8 System av ickelinjära ekvationer.

Vi skall studera numerisk lösning av system av n ickelinjära ekvationer i n obekanta,

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0 \\ f_2(x_1, x_2, \cdots, x_n) = 0 \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \cdots, x_n) = 0 \end{cases}$$

Med vektorbeteckningar kan systemet skrivas

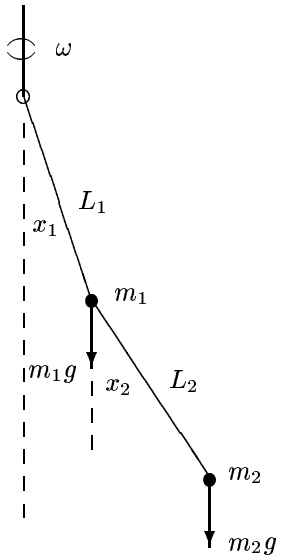
$$F(x) = 0,$$

med

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} \text{ och } x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

F är alltså en vektorvärd funktion med n komponenter, där varje komponent beror av n variabler, dvs. $F : \mathbb{R}^n \to \mathbb{R}^n$.

Exempel 8.1. En dubbel pendel roterar med vinkelhastigheten ω runt en vertikal axel.



Vid jämvikt för de två pendlarna är vinklarna mot vertikala axeln, x_1 och x_2, lösningar till följande system av ekvationer,

$$F(x) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} \tan(x_1) - \frac{\omega^2}{g}(L_1 \sin(x_1) + \frac{m_2}{m_1+m_2} L_2 \sin(x_2)) \\ \tan(x_2) - \frac{\omega^2}{g}(L_1 \sin(x_1) + L_2 \sin(x_2)) \end{bmatrix} = 0.$$

Vi skall senare i kapitlet se hur vi kan räkna fram x_1 och x_2. ■

En lösning x^* till systemet $F(x) = 0$ kallas singulär om funktionalmatrisen eller Jacobianmatrisen

$$J(x^*) = \frac{\partial F}{\partial x}(x^*) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x^*) & \cdots & \frac{\partial f_1}{\partial x_n}(x^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x^*) & \cdots & \frac{\partial f_n}{\partial x_n}(x^*) \end{bmatrix},$$

är singulär, annars kallas den reguljär. Vi nöjer oss med att bestämma reguljära lösningar i det här kompendiet.

Startapproximationer.

Att uttala sig om hur många rötter ett system av ekvationer har är betydligt svårare än för en skalär ekvation. Det gäller även startapproximationer. Vi skall se hur man kan finna en startapproximation då vi har två ekvationer i två obekanta. Senare i kapitlet skall vi återkomma till denna fråga.

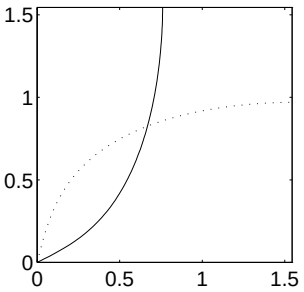
Exempel 8.2. Låt oss se på ekvationssystemet från exempel 8.1. Om vi tar $m_1 = m_2$ och $L_1 = L_2 = L$ så får vi funktionen

$$F(x) = \begin{bmatrix} \tan(x_1) - k(2\sin(x_1) + \sin(x_2)) \\ \tan(x_2) - k(2\sin(x_1) + 2\sin(x_2)) \end{bmatrix},$$

med $k = \frac{\omega^2 L}{2g}$ som parameter.

Vi tar $k = 0.4$ och använder MATLAB för att rita upp nollnivåerna för f_1 och f_2, dvs. mängderna $\{(x_1, x_2)|f_1(x_1, x_2) = 0\}$ och $\{(x_1, x_2)|f_2(x_1, x_2) = 0\}$, och se efter de punkter där både f_1 och f_2 är noll samtidigt.

```
>> x1=0:pi/120:pi/2; x2=0:pi/120:pi/2;
>> [X1,X2]=meshgrid(x1,x2);
>> k=0.4;
>> F1=tan(X1)-k*(2*sin(X1)+sin(X2));
>> F2=tan(X2)-k*(2*sin(X1)+2*sin(X2));
>> contour(x1,x2,F1,[0,0]), hold on   % Noll-nivåkurvan för f_1
>> contour(x1,x2,F2,[0,0],':')        % Noll-nivåkurvan för f_2
>> axis('square')
```



Med funktionen **ginput** kan vi läsa av en approximation av skärningspunkten.

```
>> x_start=ginput(1)
x_start =
    0.6635    0.8158
```

Vi kommer snart se att detta är en utmärkt startapproximation. ■

8.1 Newtons metod för system av ekvationer.

Antag att $x^{(0)}$ är en approximation av roten till systemet $F(x) = 0$. Taylorutvecklingen runt $x^{(0)}$ ger;

$$f_i(x) \approx f_i(x^{(0)}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^{(0)})(x_j - x_j^{(0)}), i = 1, 2, \cdots, n,$$

eller med vektor och matrisbeteckningar

$$F(x) \approx F(x^{(0)}) + J(x^{(0)})(x - x^{(0)}). \tag{8.1}$$

Vi får en lokal linjär modell av $F(x) = 0$ genom att sätta högerledet i (8.1) till 0;

$$F(x^{(0)}) + J(x^{(0)})(x - x^{(0)}) = 0.$$

och lösa detta linjära ekvationssystem med avseende på x.

Allmänt kan Newtons metod formuleras;

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

$$x^{(k+1)} = x^{(k)} + d^{(k)}, k = 0, 1, \cdots,$$

där $d^{(k)}$ är lösningen till det linjära ekvationssystemet

$$J(x^{(k)})d^{(k)} = -F(x^{(k)}).$$

Om startapproximationen ligger tillräckligt nära en rot x^* som är reguljär så konvergerar iterationerna kvadratiskt mot denna rot.

Exempel 8.3. Vi fortsätter med exempel 8.2, dvs. funktionen

$$F(x) = \begin{bmatrix} \tan(x_1) - k(2\sin(x_1) + \sin(x_2)) \\ \tan(x_2) - k(2\sin(x_1) + 2\sin(x_2)) \end{bmatrix},$$

som har derivatamatrisen

$$J(x) = \begin{bmatrix} 1 + \tan^2(x_1) - 2k\cos(x_1) & -k\cos(x_2) \\ -2k\cos(x_1) & 1 + \tan^2(x_2) - 2k\cos(x_2) \end{bmatrix}$$

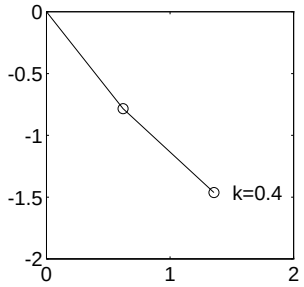
Lösningen kan vi få fram med MATLAB så här (som startapproximation tar vi $x^{(0)} = (1, 1)$)

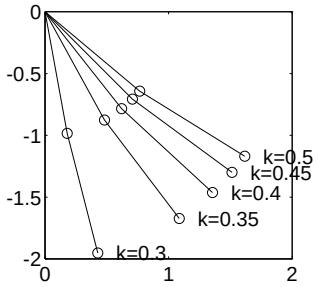
```
>> x=[1;1];
>> k=0.4;
>> for i=1:10,
      F=[tan(x(1))-k*(2*sin(x(1))+sin(x(2)))
         tan(x(2))-k*(2*sin(x(1))+2*sin(x(2)))];
      J=[1+tan(x(1))^2-2*k*cos(x(1)) -k*cos(x(2))
         -2*k*cos(x(1)) 1+tan(x(2))^2-2*k*cos(x(2))];
      x=x-J\F;
      disp([x' norm(F)])
   end
```

Vi får följande resultat

k	$x_1^{(k)}$	$x_2^{(k)}$	$\| F(x^{(k)}) \|$
0	1.00000000000000	1.00000000000000	0.58690391766643
1	0.80997024373691	0.90204950504236	0.58690391766643
2	0.69965010170664	0.84541600846910	0.16763485689433
3	0.67036024336285	0.82702065671174	0.03066795921289
4	0.66852167934804	0.82560780673114	0.00186657720807
5	0.66851437892356	0.82560123076381	0.00000824282232
6	0.66851437880113	0.82560123063839	0.00000000015517
7	0.66851437880113	0.82560123063839	0
8	0.66851437880113	0.82560123063839	0
9	0.66851437880113	0.82560123063839	0
10	0.66851437880113	0.82560123063839	0

Av tabellen ser vi att konvergensen är kvadratisk. Nu tar vi och ritar upp pendlarna, till vänster för $k = 0.4$ och till höger för några olika k-värden.





■

En dålig startapproximation kan leda till att Newtons metod divergerar, då är det lämpligt med dämpad Newton

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}, k = 0, 1, \cdots,$$
$$J(x^{(k)}) d^{(k)} = -F(x^{(k)}).$$

Dämpningsfaktorn $\alpha^{(k)}$ väljs så att $\| F(x^{(k+1)}) \| < \| F(x^{(k)}) \|$. Man kan t.ex. börja med $\alpha^{(k)} = 1$, om man inte får någon reduktion av längden av residualen kan man successivt halvera $\alpha^{(k)}$ tills man får en reduktion.

En iteration med Newtons metod kräver $\sim \frac{n^3}{3}$ operationer (Gauss-eliminationen) om Jacobianen inte har speciell struktur. Man kan modifiera Newtons metod genom att ersätta de olika matriserna $J(x^{(k)})$ med $J(x^{(0)})$. Då LU-faktoriserar man $J(x^{(0)})$ en gång för alla och använder sedan dessa faktorer för att lösa de linjära ekvationssystemen. Första iterationen kräver då $\sim \frac{n^3}{3}$ operationer, medan arbetet i de följande iterationerna bara kräver $\sim n^2$ operationer. Denna variant av Newton konvergerar dock inte kvadratiskt.

Om vi inte vill beräkna derivator så kan man approximera derivatamatrisen med differenskvoter,

$$\frac{\partial F}{\partial x_j}(x) \approx \frac{F(x + he_j) - F(x)}{h}$$

för något litet positivt h, där e_j är j:te enhetsvektorn. I MATLAB OPTIMIZATION TOOLBOX finner vi **fsolve** som använder denna tekniken.

Exempel 8.4. Vill vi använda **fsolve** måste vi skapa en fil med en beskrivning av funktionen F.

```
function F=dubbel(x,k)
F=[tan(x(1))-k*(2*sin(x(1))+sin(x(2)))
   tan(x(2))-k*(2*sin(x(1))+2*sin(x(2)))];
```

och i MATLAB löser vi problemet med

```
>> k=0.4;
>> x0=[1;1];
>> x=fsolve(@dubbel,x0,[],k)
x =
   0.66851508712394
   0.82560158848836
```

■

8.2 Inbäddning.

Ibland kan det vara svårt att hitta en tillräckligt bra startapproximation till ett ickelinjärt ekvationssystem $F(x) = 0$, då kan man definiera en familj av ekvationer (inbäddning)

$$H(x, \lambda) = 0, 0 \leq \lambda \leq 1,$$

så att

$$H(x, 0) = E(x)$$

är enkel att lösa eller har känd lösning och

$$H(x, 1) = F(x)$$

är vår ursprungliga ekvation.

Inom t.ex. strukturmekanik kan $\lambda = 0$ motsvara en obelastad konstruktion för vilken man inte har någon förskjutning medan $\lambda = 1$ motsvarar den aktuella lasten som man vill beräkna förskjutningen för.

Antag att det från en lösning $x(0)$ av $E(x) = 0$ utgår en kurva $x(\lambda)$ av lösningar till $H(x(\lambda), \lambda) = 0$ för $0 \leq \lambda \leq 1$, så kan vi följa denna kurva fram till $x(1)$, som ju är en lösning till $F(x) = 0$, genom att lösa en följd av ekvationer

$$H(x(\lambda_i), \lambda_i) = 0, 0 < \lambda_1 < \lambda_2 \cdots < \lambda_m = 1,$$

där $\lambda_1, \lambda_2, \cdots$ väljs så att $x(\lambda_i)$, lösningen till $H(x, \lambda_i) = 0$, duger som startapproximation för beräkning av lösningen $x(\lambda_{i+1})$ till ekvationen $H(x, \lambda_{i+1}) = 0$ med någon lämplig metod, t.ex. Newtons metod.

Man kan konstruera olika inbäddningar, ett exempel är;

$$H(x, \lambda) = \lambda F(x) + (1 - \lambda)E(x).$$

Tyvärr är det inte alltid lätt att hitta en inbäddning med kurvor som förbinder lösningar till $E(x) = 0$ med lösningar till $F(x) = 0$. Kurvorna kan ha singulära punkter där de kan förgrena sig eller vika sig över sig själva.

Ibland förekommer en parameter naturligt i ekvationen. T.ex. för vår dubbelpendel skulle vi kunna ta $\lambda = \frac{0.4}{k}$. Då $\lambda = 0$, dvs. då $k = \infty$, är lösningen $x_1 = x_2 = \frac{\pi}{2}$ känd. Vi kan lösa ekvationen för en följd av λ-värden och då $\lambda = 1$ är $k = 0.4$, dvs. vi har vår ursprungliga ekvation.

8.3 Övningar.

1. Bestäm lösningarna till följande ickelinjära ekvationssystem

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

   dvs. hitta alla punkter (x_1, x_2) som uppfyller bägge ekvationerna samtidigt. Ledning: Gör en variabelreduktion, dvs. inför en parametrisering av lösningarna till den ena ekvationen.

2. En 20 m lång skena är fast inspänd. På grund av värme utvidgas skenan och blir 2 mm längre, och kommer då att böjas ut i, som vi antar, cirkelform (solkurva på järnväg). Beräkna den maximala utböjningen med tre signifikanta siffror.

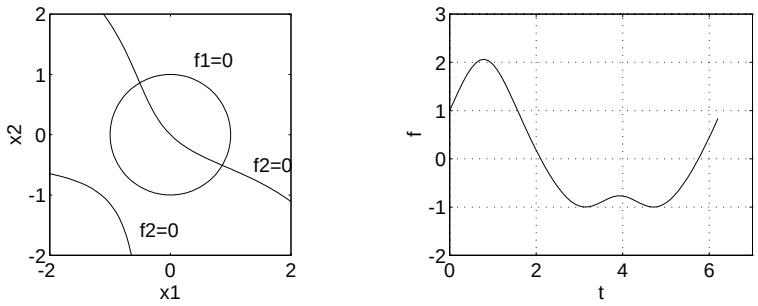
3. Bestäm med fem korrekta decimaler den lösning till systemet

$$\begin{cases} \sin(x_1 + x_2) = x_1 \\ \cos(x_1 - x_2) = x_2 \end{cases}$$

   som ligger i första kvadranten.

8.4 Lösningar.

1. Vi ritar noll-nivåkurvorna till $f_1 = x_1^2 + x_2^2 - 1$ och $f_2 = \exp(x_1 x_2) + x_1 + x_2 - 1$ och får figuren nedan till vänster.

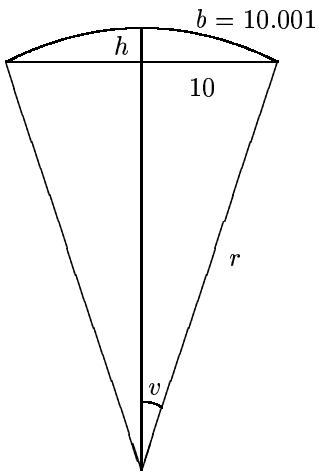


   Låt $x_1 = \cos(t)$ och $x_2 = \sin(t)$. Vi får då ekvationen

$$f(t) = \mathrm{e}^{\cos(t)\sin(t)} + \cos(t) + \sin(t) - 1 = 0$$

   som vi löser med avseende på t. Grafen till f ser vi ovan till höger, där vi lätt läser av startapproximationer. Med t.ex. Newtons metod får vi $t^* = 5.7509$ och därmed $x_1^* = 0.8617, x_2^* = -0.5075$ samt $t^* = 2.1030$ och därmed $x_1^* = -0.5075, x_2^* = 0.8617$.

2. Vi inför beteckningar enligt figuren så gäller att $h = (1 - \cos(v))r$.



Lite geometri ger ett ickelinjärt ekvationssystem i r och v

$$\begin{cases} rv = 10.001 \\ r\sin(v) = 10 \end{cases}$$

När vi väl löst detta system kan vi beräkna h. Låter vi $x_1 = r$ och $x_2 = v$ kan systemet skrivas $F(x) = 0$, där

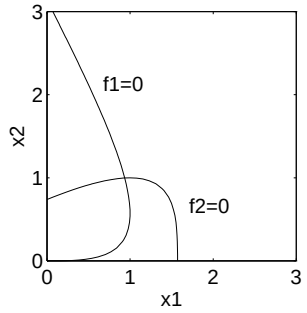
$$F(x) = \begin{bmatrix} x_1 x_2 - 10.001 \\ x_1 \sin(x_2) - 10 \end{bmatrix}$$

med Jacobianmatrisen

$$J(x) = \begin{bmatrix} x_2 & x_1 \\ \sin(x_2) & x_1\cos(x_2) \end{bmatrix}$$

Med t.ex. Newtons metod får vi $r = 408.30$ och $v = 0.12238$. Utböjningen blir $h = 12.25$ cm.

3. Vi ritar upp noll-nivåkurvorna till $f_1 = \sin(x_1 + x_2) - x_1$ och $f_2 = \cos(x_1 - x_2) - x_2$, för att kunna läsa av en startapproximation.



Vi ser att $(x_1^*, x_2^*) \approx (1, 1)$ och får en noggrann bestämning med **fsolve**

```
>> F=inline('[sin(x(1)+x(2))-x(1); cos(x(1)-x(2))-x(2)]','x')
F =
     Inline function:
     F(x) = [sin(x(1)+x(2))-x(1); cos(x(1)-x(2))-x(2)]
>> x0=[1; 1];
>> x=fsolve(F,x0)
x =
    0.9351
    0.9980
```

9 Optimering.

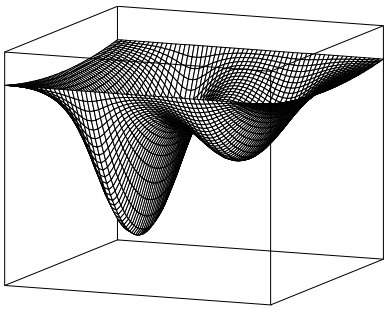
Vi skall se på numerisk lösning av minimeringsproblem skrivna på formen

$$\min_{x\in\Omega} F(x)$$

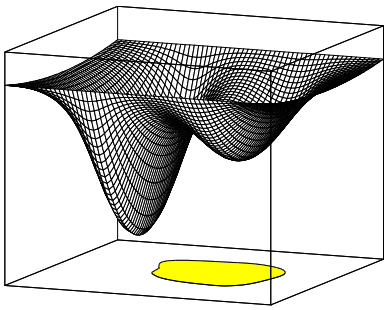
där Ω kallas *tillåten mängd* och $F : \mathbb{R}^n \mapsto \mathbb{R}$ kallas *målfunktion* eller *objektfunktion.*

Ur programvarusynpunkt delar man upp problemen i två huvudklasser

- Minimering *utan* bivillkor: $\Omega = \mathbb{R}^n$
- Minimering *med* bivillkor: $\Omega = \{x \in \mathbb{R}^n \mid c(x) = 0,\ d(x) \leq 0\}$, där $c : \mathbb{R}^n \mapsto \mathbb{R}^m$ och $d : \mathbb{R}^n \mapsto \mathbb{R}^q$.

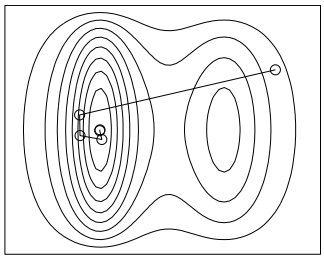


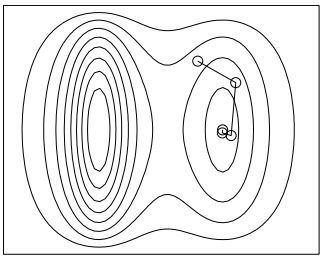
$\Omega = \mathbb{R}^2$



Ω begränsad mängd

Vi skall se på metoder som bestämmer ett lokalt minimum utgående ifrån en startapproximation. Först utan bivillkor, sedan med.





Om detta lokala min också är globalt får vi själva hålla reda på. Bilden visar (i fallet $\Omega = \mathbb{R}^2$) hur olika startapproximationer kan leda oss till olika lokala minpunkter.

9.1 Minimering utan bivillkor.

Nödvändiga villkor för att x^* skall vara ett lokalt minimum till F är

(1) $g(x^*) = 0$, dvs. x^* en stationär punkt.
(2_N) $G(x^*)$ är positivt semi-definit, dvs. inga negativa egenvärden.

Här är $g(x) = F'(x)$ gradienten (en kolonnvektor) och $G(x) = F''(x)$ är Hessianen (en matris).

$$g(x) = \left[\frac{\partial F}{\partial x_i}\right]_i \qquad G(x) = \left[\frac{\partial^2 F}{\partial x_i \partial x_j}\right]_{i,j}$$

Tillräckliga villkor för att x^* skall vara ett *starkt* lokalt minimum till F är förutom (1)

(2_T) $G(x^*)$ är positivt definit, dvs. alla egenvärden positiva.

Exempel 9.1. Ser på $F(x) = \frac{1}{2}x^T Ax + x^T b$ där A symmetrisk matris.

Om $\{u_i\}$ är en ON-bas av egenvektorer till A med motsvarande egenvärden λ_i så kan en godtycklig vektor δx skrivas $\delta x = \sum_i \alpha_i u_i$.

Antag att x^* stationär punkt. Vi har då att $g(x^*) = Ax^* + b = 0$ och $G(x^*) = A$.

Taylorutveckling ger

$$F(x^* + \delta x) = F(x^*) + \delta x^T g(x^*) + \tfrac{1}{2}\delta x^T G(x^*)\delta x = \\ = F(x^*) + \tfrac{1}{2}\delta x^T A \delta x = \\ = F(x^*) + \tfrac{1}{2}(\textstyle\sum_i \alpha_i u_i)^T A(\sum_i \alpha_i u_i) = \\ = F(x^*) + \tfrac{1}{2}\textstyle\sum_i \alpha_i^2 \lambda_i$$

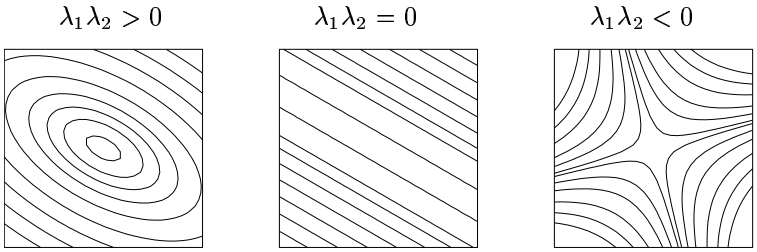
Vi har alltså $F(x^* + \delta x) = F(x^*) + \frac{1}{2}\sum_i \alpha_i^2 \lambda_i$ och får följande fall

(1) $\lambda_i > 0 \ \ \forall i \ \ \Rightarrow x^*$ starkt minimum

(2) $\lambda_i \geq 0,\ \lambda_i = 0$ för något $i \Rightarrow x^*$ svagt minimum

(3) λ_i olika tecken $\Rightarrow x^*$ sadelpunkt

I två dimensioner ser det ut så här



■

För att finna ett minimum kan man börja med att bestämma en stationär punkt och sedan avgöra om det är en minpunkt.

Först ser vi på envariabel fallet. Vi kan lösa $g(x) = F'(x) = 0$ med Newtons metod (använder 1:a och 2:a derivata)

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} \ \text{ eller } \ x_{k+1} = x_k - \frac{F'(x_k)}{F''(x_k)}$$

eller sekantmetoden (använder bara 1:a derivata)

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

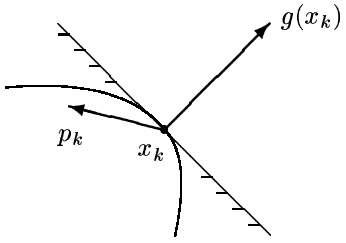
Nu ser vi på flervariabel fallet. Vi kan lösa ickelinjära systemet $g(x) = 0$ med en iterationsmetod av typen

$$x_{k+1} = x_k + \alpha_k p_k$$

där p_k är en sökriktning och α_k en steglängd.

Ett specialfall är Newtons metod $\alpha_k = 1, \ \ p_k = -G(x_k)^{-1} g(x_k)$

En sökriktning p_k kallas för en *descentriktning* om $p_k^T g(x_k) < 0$.



$F(x) = konst.$

Då finns en steglängd α_k så att $F(x_k + \alpha_k p_k) < F(x_k)$. Vi har $\frac{\partial F}{\partial p_k}(x_k) = p_k^T g(x_k) < 0 \Rightarrow F$ avtar i riktningen p_k (åtminstone lokalt).

Speciellt gäller $p_k^T g(x_k) < 0$ om vi tar

$$p_k = -g(x_k)$$

Med detta val av p_k får vi gradientmetoden eller *Steepest Descent*-metoden.

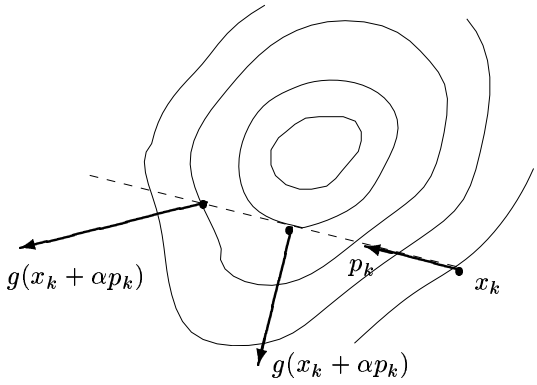
9.1.1 Linjesökning.

Antag att vi har en sökriktning p_k. Hur långt skall vi gå längs denna riktning, dvs. hur stort skall vi välja α_k ?

Helst vill vi ta α_k som lösning till

$$\min_{\alpha} s(\alpha) \equiv F(x_k + \alpha p_k)$$

Lösningen α^* uppfyller $s'(\alpha^*) = 0$. Nu är $s'(\alpha) = p_k^T g(x_k + \alpha p_k)$ så vi skall helst gå till den punkt där gradienten är vinkelrät mot p_k.



Exempel 9.2. För det kvadratisk fallet $F(x) = \frac{1}{2}x^T Ax + x^T b$ har vi $g(x) = Ax + b$ och får

$$s'(\alpha) = p_k^T g(x_k + \alpha p_k) = p_k^T A(x_k + \alpha p_k) + p_k^T b = p_k^T g(x_k) + \alpha p_k^T A p_k$$

Följer att $\quad s'(\alpha^*) = 0 \Leftrightarrow \alpha^* = -\frac{p_k^T g(x_k)}{p_k^T A p_k}$. ■

Normalt skulle det kosta för mycket att bestämma α^* exakt, så man nöjer sig med en (ibland rätt grov) approximativ lösning.

9.1.2 Parameteranpassning.

Ett speciellt fall av minimering utan bivillkor är parameteranpassning. Antag vi vill anpassa en modell $\psi(x,t)$, som är ickelinjär i x, till en serie mätdata $(t_i, y_i), i = 1, \cdots, m$. Vårt problem blir att minimera

$$F(x) = \frac{1}{2} \parallel r(x) \parallel_2^2 = \frac{1}{2} \sum_{i=1}^{m} \rho_i(x)^2$$

där $\rho_i(x) = \psi(x, t_i) - y_i$. Nu har funktionen F en speciell struktur och vi får derivatorna

$$g(x) = J(x)^T r(x), \quad G(x) = J(x)^T J(x) + \sum_{i=1}^{m} \rho_i(x) H_i(x)$$

där

$$J(x) = \left[\frac{\partial \rho_i}{\partial x_j}\right]_{i,j}, \quad H_i(x) = \left[\frac{\partial^2 \rho_i}{\partial x_k \partial x_j}\right]_{k,j}$$

Newton riktningen blir

$$p_k = -(J_k^T J_k + \sum_{i=1}^{m} \rho_i(x_k) H_i(x_k))^{-1} J_k^T r_k$$

Om ρ_i små (bra anslutning i modellen) eller H_i små (nästan linjär modell) så kan Newton riktningen ersättas med den s.k. Gauss-Newton riktningen (som bara använder 1:a derivator)

$$p_k = -(J_k^T J_k)^{-1} J_k^T r_k$$

Nu är p_k lösningen till det linjära minsta-kvadrat problemet

$$\min_{p} \parallel J_k p + r_k \parallel_2^2$$

som är en linjärisering av r, eftersom $r(x_k + p) \approx r(x_k) + J(x_k)p$.

9.2 Minimering med bivillkor. (Överkurs)

Vi nöjer oss med att se på minimeringsproblem med likhetsbivillkor

$$\min_{x\in\Omega} F(x)$$

där $\Omega = \{x \in \mathbb{R}^n \,|\, c(x) = 0\}$ med $c : \mathbb{R}^n \mapsto \mathbb{R}^m$.

Bilda *Lagrangefunktionen*

$$L(x,\lambda) = F(x) + \lambda^T c(x) = F(x) + \sum_{i=1}^{m} \lambda_i c_i(x)$$

där $\lambda \in \mathbb{R}^m$ kallas *Lagrange multiplikatorer.*

Vi deriverar

$$L'_x(x,\lambda) = g(x) + c'(x)^T\lambda = g(x) + \sum_{i=1}^{m} \lambda_i c'_i(x)$$

$$L'_\lambda(x,\lambda) = c(x)$$

$$L''_{xx}(x,\lambda) = G(x) + \sum_{i=1}^{m} \lambda_i c''_i(x)$$

$$L''_{x\lambda}(x,\lambda) = c'(x)^T, \qquad L''_{\lambda x}(x,\lambda) = c'(x)$$

där

$c'(x) = [\frac{\partial c_k}{\partial x_j}]_{k,j}$	funktionalmatrisen till $c(x)$
$c'_i(x) = [\frac{\partial c_i}{\partial x_k}]_k$	gradienten till $c_i(x)$
$c''_i(x) = [\frac{\partial^2 c_i}{\partial x_k \partial x_j}]_{k,j}$	Hessianen till $c_i(x)$

x^* kallas en *reguljär punkt* på $c(x) = 0$ om $c'(x^*)$ har full radrang, dvs. om c'_i, gradienterna av c_i, är linjärt oberoende.

Nödvändiga villkor för att en reguljär punkt x^* skall vara ett lokalt minimum

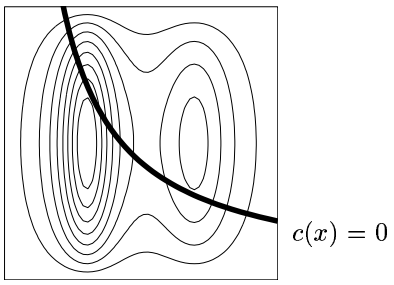
(1) $L'_x(x^*,\lambda^*) = 0$, dvs. $g(x^*) + c'(x^*)^T\lambda^* = 0$
(2) $L'_\lambda(x^*,\lambda^*) = 0$, dvs. $c(x^*) = 0$
(3_N) $L''_{xx}(x^*,\lambda^*)$ positivt semi-definit på

$$M = \mathcal{N}(c'(x^*)) = \{y \in \mathbb{R}^n \,|\, c'(x^*)y = 0\}$$

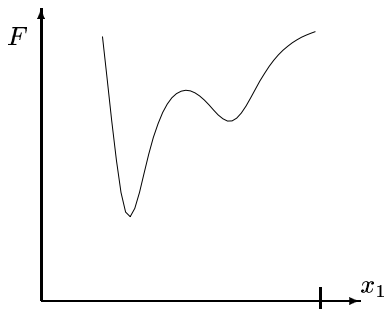
M är tangentplanet till bivillkoret $c(x) = 0$ i $x = x^*$.

Tillräckliga villkor för att x^* skall vara ett *starkt* lokalt minimum är (1), (2) och

(3_T) $L''_{xx}(x^*,\lambda^*)$ positivt definit på M



$g(x^*) + c'(x^*)^T\lambda^* = 0$



$F(x)$ över $c(x) = 0$ norrut

9.2.1 Lagrange system.

Vi kan lösa Lagrange systemet

$$H(y) = H(x,\lambda) = \begin{bmatrix} L'_x(x,\lambda) \\ L'_\lambda(x,\lambda) \end{bmatrix} = \begin{bmatrix} g(x) + c'(x)^T\lambda \\ c(x) \end{bmatrix} = 0$$

Givet en approximation $y_k = (x_k, \lambda_k)$ så kan vi använda Newtons metod

$$y_{k+1} = y_k - H'(y_k)^{-1}H(y_k)$$

där

$$H'(y) = \begin{bmatrix} L''_{xx} & L''_{x\lambda} \\ L''_{\lambda x} & 0 \end{bmatrix} = \begin{bmatrix} G(x) + \sum_{i=1}^m \lambda_i c''_i(x) & c'(x)^T \\ c'(x) & 0 \end{bmatrix}$$

9.2.2 Straff-funktion.

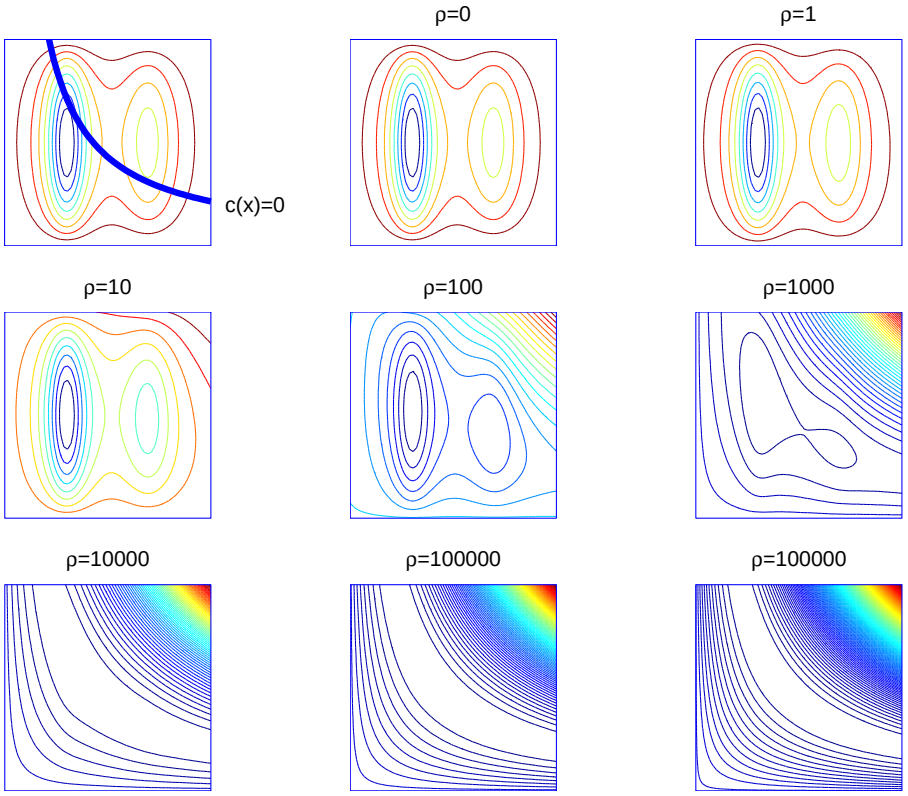
Vi kan också återföra problemet till en följd av problem utan bivillkor genom att bilda en s.k. *straff-funktion* (man inför ett straff om $c(x) \neq 0$)

$$P(x,\rho) = F(x) + \frac{\rho}{2}\|c(x)\|^2$$

som man minimerar för en växande följd av positiva ρ-värden.

Om x_k^* lösning till $\min_x P(x,\rho_k)$ så gäller oftast att $x_k^* \to x^*$ då $\rho_k \to +\infty$ där x^* lösningen till $\min_{c(x)=0} F(x)$. Vidare har vi då att $\rho_k c(x_k^*) \to \lambda^*$ då $\rho_k \to +\infty$.

Exempel 9.4. Här ser vi nivåkurvorna till vårt favoritexempel för olika värden på ρ. Vi ser att dalgångens väggar blir allt brantare.



■

Minimeringsproblemet kan bli illakonditionerat då ρ_k stort. Man kan regularisera straff-funktionsmetoden genom att införa en utökad Lagrangefunktion, men vi går inte in på det.

9.3 MATLAB OPTIMIZATION TOOLBOX.

För minimering utan bivillkor finns **fminbnd** för envariabel fallet och **fminunc** och **fminsearch** för flervariabel fallet. Den senare funktionen används i de fall då objektfunktionen inte är deriverbar.

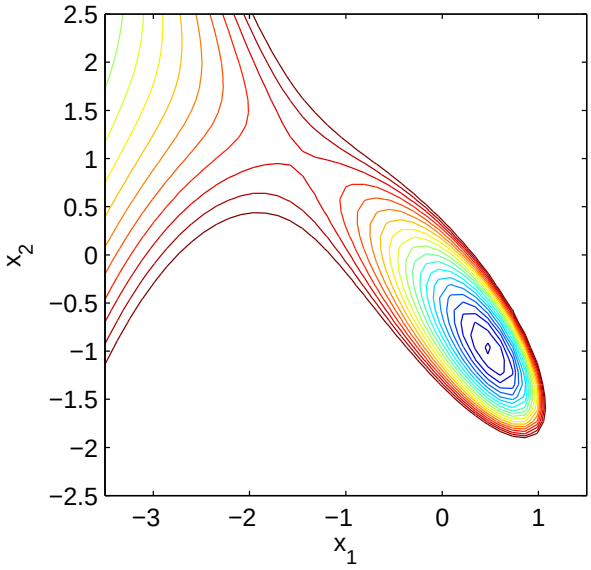
Överbestämda ickelinjära ekvationer löses med **lsqnonlin** och med **lsqcurvefit** görs parameteranpassning.

Funktionen **fmincon** är avsedd för optimeringsproblem med bivillkor.

Exempel 9.5. Antag vi vill finna ett lokalt minimum till

$$F(x) = \mathrm{e}^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

```
>> x1=linspace(-3.5,1.5,40); x2=linspace(-2.5,2.5,40); lv=linspace(0.025,2,20);
>> [X1,X2]=meshgrid(x1,x2);
>> F=exp(X1).*(4*X1.^2+2*X2.^2+4*X1.*X2+2*X2+1);
>> contour(X1,X2,F,lv)
>> xlabel('x_1'), ylabel('x_2'), axis('square')
```



Vi skapar en fil med objektfunktionen

```
function f=fun_ex9_5(x)
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

och läser in startapproximation och bestämmer noggrann lösning med

```
>> x0=ginput(1); x0=x0';
>> x=fminunc(@fun_ex9_5,x0)
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.
x =
0.5000
-1.0000
```

■

Exempel 9.6. Vi har följande modell av ett insvägningsförlopp

$$\psi(x,t) = x_1 + \sin(x_2 t)\exp(-x_3 t)$$

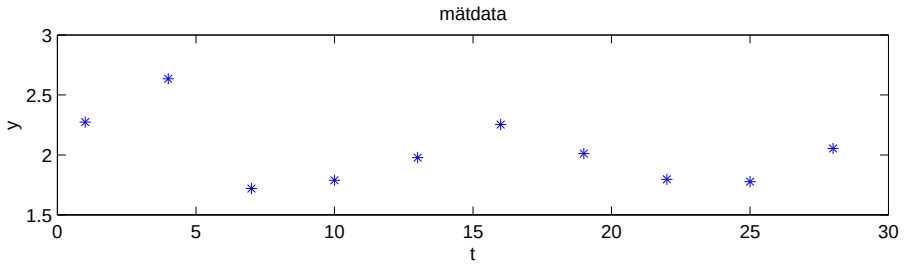
som vi vill anpassa till en mätserie $(t_1, y_1), (t_2, y_2), \cdots, (t_m, y_m)$.

Vi skall därför lösa

$$\min_{x \in \mathbb{R}^3} f(x) = \sum_{i=1}^{m} (\psi(x, t_i) - y_i)^2$$

Vi ritar upp mätserien

```
>> t=[1 4 7 10 13 16 19 22 25 28]';
>> y=[2.2734 2.6353 1.7202 1.7887 1.9781 2.2526 2.0107 1.7970 1.7775 2.0543]';
>> plot(t,y,'*')
>> xlabel('t'), ylabel('y'), title('mätdata')
```



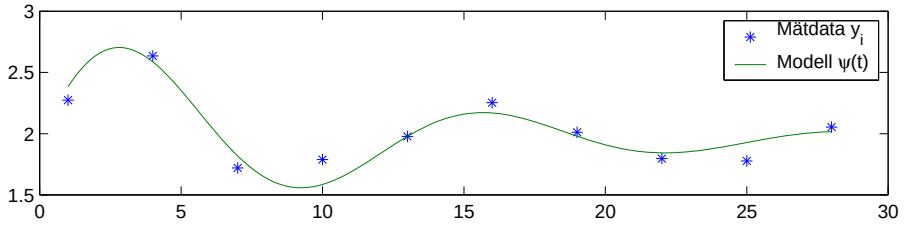
Med tanke på modellen borde x_1 vara nära medelvärdet av y-värdena. Av grafen ser vi att x_2 borde vara nära 0.4 (svängningen hinner med ungefär två perioder på tidsintervallet) och x_3 skattar vi till 0.1.

Nu beskriver vi modellens värde i de olika t-punkterna och bestämmer parametrarna $x = (x_1, x_2, x_3)$ med **lsqcurvefit**

```
>> x0=[mean(y);0.4;0.1];
>> disp(x0')
2.02878000000000 0.40000000000000 0.10000000000000

>> psi=inline('x(1)+sin(x(2)*t).*exp(-x(3)*t)','x','t');
>> x=lsqcurvefit(psi,x0,t,y);
>> disp(x')
1.95781412843194 0.48866258469118 0.09716146285612

>> te=linspace(t(1),t(end),200); % täta t-värden ger snygg graf
>> plot(t,y,'*',te,psi(x,te)) % rita data och modell
>> legend('Mätdata y_i','Modell \psi(t)')
```



■

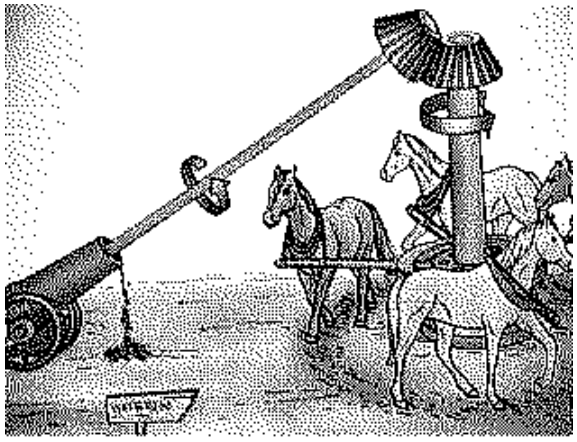
9.4 Övningar.

1. I samband med studiet av diffraktion vill man söka lokala maximum till funktionen

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

   Hur många lokala maximum finns det. Bestäm några av de minsta (positiva).

2. Greve Rumford utförde 1798 följande experiment. Ett kanonrör uppvärmdes till temperaturen u_0 genom att ett trubbigt borr vreds runt av hästar i 30 minuter.

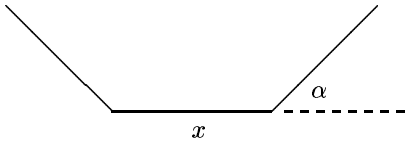


Sedan fick röret svalna medan man då och då mätte temperaturen u i röret. Man fick följande mätserie

t (min)	4	5	7	12	14	16	20	24	28	31	34	37.5	41
u (°F)	126	125	123	120	119	118	116	115	114	113	112	111	110

Ta fram en modell som bygger på Newtons avsvalningslag och anpassa denna till mätdata i minsta-kvadrat mening.

3. En timmerränna har i genomskärning formen av en likbent paralleltrapets. Vid tillverkningen används 6 meter breda metallplåtar. Hur ska dessa bockas för att rännan ska rymma så mycket som möjligt?



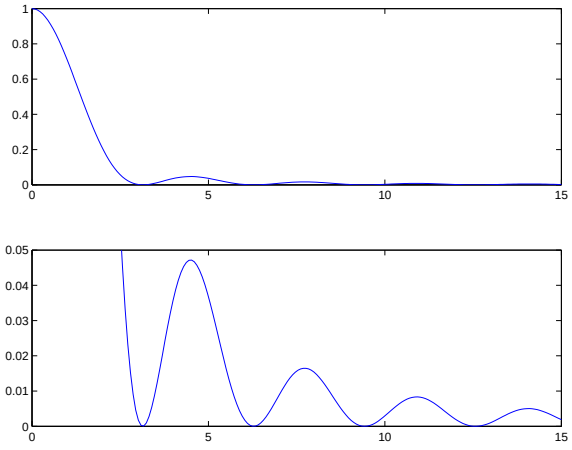
Beskriv tvärsnittsarean A som funktion av x och α.

Rita en bild av funktionsytan till $A(x, \alpha)$, rita även nivåkurvor till funktionen. Försök läsa av en startapproximation, av maxpunkten, i figuren. Använd sedan t.ex. **fminsearch** i MATLAB för att beräkna en noggrann approximation av lösningen.

9.5 Lösningar.

1. Vi minimerar $f(u) = -y$ med **fminbnd**

```
>> u=linspace(1e-3,15,200);
>> y=inline('sin(u).^2./u.^2');
>> subplot(2,1,1),  plot(u,y(u))
>> subplot(2,1,2),  plot(u,y(u))
>> axis([0 15 0 0.05])
>> [umax_l,y_0]=ginput(1);
>> [umax_r,y_0]=ginput(1);
>> f=inline('-sin(u).^2./u.^2');
>> umax=fminbnd(f,umax_l,umax_r)
umax =
    4.4934
>> y(umax)
ans =
    0.0472
```



2. Om begynnelsetemperaturen i kanonröret (efter uppvärmningen) är u_0 och omgivande temperaturen är u_{omg} så ger Newtons avsvalningslag:

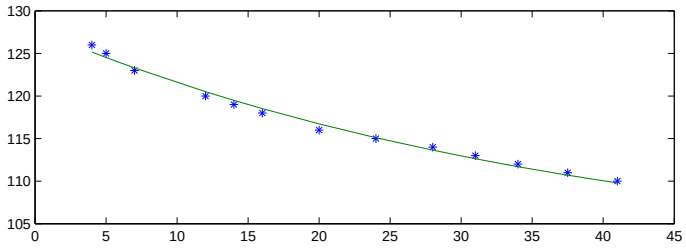
$$\begin{cases} u' = -\theta(u - u_{omg}), t > 0 \\ u(0) = u_0 \end{cases}$$

Lösningen till detta begynnelsevärdesproblem är $u(t,\theta) = u_{omg} + (u_0 - u_{omg})\exp(-\theta t)$ och vårt överbestämda systemet blir

$$f_i(\theta) = u(t_i,\theta) - u_i = 0, i = 1,\cdots,m$$

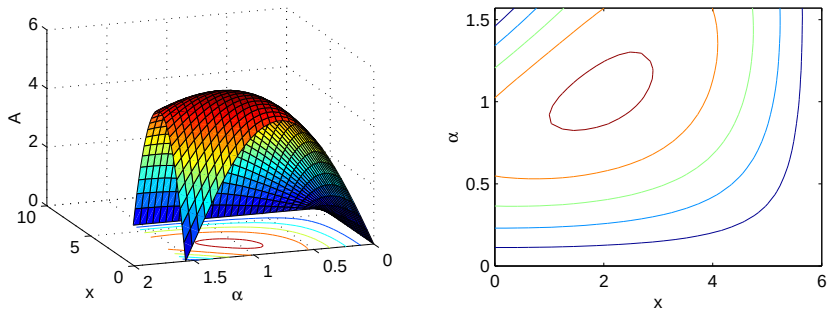
Vi antar att $u_{omg} = 100$ och bestämmer θ och u_0 med **lsqcurvefit** i MATLAB

```
>> t=[4 5 7 12 14 16 20 24 28 31 34 37.5 41]';
>> u=[126 125 123 120 119 118 116 115 114 113 112 111 110]';
>> u_omg=100;
>> plot(t,u,'*')
>> theta=0.1; u_0=130;   % Startgissning från graf
>> f=inline('u_omg+(x(2)-u_omg)*exp(-x(1)*t)','x','t','u_omg')
>> x=[theta u_0]';
>> x=lsqcurvefit(f,x,t,u,[],[],[],u_omg)
x =
   2.5527e-02
   1.2787e+02
>> plot(t,u,'*',t,f(x,t,u_omg))
>> res=norm(f(x,t,u_omg)-u)
res =
   1.6831e+00
```



3. Arean ges av $A(x,\alpha) = x(3-\frac{x}{2})\sin(\alpha) + \frac{1}{2}(3-\frac{x}{2})^2\sin(2\alpha)$ och vi ritar funktionsyta och nivåkurvor, läser av startapproximation och bestämmer noggrann lösning med:

```
>> N=30; [X,Alpha]=meshgrid(linspace(0,6,N),linspace(0,pi/2,N));
>> A=X.*(3-X/2).*sin(Alpha)+1/2*(3-X/2).^2.*sin(2*Alpha);
>> subplot(2,2,1), surfc(X,Alpha,A), grid on, view(-110,20)
>> xlabel('x'), ylabel('\alpha'), zlabel('A')
>> subplot(2,2,2), contour(X,Alpha,A), xlabel('x'), ylabel('\alpha')
```




```
>> x0=ginput(1); x0=x0';
>> area=inline('x(1)*(3-x(1)/2)*sin(x(2))+1/2*(3-x(1)/2)^2*sin(2*x(2))','x');
>> x=fminsearch(area,x0)
x =
   2.0000
   1.0472
```