MATLAB övningsuppgifter

1 Inledning

Vi skall först se hur man kan lösa system av icke-linjära ekvationer. Därefter skall vi se på optimering utan bivillkor. Vi skall bestämma maximi- och minimipunkter. Avslutningsvis skall vi se på dubbelintegraler.

2 System av icke-linjära ekvationer

Vi skall lösa system av icke-linjära ekvationer. Som exempel kan vi ta,

$$\begin{cases} x_1\left(1+x_2^2\right)-1=0 \\ x_2\left(1+x_1^2\right)-2=0 \end{cases}$$

som är ett system av två ekvationer i två obekanta. Om vi inför de två funktionerna

$$f_1(x_1,x_2)=x_1\big(1+x_2^2\big)-1,$$
$$f_2(x_1,x_2)=x_2\big(1+x_1^2\big)-2,$$

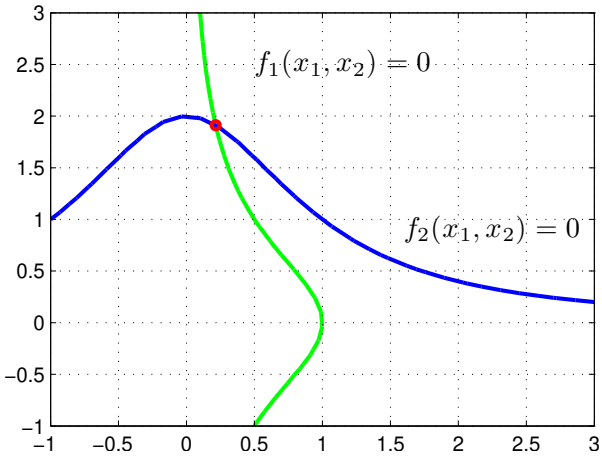
kan ekvationssystemet skrivas

$$\begin{cases} f_1(x_1,x_2)=0 \\ f_2(x_1,x_2)=0 \end{cases}$$

Med $\boldsymbol{f}=(f_1,f_2)$, $\boldsymbol{x}=(x_1,x_2)$ och $\mathbf{0}=(0,0)$, ser vi att $\boldsymbol{f}\colon\mathbb{R}^2\to\mathbb{R}^2$ och vi kan skriva ekvationerna på den kompakta formen

$$\boldsymbol{f}(\boldsymbol{x})=\mathbf{0}.$$

Vi skall se hur vi kan använda Newtons metod för att lösa sådana ekvationer. Men vi börjar med att rita upp noll-nivåkurvorna till f_1 respektive f_2.



Vi ser lösningen till $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ som den punkt där noll-nivåkurvorna till f_1 och f_2 skär varandra. Man kan grafiskt läsa av en första approximation av lösningen för att sedan förbättra denna med Newtons metod, som vi snart skall beskriva. Men först lite om linjärisering.

Linjärisering

Låt $f(x)$ vara en differentierbar (deriverbar) funktion i en variabel, $f : \mathbb{R} \to \mathbb{R}$. Linjäriseringen av f runt punkten a ges av

$$L(x) = f(a) + f'(a)(x - a).$$

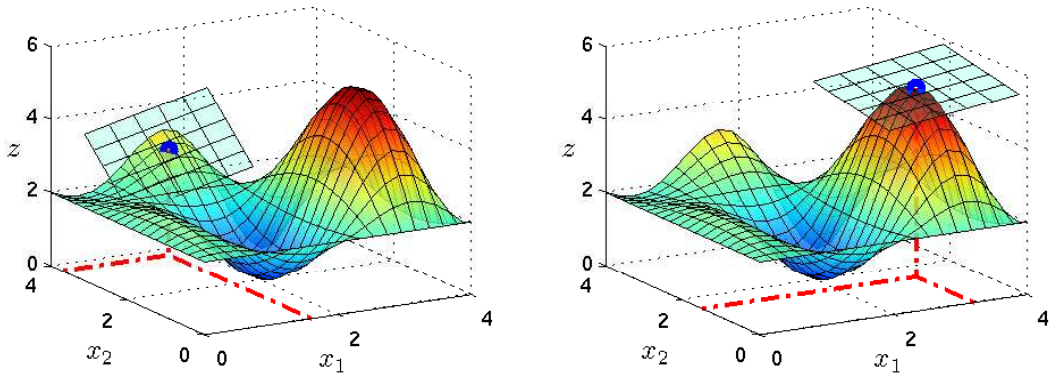
Nära punkten a har vi $f(x) \approx L(x)$. Vi påminner oss också om att den räta linjen $y = L(x)$ är tangenten till kurvan $y = f(x)$ vid a.

Låt $f(x_1, x_2)$ vara en differentierbar funktion i två variabler, $f : \mathbb{R}^2 \to \mathbb{R}$. Linjäriseringen av f runt punkten (a_1, a_2) ges av

$$L(x_1, x_2) = f(a_1, a_2) + \frac{\partial f}{\partial x_1}(a_1, a_2)(x_1 - a_1) + \frac{\partial f}{\partial x_2}(a_1, a_2)(x_2 - a_2)$$

och nära punkten (a_1, a_2) har vi $f(x_1, x_2) \approx L(x_1, x_2)$.

Det räta planet $z = L(x_1, x_2)$ är tangentplanet till ytan $z = f(x_1, x_2)$ vid punkten (a_1, a_2).



Vi låter $\boldsymbol{x} = (x_1, x_2)$, $f(\boldsymbol{x}) = f(x_1, x_2)$ och $\boldsymbol{a} = (a_1, a_2)$. Använder vi gradienten

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

kan vi skriva linjäriseringen

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

Lägg märke till att $\boldsymbol{\nabla} f(\boldsymbol{a})$ och $\boldsymbol{x} - \boldsymbol{a}$ är kolonnvektorer så $\boldsymbol{\nabla} f(\boldsymbol{a})^T (\boldsymbol{x} - \boldsymbol{a})$ är skalärprodukten $\boldsymbol{\nabla} f(\boldsymbol{a}) \cdot (\boldsymbol{x} - \boldsymbol{a})$.

Låt oss som exempel ta $f(\boldsymbol{x}) = x_1(1 + x_2^2) - 1$. Då gäller

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

och därmed $\boldsymbol{\nabla} f(\boldsymbol{x})^T = \begin{bmatrix} 1+x_2^2 & 2x_1x_2 \end{bmatrix}$. Linjäriseringen vid $\boldsymbol{a} = (2,1)$ blir

$$L(\boldsymbol{x}) = f(\boldsymbol{a}) + \boldsymbol{\nabla} f(\boldsymbol{a})^T(\boldsymbol{x}-\boldsymbol{a}) = 3 + \begin{bmatrix} 2 & 4 \end{bmatrix} \begin{bmatrix} x_1-2 \\ x_2-1 \end{bmatrix}.$$

Låt $f_1(x_1,x_2)$ och $f_2(x_1,x_2)$ vara två differentierbara funktioner i två variabler, $f_1 : \mathbb{R}^2 \to \mathbb{R}$ och $f_2 : \mathbb{R}^2 \to \mathbb{R}$. Linjäriseringen av f_1 respektive f_2 runt punkten (a_1,a_2) ges av

$$\begin{aligned} L_1(x_1,x_2) &= f_1(a_1,a_2) + \frac{\partial f_1}{\partial x_1}(a_1,a_2)(x_1-a_1) + \frac{\partial f_1}{\partial x_2}(a_1,a_2)(x_2-a_2), \\ L_2(x_1,x_2) &= f_2(a_1,a_2) + \frac{\partial f_2}{\partial x_1}(a_1,a_2)(x_1-a_1) + \frac{\partial f_2}{\partial x_2}(a_1,a_2)(x_2-a_2), \end{aligned}$$

Låter vi $\boldsymbol{x} = (x_1,x_2)$, $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ och $\boldsymbol{a} = (a_1,a_2)$ kan vi skriva

$$\begin{aligned} L_1(\boldsymbol{x}) &= f_1(\boldsymbol{a}) + \boldsymbol{\nabla} f_1(\boldsymbol{a})^T(\boldsymbol{x}-\boldsymbol{a}), \\ L_2(\boldsymbol{x}) &= f_2(\boldsymbol{a}) + \boldsymbol{\nabla} f_2(\boldsymbol{a})^T(\boldsymbol{x}-\boldsymbol{a}), \end{aligned}$$

eller med matrisbeteckningar

$$\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{a}) + \boldsymbol{f}'(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}),$$

där $\boldsymbol{L}(\boldsymbol{x}) = (L_1(\boldsymbol{x}), L_2(\boldsymbol{x}))$ och

$$\boldsymbol{f}'(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(\boldsymbol{x}) & \dfrac{\partial f_1}{\partial x_2}(\boldsymbol{x}) \\ \dfrac{\partial f_2}{\partial x_1}(\boldsymbol{x}) & \dfrac{\partial f_2}{\partial x_2}(\boldsymbol{x}) \end{bmatrix}$$

är den s.k. Jacobimatrisen eller funktionalmatrisen.

Låt oss som exempel ta $\boldsymbol{f}(\boldsymbol{x}) = (x_1(1+x_2^2)-1, x_2(1+x_1^2)-2)$. Då gäller

$$\begin{aligned} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}) &= 1+x_2^2, \quad \frac{\partial f_1}{\partial x_2}(\boldsymbol{x}) = 2x_1x_2, \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{x}) &= 2x_1x_2, \quad \frac{\partial f_2}{\partial x_2}(\boldsymbol{x}) = 1+x_1^2, \end{aligned}$$

och därmed

$$\boldsymbol{f}'(\boldsymbol{x}) = \begin{bmatrix} 1+x_2^2 & 2x_1x_2 \\ 2x_1x_2 & 1+x_1^2 \end{bmatrix}.$$

Linjäriseringen vid $\boldsymbol{a} = (2,1)$ blir

$$\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{a}) + \boldsymbol{f}'(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}) = \begin{bmatrix} 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} x_1-2 \\ x_2-1 \end{bmatrix}.$$

Vi generaliserar nu till godtyckligt antal funktioner i godtyckligt många variabler. Låt f_i beteckna m funktioner i n variabler, dvs. $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$. Vi låter

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_m(x_1,\ldots,x_n) \end{bmatrix},$$

och

$$\boldsymbol{f}'(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(\boldsymbol{x}) & \dots & \dfrac{\partial f_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial x_1}(\boldsymbol{x}) & \dots & \dfrac{\partial f_m}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}.$$

Här är $m \times n$ matrisen $\boldsymbol{f}'(\boldsymbol{x})$ Jacobimatrisen (funktionalmatrisen) av $\boldsymbol{f}$ i $\boldsymbol{x}$. Linjäriseringen av $\boldsymbol{f}$ i punkten $\boldsymbol{a}$ blir

$$\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{a}) + \boldsymbol{f}'(\boldsymbol{a})(\boldsymbol{x} - \boldsymbol{a}).$$

Newtons metod

Låt $f : \mathbb{R} \to \mathbb{R}$ vara en deriverbar funktion. Vi skall lösa ekvationen $f(x) = 0$ med Newtons metod. Det här gjorde vi redan i läsperiod 1 så detta blir lite repetition.

Antag att vi har en approximativ lösning x_k och vi vill hitta en bättre approximation x_{k+1}. Vi bildar linjäriseringen av f i x_k:

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

och löser $L(x) = 0$ istället för $f(x) = 0$.

$$f(x_k) + f'(x_k)(x - x_k) = 0$$

Lösningen får bli nästa approximation:

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

Som exempel tar vi: Lös ekvationen $f(x) = 0$ där $f(x) = \cos(x) - x$. En graf visar att vi har ett nollställe och vi tar $x_0 = 1$ som startapproximation.

```
>> f=@(x)cos(x)-x; Df=@(x)-sin(x)-1;
>> x=1;
>> 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

   0.750363867840244  -0.249636132159756
   0.739112890911362  -0.011250976928882
   0.739085133385284  -0.000027757526078
   0.739085133215161  -0.000000000170123
```

Nu är det dags för system av n ekvationer i n obekanta

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

Om vi låter

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \quad \boldsymbol{0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix},$$

så har vi $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, och vi kan skriva systemet på formen

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

Antag att vi har en approximativ lösning $\boldsymbol{x}_k$ och vi vill hitta en bättre approximation $\boldsymbol{x}_{k+1}$. Vi bildar linjäriseringen av $\boldsymbol{f}$ i $\boldsymbol{x}_k$:

$$\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{f}'(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

och löser $\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{0}$ istället för $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

Om vi låter $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{x}_k$ så kan $\boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{0}$ skrivas som ekvationen

$$\boldsymbol{f}'(\boldsymbol{x}_k)\boldsymbol{h} = -\boldsymbol{f}(\boldsymbol{x}_k).$$

Detta är ett linjärt ekvationssystem, Jacobimatrisen $\boldsymbol{f}'(\boldsymbol{x}_k)$ är en $n \times n$-matris, som vi löser med avseende på $\boldsymbol{h}$. Nu bildar vi nästa approximation av lösningen till $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$:

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

Nu ser vi åter på exemplet vi började med. Vi har alltså

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1(1+x_2^2) - 1 \\ x_2(1+x_1^2) - 2 \end{bmatrix}, \quad \boldsymbol{f}'(\boldsymbol{x}) = \begin{bmatrix} 1+x_2^2 & 2x_1x_2 \\ 2x_1x_2 & 1+x_1^2 \end{bmatrix},$$

och skall lösa $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Från bilden med noll-nivåkurvorna ser vi att $\boldsymbol{x}_0 = (0.25, 2)$ nog är en bra startapproximation.

```
>> f=@(x)[x(1)*(1+x(2)^2)-1;x(2)*(1+x(1)^2)-2];
>> Df=@(x)[1+x(2)^2 2*x(1)*x(2);2*x(1)*x(2) 1+x(1)^2];
>> x=[0.25;2];
>> kmax=10; tol=0.5e-8;
>> for k=1:kmax
       h=-Df(x)\f(x);
       x=x+h;
       disp([x' norm(h)])
       if norm(h)<tol, break, end
   end

   0.217391304347826   1.913043478260870   0.092869605927364
   0.214829670172721   1.911781803315968   0.002855484777347
   0.214829232694196   1.911768811990568   0.000012998689285
   0.214829232680284   1.911768811998807   0.000000000016168
```

Uppgift 1. Låt $\boldsymbol{f}(\boldsymbol{x}) = (x_1^3 + x_2^2 - 1, \exp(x_1x_2) + x_1 + x_2 - 2)$. Lös ekvationssystemet $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Rita upp noll-nivåkurvorna till f_1 och f_2 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 systemet.

Färdiga program i MATLAB

I MATLAB OPTIMIZATION TOOLBOX finner vi `fsolve` som löser system av ekvationer. För en sista gång ser vi på vårt exempel. Vi har alltså

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1(1+x_2^2)-1 \\ x_2(1+x_1^2)-2 \end{bmatrix} = \boldsymbol{0},$$

och i MATLAB löser vi med (samma startapproximation som tidigare)

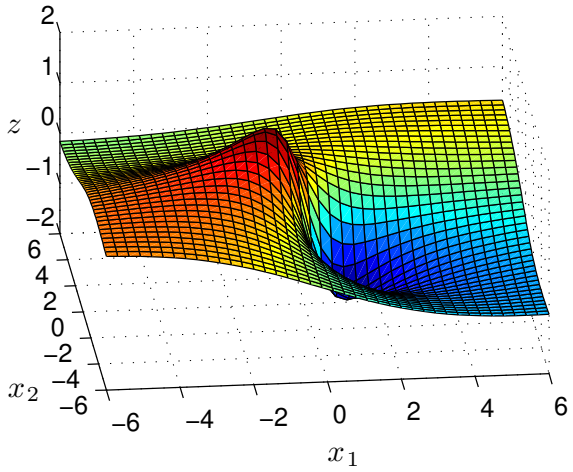
```
>> f=@(x)[x(1)*(1+x(2)^2)-1;x(2)*(1+x(1)^2)-2];
>> x0=[0.25;2];
>> x=fsolve(f,x0)
x =
   0.214829232694215
   1.911768811990538
```

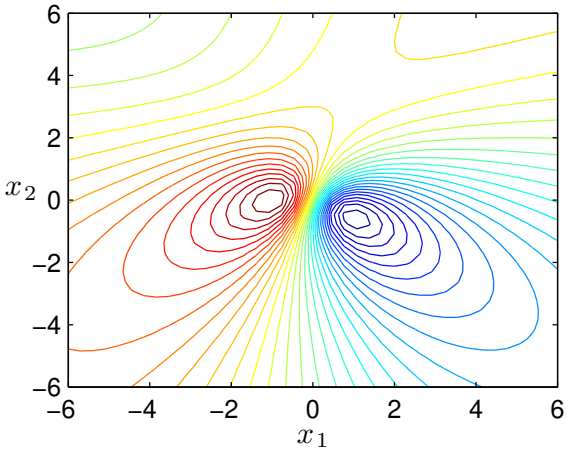
3 Optimering

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.

I förra avsnittet 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 den s.k. 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

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

I läsperiod 4 skall vi lära oss om s.k. egenvärdesproblem för matriser. Genom att beräkna egenvärdena till Hessianmatrisen $\boldsymbol{H}(\boldsymbol{a})$ kan vi avgöra vilken typ av stationär punkt vi har. Är egenvärdena positiva har vi en minimipunkt, är de negativa har vi en maximipunkt och har de olika tecken har vi en sadelpunkt.

För tillfället nöjer vi oss med den teknik som bygger på kvadratkomplettering som beskrivs i Persson-Böiers sid. 101 och framåt.

Uppgift 2. 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.

Newtons metod på $\boldsymbol{\nabla} f = \boldsymbol{0}$

Villkoret för en stationär punkt $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$ är ett 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:

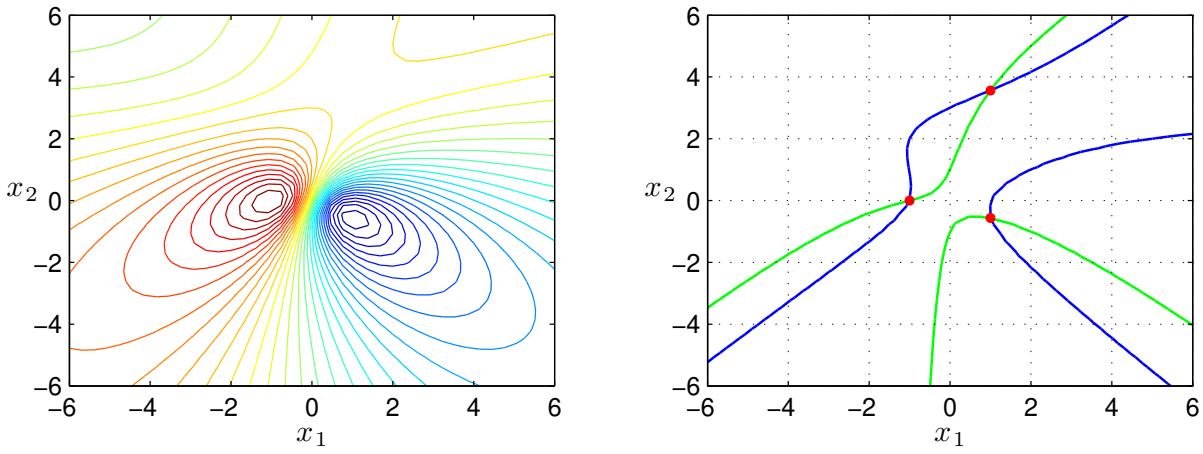
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h},$$

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

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

Här är Jacobimatrisen $\boldsymbol{g}'(\boldsymbol{x}_k)$ 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.

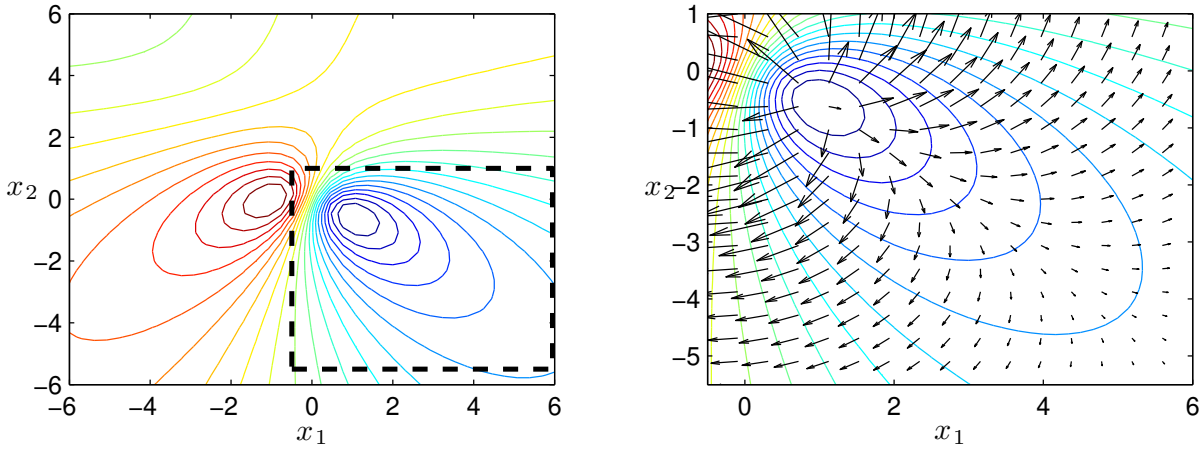


Nu kan vi läsa av startapproximationer av de stationära punkterna och bestämma dem noggrannt med Newtons metod.

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, 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

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

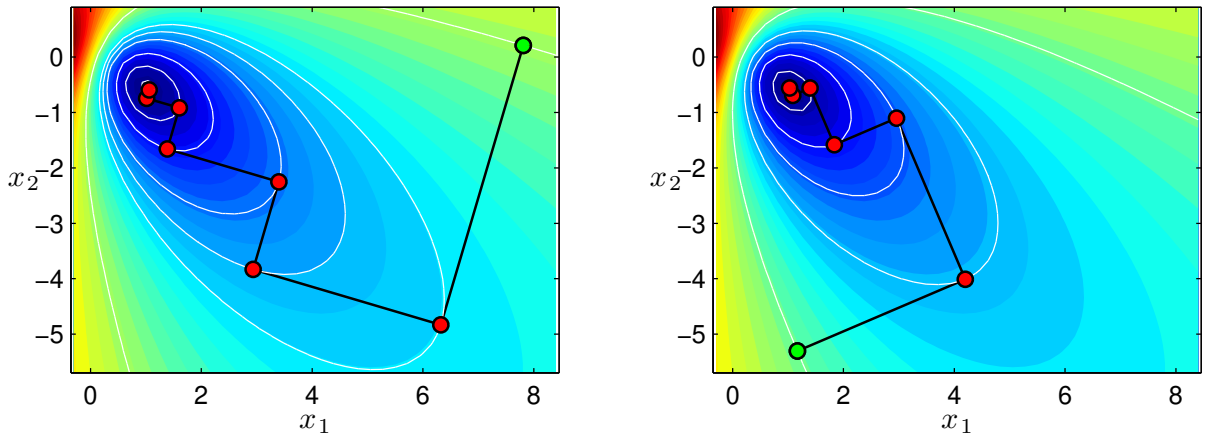
vilket garanterar en minskning av funktionsvärdet.

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 (varför?) och tar ett steg längs $-\boldsymbol{\nabla} f(\boldsymbol{x}_k)$ tills vi tangerar en nivåkurva (varför?), 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 $f(x_1, x_2) = x_1^2 x_2 \exp(-(x_1^2 + x_2^2))$. 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.

Färdiga program i MATLAB

I OPTIMIZATION TOOLBOX i MATLAB finns `fminunc` för minimering av funktioner i flera variabler. För funktioner vi vill minimera men som inte är deriverbara 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)*exp(-(x(1)^2+x(2)^2)),[1;-1])
```

vilket minimerar funktionen i uppgift 3 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 funktion i MATLAB, till exempel

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

4 Dubbelintegral

Vi skall börja med att approximera dubbelintegralen av en funktion över ett rektangulärt område

$$\iint_R f(x,y)\,dA$$

där $R = \{(x,y)\colon\ a \le x \le b,\ c \le y \le d\}$. Sedan skall vi se på allmännare områden.

Enkelintegraler approximerade vi redan i läsperiod 1 och vi skall för dubbelintegraler använda samma typ av approximationer. Då beskrev vi vänster och höger rektangelregel, mittpunktsregeln och trapetsregeln. Samtliga dessa metoder för enkelintegralen kan generaliseras till multipelintegraler, men vi kommer nöja oss med att titta på dubbelintegraler.

Rektangelregeln

I läsperiod 1 betraktade vi enkelintegralen över ett intervall

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

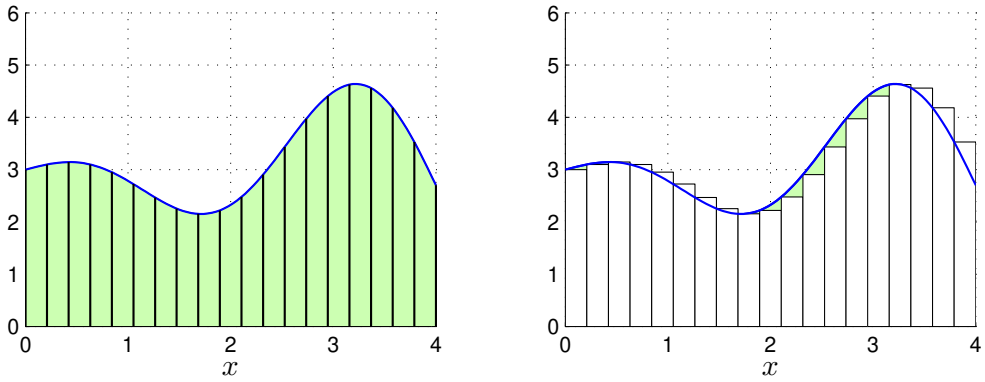
Vi gjorde en likformig indelning av intervallet $a \le x \le b$ enligt

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

så att vi fick n lika långa delintervall $x_{i-1} \le x \le x_i$ med samma bredd $\Delta x = h = \frac{b-a}{n}$.

Vi approximerade $f(x)$ med $f(x_{i-1})$ i intervallen $x_{i-1} \le x \le x_i$ och fick *vänster rektangelregel*

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



Nu skall vi upprepa samma resonemang för dubbelintegralen

$$\iint_R f(x,y)\,dA$$

där $R = \{(x,y)\colon\ a \le x \le b,\ c \le y \le d\}$.

Förutom indelning av intervallet $a \le x \le b$, gör vi nu även en likformig indelning av intervallet $c \le y \le d$ enligt

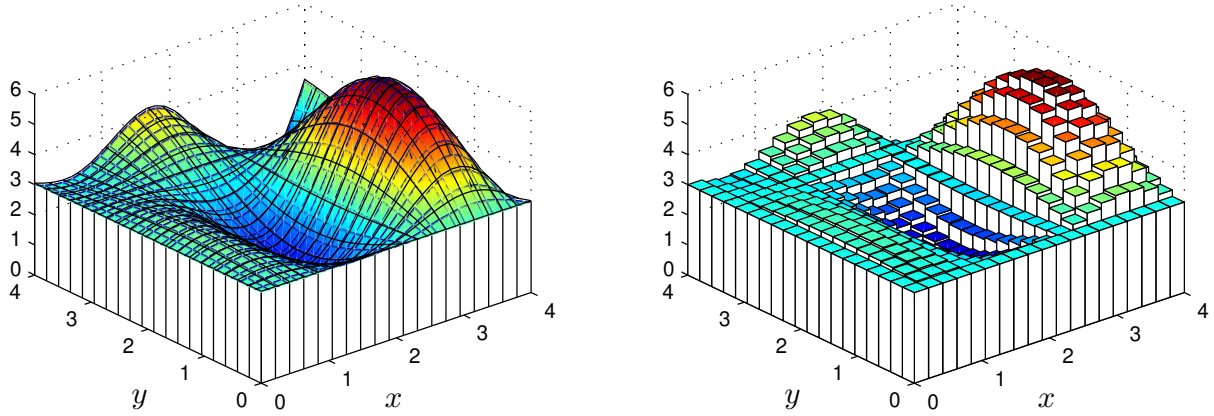
$$c = y_0 < y_1 < y_2 < \cdots < y_{m-1} < y_m = d$$

så att vi får m lika långa delintervall $y_{j-1} \le y \le y_j$ med samma bredd $k = \frac{d-c}{m}$.

Vi får därmed en indelning av området $a \le x \le b,\ c \le y \le d$ i nm stycken likformiga små rektanglar som har arean $\Delta A = hk$ var och en.

Om vi approximerar $f(x,y)$ med $f(x_{i-1},y_{j-1})$ på området $x_{i-1} \le x \le x_i,\ y_{j-1} \le y \le y_j$, får vi **vänster rektangelregel** för dubbelintegralen

$$\iint_R f(x,y)\,dA = \sum_{i=1}^{n}\sum_{j=1}^{m} \iint_{R_{i,j}} f(x,y)\,dA$$
$$\approx \sum_{i=1}^{n}\sum_{j=1}^{m} f(x_{i-1},y_{j-1})\,\Delta A$$



Vi approximerar varje liten delintegral med volymen av ett rätblock vars bas har arean hk och som har höjden $f(x_{i-1},y_{j-1})$ och sedan summerar vi alla bidragen.

Om vi istället approximerar $f(x,y)$ med $f(x_i,y_j)$ får vi **höger rektangelregel** för dubbelintegralen

$$\iint_R f(x,y)\,dA = \sum_{i=1}^{n}\sum_{j=1}^{m} \iint_{R_{i,j}} f(x,y)\,dA$$
$$\approx \sum_{i=1}^{n}\sum_{j=1}^{m} f(x_i,y_j)\,\Delta A$$

Slutligen så får vi ett betydligt noggrannare resultat med **mittpunktsregeln**, där vi beräknar höjden mittpunkten i varje delrektangel, och **trapetsregeln** som vi får genom att ta medelvärdet av höjderna (funktionsvärdena) i alla fyra hörnpunkterna i varje delrektangel.

Uppgift 4. Beräkna i MATLAB en approximation av enkelintegralen

$$\int_0^1 x\,\sin(x)\,dx$$

med vänster och höger rektangelregel samt trapetsregeln. Använd funktionen `sum` i MATLAB för att få en enklare kod än om man använder `for`-satser.

Uppgift 5. Beräkna nu i MATLAB en approximation av dubbelintegralen

$$\iint_R y\,\sin(y+xy)\,dA$$

där $R = \{(x,y)\colon\ 0 \le x \le 1,\ -\pi/2 \le y \le \pi/2\}$.

Använd vänster och höger rektangelregel samt trapetsregeln. Jämför deras noggrannhet genom att räkna ut det exakta värdet av dubbelintegralen för hand. (Alla elementen i en matris summeras genom att man tar `sum` av `sum`.)

Färdiga program i MATLAB

I MATLAB finns t.ex. `integral` för beräkning av enkelintegraler samt `integral2` och `integral3` för beräkning av dubbelintegraler respektive trippelintegraler.

Vi skall som exempel beräkna dubbelintegralen

$$\iint_R y\,\sin(x) + x\,\cos(y)\,dA$$

där $R = \{(x, y)\colon\ \pi \le x \le 2\pi,\ 0 \le y \le \pi\}$.

Beskriv alltid integranden som om du skulle rita dess funktionsyta, dvs. tänk på `x` och `y` som matriser och använd komponentvisa operationer. Beräkningen utförs enligt

```
>> f=@(x,y)y.*sin(x)+x.*cos(y);
>> q=integral2(f,pi,2*pi,0,pi)
```

Som ytterligare ett exempel tar vi

$$\iint_D x^2\cos(y^3) - 3\sin(y)\,dA$$

där $D = \{(x, y)\colon\ |x| + |y| \le 1\,\}$.

Området kan beskrivas $-1 \le x \le 1$, $c(x) \le y \le d(x)$, där $c(x) = |x| - 1$ och $d(x) = 1 - |x|$.

Så här utför vi beräkningen i MATLAB:

```
>> f=@(x,y)x.^2.*cos(y.^3)-3*sin(y);
>> a=-1; b=1; c=@(x)abs(x)-1; d=@(x)1-abs(x);
>> q=integral2(f,a,b,c,d)
```

Integrationsgränserna i y-led kan ges av två funktioner, medan integrationsgränserna i x-led måste ges av två tal.