Optimering

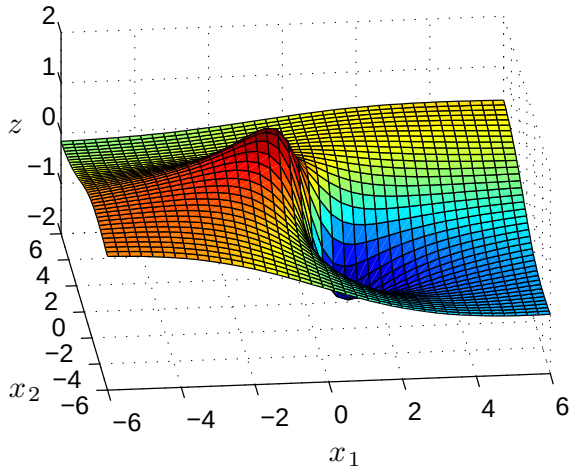
Analys och Linjär Algebra, del C, K1/Kf1/Bt1, vt10

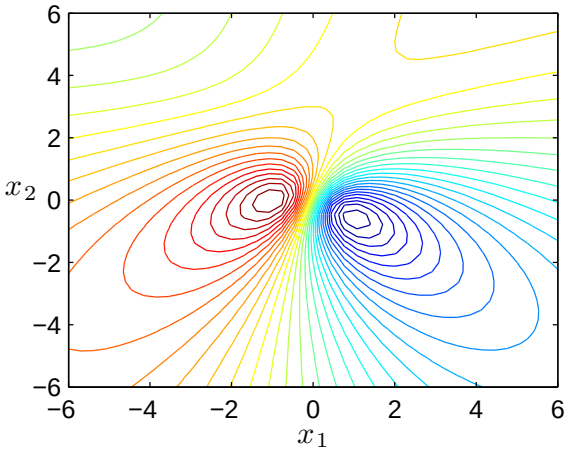
1 Inledning

Vi skall bestämma maximi- och minimipunkter samt sadelpunkter till en funktion $f : \mathbf{R}^2 \to \mathbf{R}$. 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 $\boldsymbol{\nabla} f(\boldsymbol{a}) = \mathbf{0}$. Bestämningen av vilken typ av stationär punkt det är kan vi göra med hjälp av egenvärdena till Hessianmatrisen.

I den förra studioövningen 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 $\nabla f(\boldsymbol{a}) = \boldsymbol{0}$ 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. Avslutningsvis skall vi beskriva de i MATLAB inbyggda optimeringsrutinerna. Men först en liten övning.

Övning 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.

2 Newtons metod

I förra studioövningen generaliserade vi 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 stationära punkter till en funktion $f(\boldsymbol{x})$ genom att försöka lösa ekvationen $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$, där

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

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:

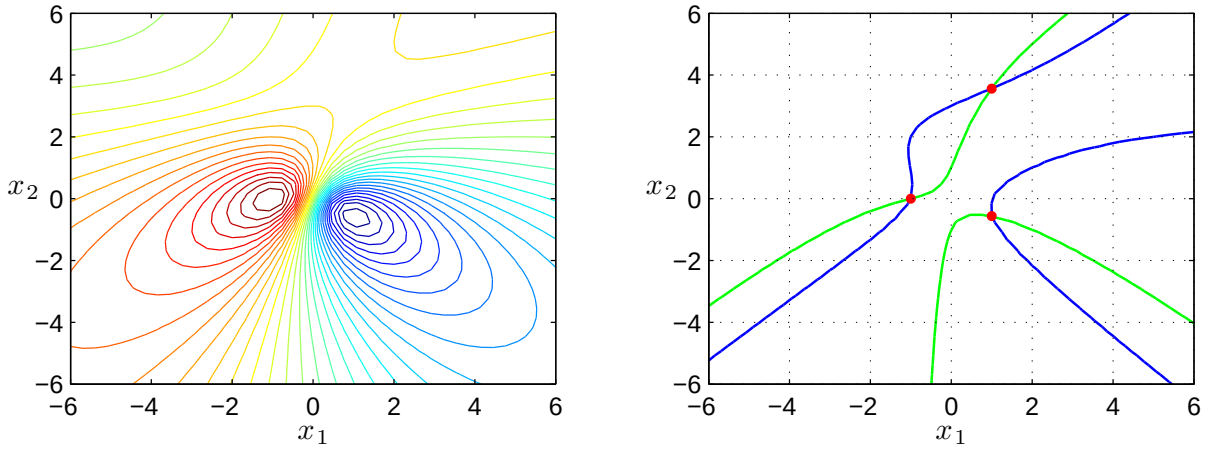
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h},$$

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



Nu kan vi läsa av startapproximationer av de stationära punkterna och bestämma dem noggrannt med Newtons metod.

Övning 2. Vi återvänder till funktionen $f(x_1, x_2) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$ i övning 1.

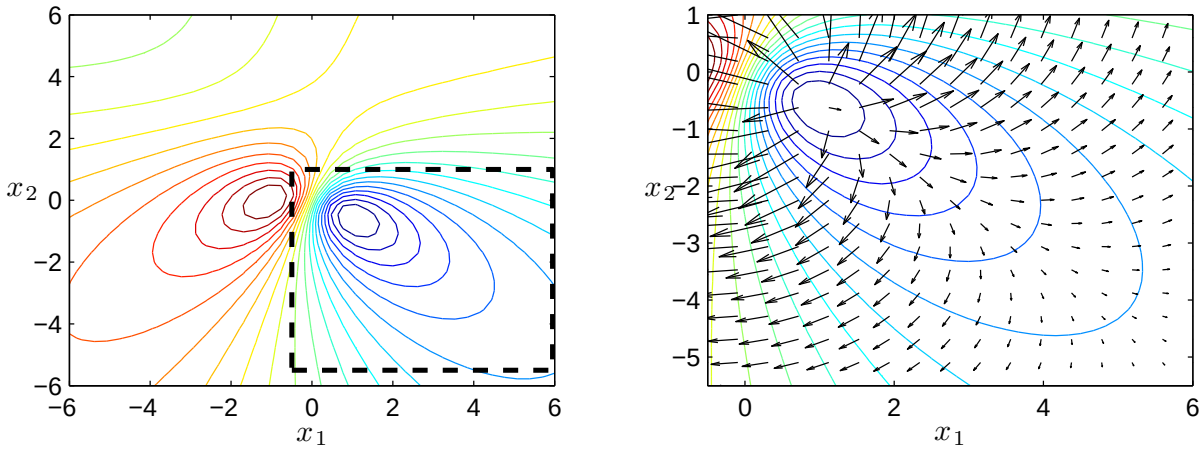
a) Beräkna alla stationära punkter noggrannt med Newtons metod,
dvs. lös ekvationen $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$.

b) Bestäm vilken typ av stationära punkter vi har genom att studera egenvärdena till Hessianmatrisen $\boldsymbol{H}(\boldsymbol{x})$.

3 Steepest descentmetoden

En funktion växer som 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 de 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, vi bildar nästa approximation:

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

där α_k en konstant som avgör hur långt vi går i riktningen $-\nabla \boldsymbol{f}(\boldsymbol{x}_k)$.
Ett bästa val av α_k är sådant att

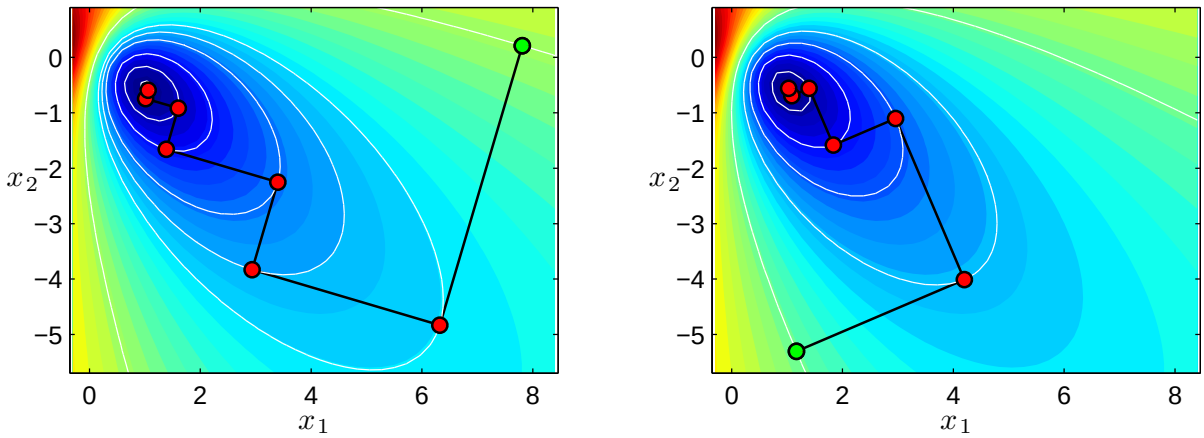
$$g(s) = f(\boldsymbol{x}_k - s\boldsymbol{\nabla} f(\boldsymbol{x}_k))$$

minimeras, vilket leder till ett en-dimensionellt optimeringsproblem. Man kan också välja ett α_k så att

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

vilket garanterar en minskning av funktionsvärdet.

Vi ser hur det går då vi använder Steepest descentmetoden med optimalt α_k-värde för två olika startapproximationer av minimipunkten, (vi väljer ”dåliga” startapproximationer för att bättre kunna 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 (varför?) och tar ett steg längs $-\nabla f(\boldsymbol{x}_k)$ tills

vi tangerar en nivåkurva (varför?), där har vi nästa approximation $\boldsymbol{x}_{k+1}$. Detta får du reda ut i övning 4 nedan.

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

På hemsidan finns filerna `SD.m` och `Jacobi.m`. I filen `SD.m` utförs beräkningar enligt Steepest descentmetoden och i `Jacobi.m` beräknas en approximation till gradientvektorn.

Övning 3. Betrakta funktionen $f(x_1, x_2) = x_1^2 \cdot x_2 \cdot e^{-(x_1^2+x_2^2)}$.

a) Rita upp funktionsytan och nivåkurvor, så att eventuella lokala maximi- och minimipunkter samt sadelpunkter blir synliga.

b) Använd Steepest descentmetoden för att beräkna lokala maximi- och minimipunkter.

4 Optimeringsprogram i MATLAB

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

Ett anrop av `fminunc` kan se ut så här:

```
>>  x = fminunc(@(x)x(1)^2+x(2)^2,[1;1])
```

vilket minimerar funktionen $f(x_1, x_2) = x_1^2 + x_2^2$ från punkten $(1, 1)$, eller

```
>>  x0 =  ginput(1)
>>  x=fminunc(@funk,x0')
```

som minimerar funktionen `funk` från en punkt som ges av ett klick från musen. Här är `funk.m` en MATLABfunktion, till exempel

```
function f=funk(x)
f = x(1)^2+x(2)^2;
```

Övning 4. Vi gör här en utförligare undersökning av funktionen från Studio 1,

$$f(x,y) = x^3 + 3xy^2 - 15x + y^2 - 15y.$$

a) Bestäm alla stationära punkter till f och klassificera punkterna som maximi-, minimi- eller sadelpunkt med hjälp av Hessianmatrisen. Bestäm gradientvektorn och Hessianmatrisen för hand. Gör de återstående beräkningarna med MATLAB.

b) Plotta funktionen och gör en konturplot, som är så pass detaljerad att du ser var eventuella lokala maximi-, minimi- och sadelpunkter finns. Jämför med vad du kom fram till i (a).

c) Funktionen har en minimipunkt. Bestäm denna med `SD.m`. Klicka in en startpunkt till `SD.m` med MATLABs kommando `ginput`.

d) Bestäm minimipunkten med `fminunc`. Jämför med resultatet i (c).

e) Beskriv med penna och papper hur Steepest descentmetoden fungerar. Använd Dig av gradientvektorer och nivåkurvor.