Optimeringsproblem

1 Inledning

Vi skall söka minsta eller största värdet hos en funktion på en mängd, dvs. vi skall lösa s.k. optimeringsproblem

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}) \quad \text{eller} \quad \max_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$$

där $f : \mathbb{R}^n \to \mathbb{R}$ är en funktion och Ω är en mängd som utgörs av hela eller en del av definitionsmängden D_f.

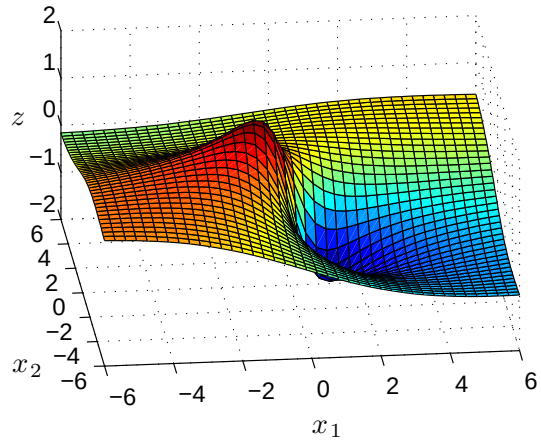
Om Ω är hela definitionsmängden säger vi att vi har ett problem *utan bivillkor* och om Ω är är en del av definitionsmängden har vi ett problem *med bivillkor*.

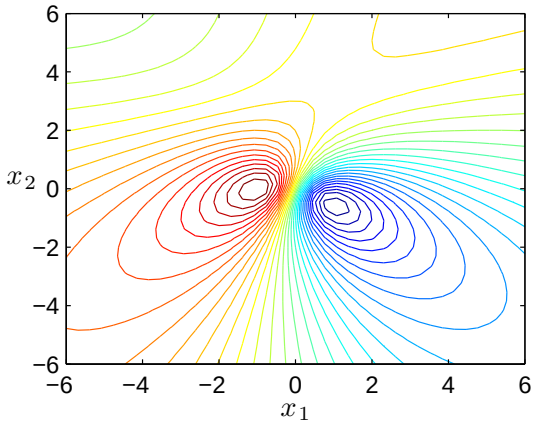
2 Optimering utan bivillkor

Vi skall bestämma maximi- och minimipunkter samt sadelpunkter till en funktion $f : \mathbb{R}^2 \to \mathbb{R}$. Som exempel tar vi funktionen

$$f(\boldsymbol{x}) = f(x_1, x_2) = \frac{x_2 - 3\,x_1 + x_1\,x_2}{1 + x_1^2 + x_2^2}$$

Vi ritar upp funktionsytan samt nivåkurvor för att få en känsla för var de stationära punkterna finns och av vilken typ de är.





Det ser ut som vi har tre stationära punkter, en lokal minimipunkt, en lokal maximipunkt och en sadelpunkt.

En stationär punkt till $f(\boldsymbol{x})$ är en punkt $\boldsymbol{a}$ för vilken gradientvektorn $\nabla f(\boldsymbol{a}) = \boldsymbol{0}$. Bestämningen av vilken typ av stationär punkt det är kan vi göra med hjälp av egenvärdena till Hessianmatrisen.

I en tidigare laboration linjäriserade vi en funktion $f(\boldsymbol{x})$ runt $\boldsymbol{a}$ för att beskriva funktionsytans lutning, dvs. vi gjorde en linjär modell av funktionen runt $\boldsymbol{a}$ med

$$f(\boldsymbol{x}) \approx L(\boldsymbol{x}) = f(\boldsymbol{a}) + \boldsymbol{\nabla} f(\boldsymbol{a})^T(\boldsymbol{x} - \boldsymbol{a})$$

Nu vill vi ha en modell av $f(\boldsymbol{x})$ runt den stationära punkten $\boldsymbol{a}$ som beskriver hur funktionsytan buktar (har vi en kulle, en svacka eller ett pass på ytan), och då ger den linjära modellen inte tillräckligt med information.

En kvadratisk modell av funktionen $f(\boldsymbol{x})$ runt $\boldsymbol{a}$ kommer däremot att beskriva hur funktionsytan buktar och den modellen ges av (Taylorutveckling runt $\boldsymbol{a}$)

$$f(\boldsymbol{x}) \approx f(\boldsymbol{a}) + \boldsymbol{\nabla} f(\boldsymbol{a})^T(\boldsymbol{x} - \boldsymbol{a}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{a})^T \boldsymbol{H}(\boldsymbol{a})(\boldsymbol{x} - \boldsymbol{a})$$

där $\boldsymbol{H}(\boldsymbol{x})$ är Hessianmatrisen av andra derivator

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} f''_{x_1x_1}(\boldsymbol{x}) & f''_{x_1x_2}(\boldsymbol{x}) \\ \\ f''_{x_2x_1}(\boldsymbol{x}) & f''_{x_2x_2}(\boldsymbol{x}) \end{bmatrix}$$

Eftersom $\boldsymbol{a}$ en stationär punkt så är $\boldsymbol{\nabla} f(\boldsymbol{a}) = \boldsymbol{0}$ och vi får att

$$f(\boldsymbol{x}) \approx f(\boldsymbol{a}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{a})^T \boldsymbol{H}(\boldsymbol{a})(\boldsymbol{x} - \boldsymbol{a})$$

Genom att beräkna egenvärdena till $\boldsymbol{H}(\boldsymbol{a})$ kan vi avgöra vilken typ av stationär punkt vi har (se Adams s. 746). Är egenvärdena positiva har vi en minimipunkt, är de negativa har vi en maximipunkt och har de olika tecken har vi en sadelpunkt.

Närmast skall vi beskriva hur vi kan använda Newtons metod för att beräkna stationära punkter genom att lösa ekvationen $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$ och sedan skall vi beskriva Steepest descentmetoden för att beräkna lokala minimipunkter. Men först en liten uppgift.

Uppgift 1. Betrakta funktionen $f(x_1, x_2) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$. Rita upp funktionsytan och nivåkurvor, så att eventuella lokala maximi- och minimipunkter samt sadelpunkter blir synliga.

3 Newtons metod

Vi har sett på Newtons metod för att lösa system av icke-linjära ekvationer $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Villkoret för en stationär punkt $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$ är ett sådant ekvationssystem.

Vi kan alltså beräkna stationär punkt till en funktion $f(\boldsymbol{x})$ genom att försöka lösa ekvationen $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$, där $\boldsymbol{g} : \mathbb{R}^2 \to \mathbb{R}^2$ med

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\nabla} f(\boldsymbol{x}) = \begin{bmatrix} f'_{x_1}(\boldsymbol{x}) \\ f'_{x_2}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} f'_{x_1}(x_1, x_2) \\ f'_{x_2}(x_1, x_2) \end{bmatrix}$$

med Newtons metod. Antag att vi har en approximation $\boldsymbol{x}_k$ av en stationär punkt, dvs. en approximation av lösningen till $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$. Vi bildar nästa approximation av lösningen:

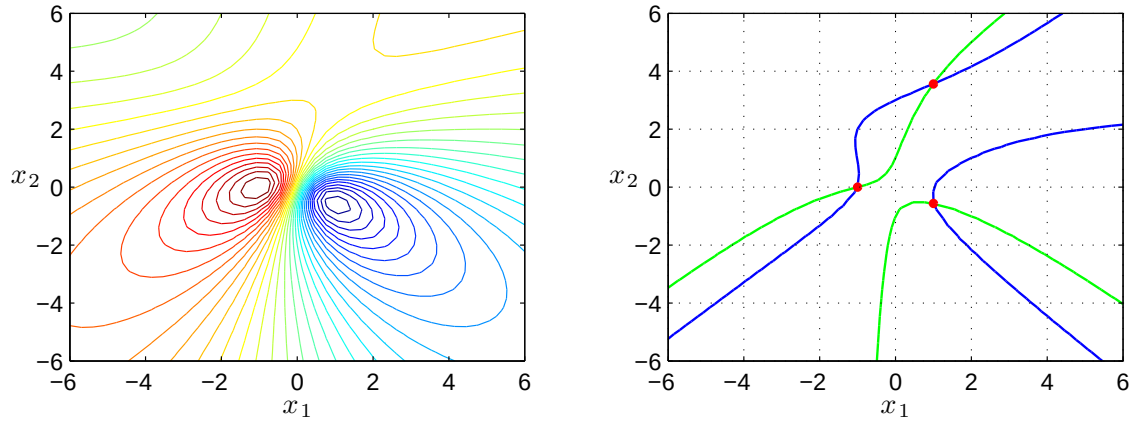
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h}$$

där $\boldsymbol{h}$ är lösning till det linjära ekvationssystemet

$$\boldsymbol{D}\boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{h} = -\boldsymbol{g}(\boldsymbol{x}_k)$$

Här är Jacobimatrisen $\boldsymbol{Dg}(\boldsymbol{x}_k)$ är en $n \times n$-matris. (Jacobimatrisen till $\boldsymbol{g}$ är faktiskt Hessianmatrisen till f.)

Åter till vårt exempel. Nu måste vi finna bra startapproximationer. Vi har redan en graf av nivåkurvorna till funktionen, men för att lite noggrannare se var de stationära punkterna ligger ritar vi också noll-nivåkurvor till de två komponenterna i gradientvektorn. Sedan kan vi läsa av startapproximationer av de stationära punkterna och bestämma dem noggrant med Newtons metod.

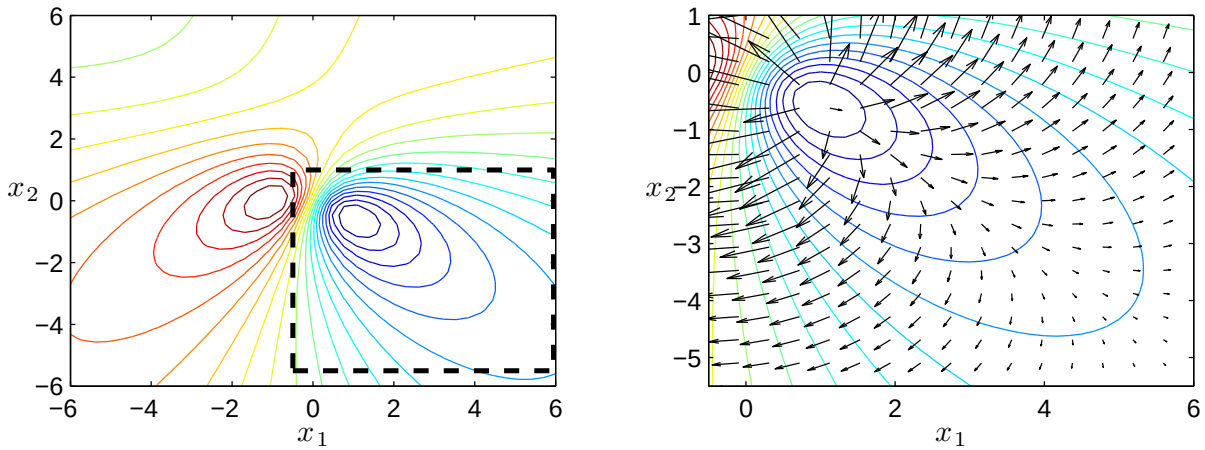


Uppgift 2. Vi återvänder till funktionen $f(x_1, x_2) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$ från uppgift 1. Beräkna nu alla stationära punkter noggrant med Newtons metod, dvs. lös ekvationen $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$. Därefter bestämmer du vilken typ av stationära punkter vi har genom att studera egenvärdena till Hessianmatrisen $\boldsymbol{H}(\boldsymbol{x})$.

4 Steepest descentmetoden

En funktion växer som ni vet snabbast i gradientens riktning och minskar snabbast i negativa gradientens riktning. Vi skall söka efter minimipunkter genom att utgående från en startapproximation röra oss i negativa gradientens riktning för att komma ned så snabbt som möjligt längs funktionsytan ungefär som vi kan låta en snöboll rulla ut för en sluttning.

Vi ser nedan till vänster nivåkurvorna till vår funktion. Området runt minimipunkten ser vi uppförstorat till höger. Där har vi även ritat ut gradienterna på några ställen. Vi ser att dessa pekar mot det håll funktionen växer som snabbast och vi skall alltså gå i motsatta riktningarna.



Vi beskriver nu Steepest descentmetoden. Antag att vi har en approximation $\boldsymbol{x}_k$ av en lokal minimipunkt, bilda då nästa approximation enligt:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - s_k \boldsymbol{\nabla} f(\boldsymbol{x}_k)$$

där s_k är en konstant som avgör hur långt vi går i riktningen $-\boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{x}_k)$.

Vi väljer s_k så att

$$f(\boldsymbol{x}_k - s_k \boldsymbol{\nabla} f(\boldsymbol{x}_k)) < f(\boldsymbol{x}_k)$$

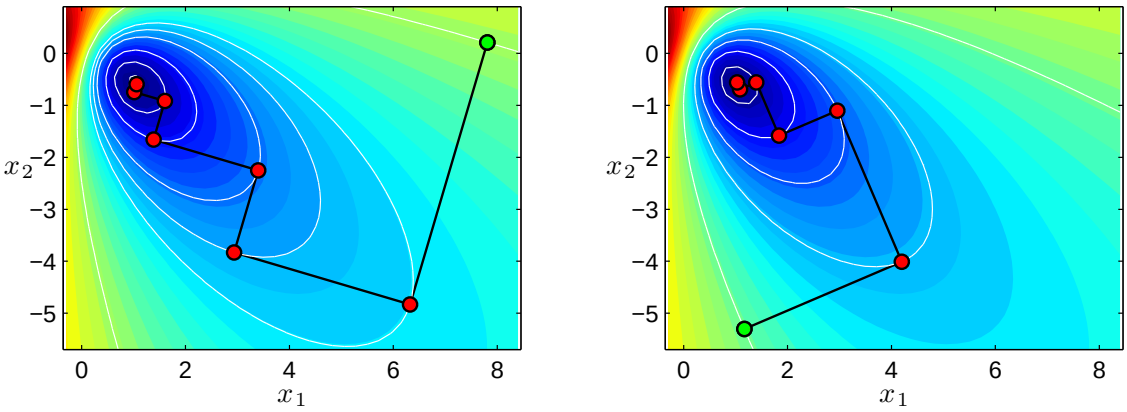
vilket garanterar en minskning av funktionsvärdet.

Ett bästa (optimalt) val av s_k är sådant att

$$g(s) = f(\boldsymbol{x}_k - s \boldsymbol{\nabla} f(\boldsymbol{x}_k))$$

minimeras, vilket leder till ett optimeringsproblem i variabeln s.

Vi ser hur det går då vi använder Steepest descentmetoden med optimalt s_k-värde för två olika startapproximationer av minimipunkten. Riktigt dåliga approximationer så att vi skall få se hur metoden stegar sig fram.



Startpunkten $\boldsymbol{x}_0$ är markerad med grönt och följande punkter $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots$ med rött. Vi lämnar $\boldsymbol{x}_k$ med rät vinkel mot nivåkurvan genom punkten (gradienten vinkelrät mot nivåkurvan) och tar ett steg längs $-\boldsymbol{\nabla} f(\boldsymbol{x}_k)$ tills vi tangerar en nivåkurva (sedan börjar funktionen växa igen), där har vi nästa approximation $\boldsymbol{x}_{k+1}$.

Vill man istället söka en lokal maximipunkt går man antingen i positiv gradientriktning (steepest ascent) eller tillämpar steepest descent på $-f(\boldsymbol{x})$.

Uppgift 3. Betrakta funktionen $f(x_1, x_2) = x_1^2 x_2 e^{-(x_1^2+x_2^2)}$. Rita upp funktionsytan och nivåkurvor, så att eventuella lokala maximi- och minimipunkter samt sadelpunkter blir synliga. Använd sedan Steepest descentmetoden för att beräkna lokala maximi- och minimipunkter. Välj steglängd genom att minimera $g(s) = f(\boldsymbol{x}_k - s \boldsymbol{\nabla} f(\boldsymbol{x}_k))$ med `fminbnd`. Markera mellanresultaten med små ringar så vi ser hur metoden stegar sig fram. Läs hjälptexten för `fminbnd` allra först.

5 Optimering utan bivillkor i MATLAB

I OPTIMIZATION TOOLBOX i MATLAB finns `fsolve` för att lösa icke-linjära ekvationssystem och `fminunc` för minimering av funktioner i flera variabler. För funktioner vi vill minimera men som inte är derivarbara finns `fminsearch`. Vill vi minimera en funktion i en enda variabel använder vi `fminbnd`.

Minimering av funktionen i uppgift 3 med `fminunc` kan se ut så här:

```
>> f=@(x)x(1)^2*x(2)*exp(-(x(1)^2+x(2)^2));
>> x0=[1;-1];
>> x=fminunc(f,x0)
```

Uppgift 4. Betrakta funktionen $f(x_1, x_2) = x_1^3 + 3x_1x_2^2 - 15x_1 + x_2^2 - 15x_2$. Rita upp funktionsytan och nivåkurvor, så att eventuella lokala maximi- och minimipunkter samt sadelpunkter blir synliga. Bestäm alla stationära punkter till f med `fsolve` och klassificera punkterna som maximi-, minimi- eller sadelpunkt med hjälp av Hessianmatrisen. Bestäm gradientvektorn och Hessianmatrisen för hand. Gör de återstående beräkningarna med MATLAB. Ange startpunkter med kommandot `ginput`. Bestäm slutligen minimipunkten med `fminunc`.

6 Optimering med bivillkor

Vi skall se på lösning av optimeringsproblem med bivillkor

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}) \quad \text{eller} \quad \max_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$$

där är Ω en mängd som begränsas av bivillkor.

Vi antar att mängden Ω kan beskrivas på följande sätt.

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}\}$$

där $\boldsymbol{g} : \mathbb{R}^n \mapsto \mathbb{R}^m$ och $\boldsymbol{h} : \mathbb{R}^n \mapsto \mathbb{R}^q$.

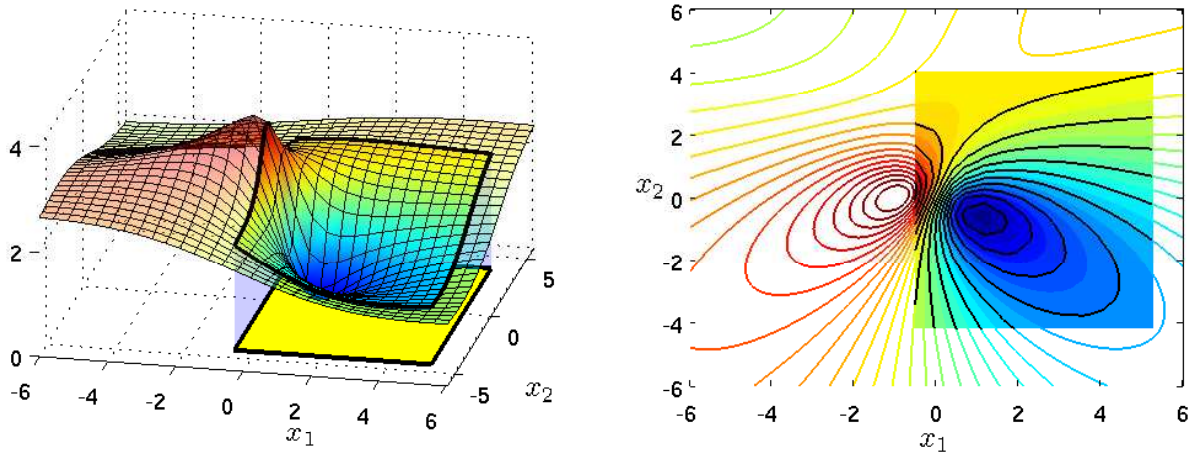
Här kallas f ofta för *objektfunktion* och Ω för *tillåtna mängden*. Bivillkoret och $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ kallas *likhetsbivillkor* och $\boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}$ kallas *olikhetsbivillkor*.

Som exempel ser vi på minimering eller maximering av

$$f(\boldsymbol{x}) = \frac{x_2 - 3\,x_1 + x_1\,x_2}{1 + x_1^2 + x_2^2}$$

då Ω är ett rektangulärt område.

Vi ritar en figur där vi ser funktionsyta och nivåkurvor samt tillåtna mängden.



Vi ser att vi måste undersöka stationära punkter, men också titta på randen av området. Det ser ut som vi har en minimipunkt i det inre av Ω och en maximipunkt på randen av Ω.

Vi kommer inte göra så mycket mer åt denna problemställning, utan nöjer oss med att se vad man kan göra med färdiga program genom att se på ett exempel.

7 Optimering med bivillkor i MATLAB

I OPTIMIZATION TOOLBOX finns funktionen `fmincon` som är avsedd för optimeringsproblem med icke-linjär objektfunktion och icke-linjära bivillkor.

Objektfunktionen måste beskrivas som en funktion, samma gäller för de icke-linjära bivillkoren.

Bivillkor som är linjära beskrivs med matriser och vektorer, $\boldsymbol{Ax} \leq \boldsymbol{c}$ för olikhetsbivillkor och $\boldsymbol{Bx} = \boldsymbol{d}$ för likhetsbivillkor.

Med $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ ges enkla gränser på variablerna ($\boldsymbol{l}$ och $\boldsymbol{u}$ är vektorer). Skulle en variabel x_i inte vara begränsad nedåt (uppåt) så låter vi $l_i = -\infty$ ($u_i = +\infty$).

Vi måste också ge en startapproximation och `fmincon` används enligt

```
>> x=fmincon(@fun,x0,A,c,B,d,l,u,@nonlcon)
```

Bivillkor som inte finns med i ett aktuellt problem ersätts med tomma mängden.

Som exempel tar vi: Minimera plåtåtgången vid tillverkningen av en burk med radien x_1 och höjden x_2, givet att den skall rymma volymen $V = 1$.

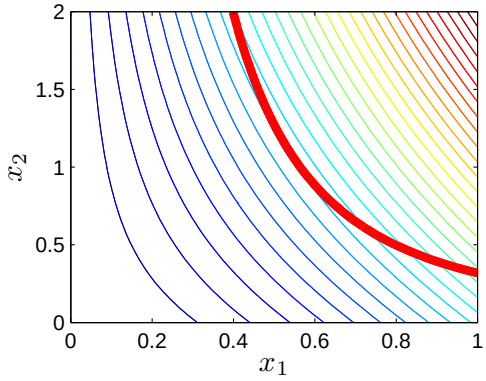
Arean av burkens yta blir $A = 2\pi x_1 x_2 + 2\pi x_1^2$ och dess volym blir $V = \pi x_1^2 x_2$. Vi skall alltså lösa

$$\min_{h(\boldsymbol{x})=0} f(\boldsymbol{x})$$

där $f(\boldsymbol{x}) = 2\pi x_1 x_2 + 2\pi x_1^2$ och $h(\boldsymbol{x}) = \pi x_1^2 x_2 - 1$.

Vi ritar en bild med nivåkurvor till objektfunktionen samt noll-nivåkurvan till bivillkoret, så att vi ser var bivillkoret är uppfyllt.

```
>> x1=linspace(0,1,40); x2=linspace(0,2,40);
>> [X1, X2]=meshgrid(x1,x2);
>> F=2*pi*X1.*(X1+X2);
>> H=pi*X1.^2.*X2-1;
>> contour(x1,x2,F,30), hold on
>> contour(x1,x2,H,[0 0],'r','linewidth',4), hold off
>> xlabel('x_1'), ylabel('x_2')
```



Vi ser att $0.5 \leq x_1 \leq 0.6$ och $0.8 \leq x_2 \leq 1.2$, det blir våra enkla gränser.

Nu beskriver vi objektfunktion och bivillkor enligt

```
function f=fun_burk(x)
f=2*pi*x(1)*(x(1)+x(2));

function [g,h]=con_burk(x)
g=[];                  % Vi har inget olikhetsbivillkor
h=pi*x(1)^2*x(2)-1;
```

Läser av en approximation av de optimala värdena på radien x_1 och höjden x_2 och löser sedan problemet noggrant med `fmincon`

```
>> x0=ginput(1);
>> l=[0.5;0.8]; u=[0.6;1.2];      % Enkla gränser för x
>> x=fmincon(@fun_burk,x0,[],[],[],[],l,u,@con_burk)
x =
    0.5419    1.0839
```

7.1 Linjärprogrammering LP

Vi skall se hur man med MATLAB löser s.k. linjärprogrammeringsproblem eller LP-problem

$$\min_{\boldsymbol{x}} \phi(\boldsymbol{x}) = \boldsymbol{f}^T\boldsymbol{x}$$

då $\boldsymbol{x}$ uppfyller bivillkoren $\boldsymbol{Ax} \leq \boldsymbol{c}$, $\boldsymbol{Bx} = \boldsymbol{d}$ och $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

Objektfunktionen $\phi(\boldsymbol{x}) = \boldsymbol{f}^T\boldsymbol{x}$ är linjär och vi har endast linjära bivillkor $\boldsymbol{Ax} \leq \boldsymbol{c}$ och $\boldsymbol{Bx} = \boldsymbol{d}$ samt enkla gränser på variablerna $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

Ofta skriver man LP-problemet på följande form

$$\begin{cases} \min_{\boldsymbol{x}} \phi = \boldsymbol{f}^T\boldsymbol{x} \\ \boldsymbol{Ax} \leq \boldsymbol{c} \\ \boldsymbol{Bx} = \boldsymbol{d} \\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \end{cases}$$

och man löser det med `linprog` i MATLAB enligt

```
>> x=linprog(f,A,c,B,d,l,u)
```

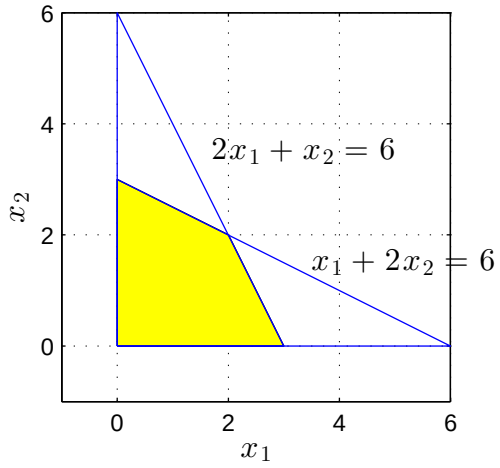
Vi ser på exempel 4 i Adams kapitel 13.2 där man löser problemet

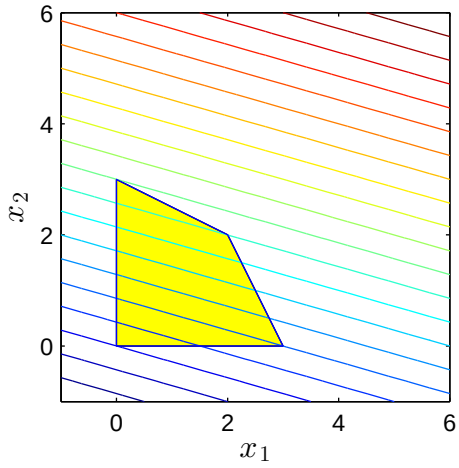
$$\begin{cases} \max_{\boldsymbol{x}} z = 2x_1 + 7x_2 \\ x_1 + 2x_2 \leq 6 \\ 2x_1 + x_2 \leq 6 \\ 0 \leq x_1,\ 0 \leq x_2 \end{cases}$$

Vi får ta $\phi = -z = -(2x_1 + 7x_2)$ eftersom vi minimerar med `linprog`.

```
>> f=-[2; 7];
>> A=[1 2; 2 1]; c=[6; 6];
>> l=[0; 0]; u=[Inf; Inf];
>> x=linprog(f,A,c,[],[],l,u)
x =
    0.0000     +3.0000
```

Låt oss rita upp den tillåtna mängden samt nivåkurvor till objektfunktionen.





7.2 Kvadratisk programmering QP

Vi skall se hur man med MATLAB löser QP-problem

$$\min_{\boldsymbol{x}} \phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}$$

då $\boldsymbol{x}$ uppfyller bivillkoren $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}$, $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{d}$ och $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

Matrisen $\boldsymbol{H}$ i objektfunktionen $\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}$ är en symmetrisk matris. Bivillkoren är av samma typ som för LP-problemet. Skillnaden är alltså objektfunktionen som är kvadratisk och inte linjär.

Ofta skriver man problemet på följande form

$$\begin{cases} \min_{\boldsymbol{x}} \phi = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} \\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c} \\ \boldsymbol{B}\boldsymbol{x} = \boldsymbol{d} \\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \end{cases}$$

och man löser det med `quadprog` i MATLAB enligt

```
>> x=quadprog(H,f,A,c,B,d,l,u)
```

Vi ser på exemplet

$$\begin{cases} \min_{\boldsymbol{x}} \phi = x_1^2 + 4x_2^2 - 2x_1x_2 + 5x_2 \\ x_1 + x_2 = 1 \end{cases}$$

som vi löser enligt

```
>> H=[2 -2;-2 8];
>> f=[0;5];
>> B=[1 1]; d=1;
>> l=[-Inf; -Inf]; u=[Inf; Inf];
>> x=quadprog(H,f,[],[],B,d,l,u)
x =
    1.0714   -0.0714
```

1 Målsättning

Avsikten med denna laboration är att ge kunskap om hur man löser optimeringsproblem utan bivillkor med metoder som Newtons metod eller Steepest descentmetoden samt med färdiga program i MATLAB. Vidare är avsikten att ge en första introduktion om hur man löser optimeringsproblem med bivillkor.

2 Kommentarer och förklaringar

Den kvadratisk modellen får vi från Taylorutvecklingen

$$f(x,y) = f(a,b) + f'_x(a,b)h + f'_y(a,b)k + \frac{1}{2}\left(f''_{xx}(a,b)h^2 + 2f''_{xy}(a,b)hk + f''_{yy}(a,b)k^2\right) + \cdots$$

Om vi samlar ihop förstaderivator i en radmatris (Jacobianen), andraderivator i en kvadratisk matris (Hessianen) samt h och k i en kolonnvektor så får vi

$$f(x,y) = f(a,b) + \begin{bmatrix} f'_x(a,b) & f'_y(a,b) \end{bmatrix} \begin{bmatrix} h \\ k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} h & k \end{bmatrix} \begin{bmatrix} f''_{xx}(a,b) & f''_{xy}(a,b) \\ f''_{yx}(a,b) & f''_{yy}(a,b) \end{bmatrix} \begin{bmatrix} h \\ k \end{bmatrix} + \cdots$$

eller

$$f(x,y) = f(a,b) + \boldsymbol{\nabla} f(a,b)^T \begin{bmatrix} h \\ k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} h & k \end{bmatrix} \boldsymbol{H}(a,b) \begin{bmatrix} h \\ k \end{bmatrix} + \cdots$$

Med vektorbeteckningar $\boldsymbol{x} = (x,y)$, $\boldsymbol{a} = (a,b)$ och $\boldsymbol{x}-\boldsymbol{a} = (x-a, y-b) = (h,k)$ kan detta skrivas

$$f(\boldsymbol{x}) = f(\boldsymbol{a}) + \boldsymbol{\nabla} f(\boldsymbol{a})^T(\boldsymbol{x}-\boldsymbol{a}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{a})^T \boldsymbol{H}(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}) + \cdots$$

Om $\boldsymbol{a}$ är en stationär punkt så är $\boldsymbol{\nabla} f(\boldsymbol{a}) = \boldsymbol{0}$ och vi får att

$$f(\boldsymbol{x}) \approx f(\boldsymbol{a}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{a})^T \boldsymbol{H}(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a})$$

Genom att beräkna egenvärdena till $\boldsymbol{H}(\boldsymbol{a})$ kan vi avgöra vilken typ av stationär punkt vi har (se Adams s. 746).

3 Fördjupning

En speciell typ av minimeringsproblem är anpassning av en icke-linjär modell $f(\boldsymbol{x}, t)$, till en serie mätdata $(t_i, y_i),\ i = 1, \cdots, m$. I kursen i linjär algebra tittade vi på anpassning av linjära modeller till mätdata. Då använde vi minsta-kvadratmetoden för att få den bästa anpassningen.

Vårt problem blir nu att minimera

$$r(\boldsymbol{x}) = \sum_{i=1}^{m} (f(\boldsymbol{x}, t_i) - y_i)^2$$

vilket är ett icke-linjärt minsta-kvadratproblem.

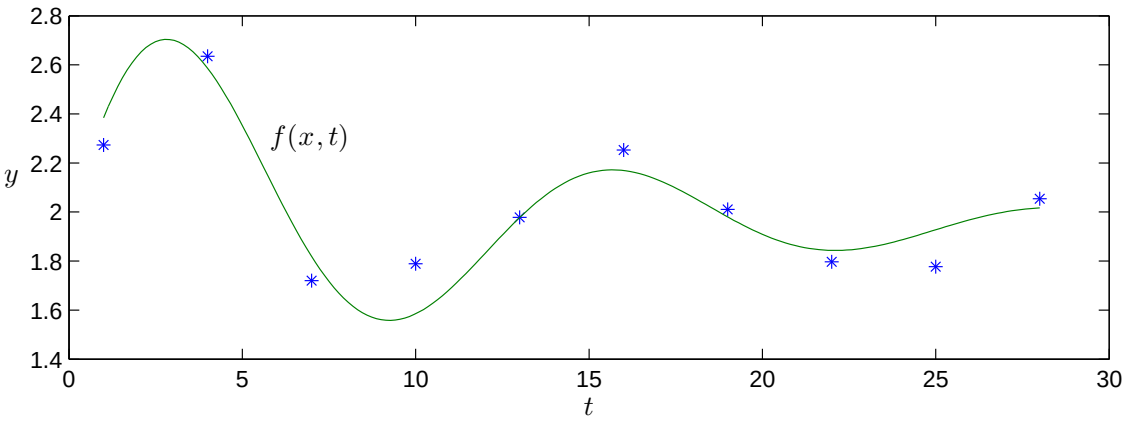
Det finns en funktion `lsqcurvefit` i Optimization Toolbox som bygger på att man ser problemet som ett överbestämt system av icke-linjära ekvationer. Som exempel tar vi: Vi har en mätserie av ett insvägningsförlopp som vi vill beskriva med följande modell

$$f(\boldsymbol{x}, t) = x_1 + \sin(x_2 t)\, \mathrm{e}^{-x_3 t}$$

Antag vi har en mätserie lagrad i två vektorer `t` och `y`. Vi ritar upp mätserien och med ledning av modellens egenskaper och mätdata bildar vi en skattning av parametervärdena, dvs. värdena på $\boldsymbol{x} = (x_1, x_2, x_3)$, och tar detta som startapproximation. Sedan löser vi med `lsqcurvefit` och ritar upp modellen.

```
>> x0=[2.0 0.4 0.1];
>> f=@(x,t)x(1)+sin(x(2)*t).*exp(-x(3)*t);
>> x=lsqcurvefit(f,x0,t,y)
x =
    1.9578     0.4887     0.0972

>> te=linspace(t(1),t(end),200);     % täta t-värden ger snygg graf
>> plot(t,y,'*',te,f(x,te))          % rita data och modell
>> gtext('f(x,t)'), xlabel('t'), ylabel('y')
```



4 Lärandemål

Efter denna laboration skall du

- kunna bestämma stationära punkter med Newtons metod och avgöra de stationära punkternas typ genom att beräkna egenvärden till Hessianen med `eig`
- kunna tillämpa Steepest descentmetoden på ett enklare minimeringsproblem samt känna till metodens geometriska egenskaper
- kunna använda `fminunc` för att lösa optimeringssproblem utan bivillkor och ha kännedom om funktioner i Matlab för optimering med bivillkor