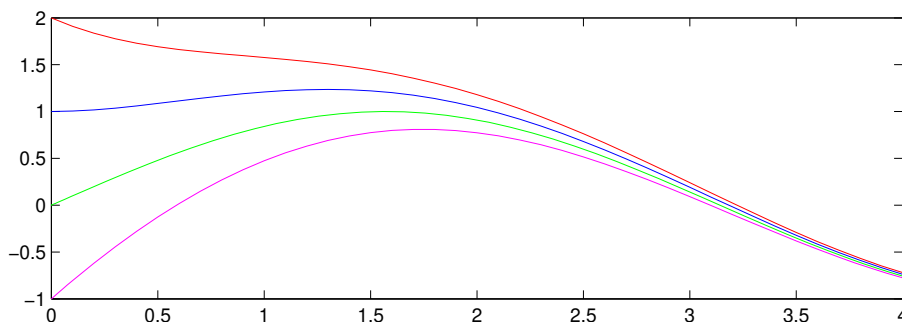
Alternativet med `opts` använder vi då vi t.ex. vill ange hur noggrant lösningen skall beräknas. Det finns också möjlighet till avbrotts hantering, mer om detta i läsperiod 4. Man skapar vektorn `opts` med funktionen `odeset`, se hjälptexten.

**Exempel 4.** Beräkna och rita lösningar till differentialekvationen

$$\begin{cases} u' = -u(t) + \sin(t) + \cos(t), & 0 \leq t \leq 4 \\ u(0) = u_0 \end{cases}$$

Vi beskriver högerledet i differentialekvationen beräknar och ritar lösningen för  $u_0 = 1$  med

```
>> f=@(t,u)-u+sin(t)+cos(t); tspan=[0 4]; u0=1;
>> [t,U]=ode45(f,tspan,u0);
>> plot(t,U)
```



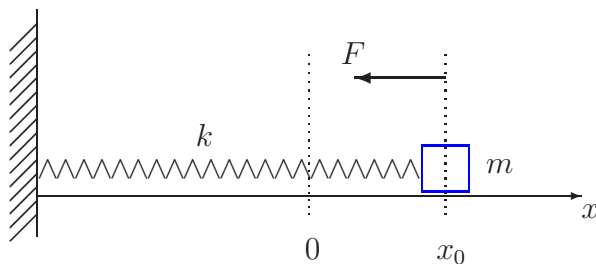
I bilden har vi även ritat ut lösningar för några andra begynnelsevärden.

**Uppgift 6.** Lös följande differentialekvation med begynnelsevillkor

$$\begin{cases} u' = \cos(3t) - \sin(5t)u, & 0 \leq t \leq 15 \\ u(0) = 2 \end{cases}$$

Rita en graf av lösningen. Använd ode45.

**Exempel 5.** Vi skall se på en harmonisk oscillator. Antag att vi har en partikel med massan  $m$  som är fastsatt i en fjäder med fjäderkonstanten  $k$ . Detta innebär att kraften då fjädern sträcks en sträcka  $x$  blir  $-kx$ . Fjäders 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  $x_0$  från jämviktspunkten, och man vill beräkna den fortsatta rörelsen.

Inför ett koordinatsystem enligt figuren ovan med origo där partikeln har sitt jämviktsläge. Då  $x$  betecknar partikeln läge får vi rörelseekvationen  $mx'' = -kx$  från Newtons andra lag ( $F = ma$ ). Detta är en differentialekvation av 2:a ordningen.

Vidare har vi begynnelsevärdena  $x(0) = x_0$ , läget vid tiden  $t = 0$ , och  $x'(0) = 0$ , partikeln i vila vid tiden  $t = 0$ .

Vi har begynnelsevärdesproblemet

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

Om vi låter  $v = x'$ , dvs. inför hastigheten, kan differentialekvationen med begynnelsevärden skrivas

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

dvs. som ett system av ekvationer av 1:a ordningen.

För att komma till standardform låter vi  $u_1 = x$  och  $u_2 = v$  och får

$$\begin{cases} u_1' = u_2, & u_1(0) = x_0 \\ u_2' = -\frac{k}{m}u_1, & u_2(0) = 0 \end{cases}$$

Med vektorbeteckningar kan detta skrivas

$$\begin{cases} \mathbf{u}' = \mathbf{f}(t, \mathbf{u}) \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{u}) = \begin{bmatrix} u_2 \\ -\frac{k}{m}u_1 \end{bmatrix}, \quad \mathbf{u}_0 = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}$$

vilket är den form som MATLAB använder.

Vi beskriver differentialekvationen med funktionen

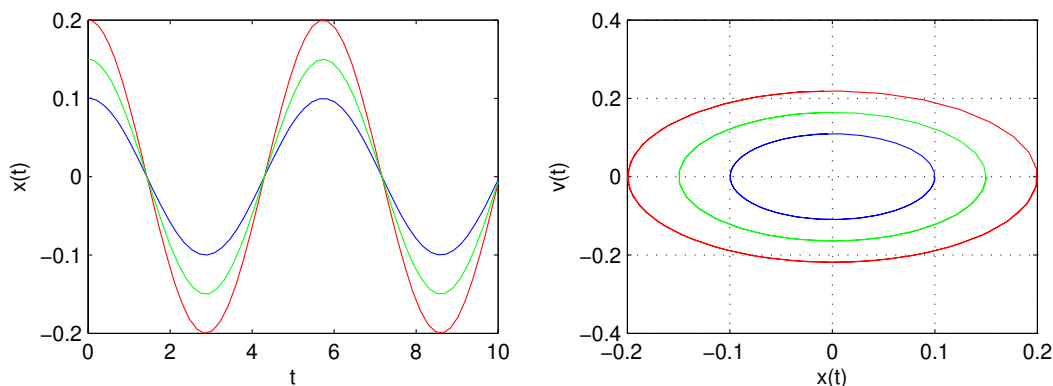
```
function f=harmonisk(t,u,k,m)
    f=[u(2)
        -k*u(1)/m];
```

Låt oss ta  $m = 0.1$  kg,  $k = 0.12$  N/m och  $x_0 = 0.1$  m. Vi beräknar nu lösning med `ode45` och ritar en bild som visar läget  $x(t)$  med

```
>> m=0.1; k=0.12; x0=0.1; v0=0; tspan=[0 10]; u0=[x0;v0];
>> [t,U]=ode45(@(t,u)harmonisk(t,u,k,m),tspan,u0);
>> plot(t,U(:,1),'b')
>> xlabel('t'), ylabel('x(t)')
```

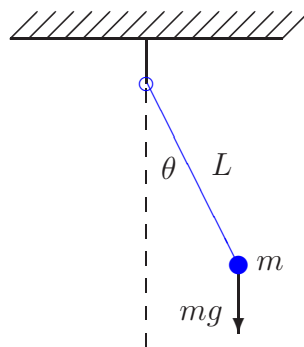
och det s.k. fasporträttet  $(x(t), v(t))$  med

```
>> plot(U(:,1),U(:,2),'b')
>> xlabel('x(t)'), ylabel('v(t)')
```



I bilden har vi även ritat ut lösningar för några andra begynnelsevärden. Ekvationen för den harmoniska oscillatorn kommer vi kunna lösa analytiskt (räkna ut en formel för lösning) i läspeiod 2. Vi avslutar med att se på en uppgift med en differentialekvation som inte har någon analytisk lösning.

**Uppgift 7.** Den matematiska pendeln. 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

$$mL\ddot{\theta}(t) = -mg \sin(\theta(t))$$

Vi vill bestämma lösningen för olika begynnelseutslag  $\theta_0$ , dvs.  $\theta(0) = \theta_0$ , då vi släpper pendeln från vila, dvs.  $\dot{\theta}(0) = 0$ . Tag  $L = 0.1$  m och begynnelseutslagen  $\theta_0 = 30^\circ, 45^\circ, 60^\circ$ . Använd `ode45`.