1 Inledning

Vi skall lösa system av icke-linjära ekvationer $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, där $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, genom att generalisera Newtons metod. Sedan skall vi 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 $f : \mathbb{R}^n \to \mathbb{R}$ och mängden Ω är hela eller en del av definitionsmängden D_f. I det första fallet säger vi att vi har ett problem *utan bivillkor* och i det andra ett problem *med bivillkor*. Avslutningsvis skall vi beräkna dubbel- och trippelintegraler, genom att generalisera de metoder för enkelintegraler som vi redan tittat på.

2 Funktionsytor och nivåkurvor

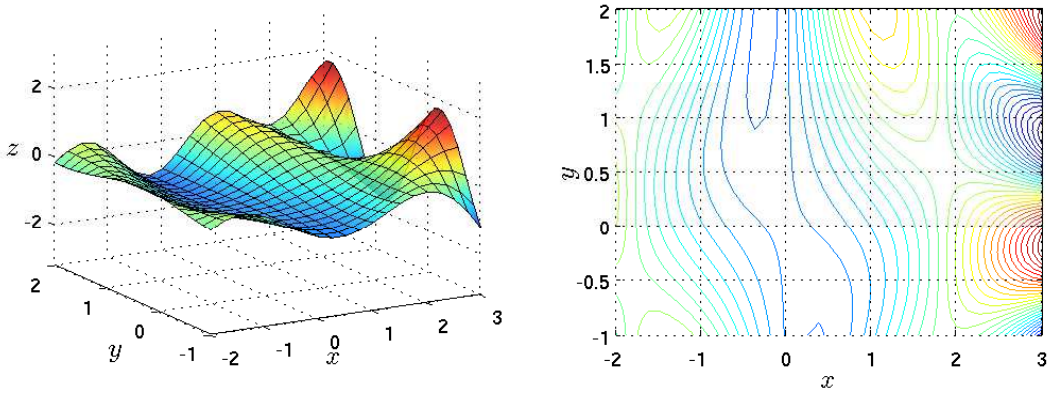
I envariabelanalys kan man med grafer illustrera det mesta, när det gäller flervariabelanalys är det betydligt svårare (dimensionerna tar slut). Vi skall nöja oss med att rita funktionsytor och nivåkurvor till funktioner i två variabler.

En graf till en funktion i en variabel $f : \mathbb{R} \to \mathbb{R}$ är mängden $\{(x, y) : y = f(x)\}$, dvs. en kurva i planet. En graf till en funktion i två variabler $f : \mathbb{R}^2 \to \mathbb{R}$ är mängden $\{(x, y, z) : z = f(x, y)\}$, dvs. en yta i rummet.

Som exempel tar vi $f(x, y) = (\frac{1}{3}x^2 - 1)\sin(1 - xy)$ över området $-2 \le x \le 3$, $-1 \le y \le 2$.

Resultatet får vi med kommandot `surf`, vilket är motsvarigheten till `plot` då vi skall rita ytor.

```
>> f=@(x,y)(0.3*x.^2-1).*sin(1-x.*y);
>> x=linspace(-2,3,40); y=linspace(-1,2,40);
>> [X,Y]=meshgrid(x,y); Z=f(X,Y);
>> surf(X,Y,Z)
>> xlabel('x'), ylabel('y'), zlabel('z'), grid on
```



Ser vi ytan som ett landskap så anger färgen på ytan höjden över havet.

Ett annat sätt att åskådliggöra en funktion i två variabler $f\colon \mathbb{R}^2 \to \mathbb{R}$ är att rita nivåkurvor, dvs. mängderna $\{(x,y)\colon f(x,y) = c\}$, där c är en konstant som anger nivån.

Resultatet som vi ser ovan till höger får vi med kommandot `contour` så här

```
>> contour(X,Y,Z,30)
>> xlabel('x'), ylabel('y')
```

Vi får en slags topografisk karta av funktionen om vi ser funktionsytan som ett landskap och då blir nivån helt enkelt höjden över havet. Vi får bilden ovan till höger med

Uppgift 1. Rita funktionsyta och nivåkurvor till funktionen $f : \mathbb{R}^2 \to \mathbb{R}$ där

$$f(x,y) = -xy\exp(-2(x^2+y^2))$$

över området $-2 \le x \le 2,\ -2 \le y \le 2$.

3 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

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

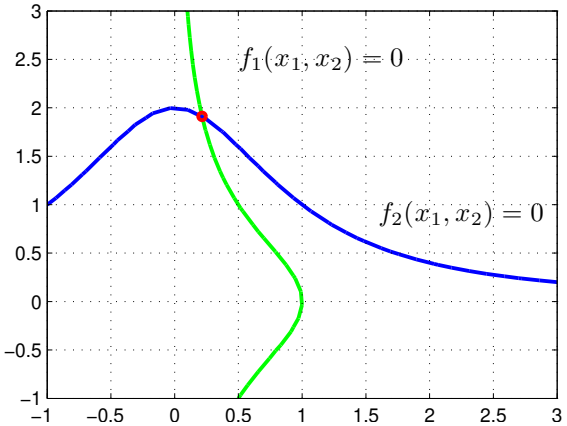
kan ekvationssystemet skrivas

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

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

Vi måste börja med att i MATLAB beskriva f_1 och f_2. För att vi skall kunna beräkna dessa med matriser av x_1- och x_2-värden låter vi dem bero av två separata variabler.

```
>> f1=@(x1,x2)x1.*(1+x2.^2)-1;
>> f2=@(x1,x2)x2.*(1+x1.^2)-2;
```

Nu kan vi använda `contour` på ett enkelt sätt. Eftersom `contour` kan ges antal nivåer eller en vektor med nivåvärden måste vi ge vektorn `[0 0]`, ger vi bara `0` tolkas det som ingen nivå.

```
>> x1=linspace(-1,3,30); x2=linspace(-1,3,30);
>> [X1,X2]=meshgrid(x1,x2);
>> Z1=f1(X1,X2); Z2=f2(X1,X2);
>> contour(x1,x2,Z1,[0 0],'g')
>> hold on
>> contour(x1,x2,Z2,[0 0],'b')
>> grid on
```

Som resultat får vi figuren på förra sidan.

När vi skall använda Newtons metod vill vi beskriva problemet med den vektorvärda funktionen $\boldsymbol{f}$ och med vektorn $\boldsymbol{x}$ som variabel. Med koden

```
>> f=@(x)[f1(x(1),x(2))
          f2(x(1),x(2))];
```

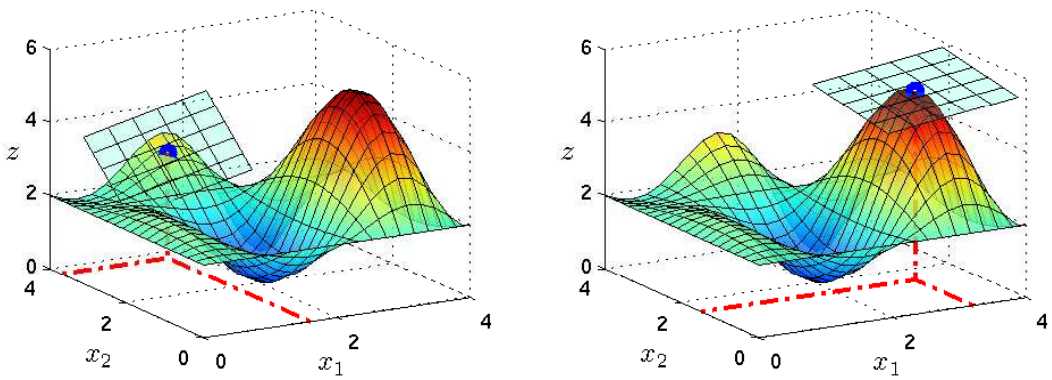
bildar vi denna funktion. Lägg märke till hur vi använder komponentfunktionerna f_1 och f_2 samt hur vi plockar ut x_1- och x_2-värdena ur vektorn $\boldsymbol{x}$. Med `[...]` ser vi till att $\boldsymbol{f}$ blir en vektor.

3.1 Linjärisering

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

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

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

$$L_1(\boldsymbol{x}) = f_1(\boldsymbol{a}) + \boldsymbol{\nabla} f_1(\boldsymbol{a})^T (\boldsymbol{x} - \boldsymbol{a})$$
$$L_2(\boldsymbol{x}) = f_2(\boldsymbol{a}) + \boldsymbol{\nabla} f_2(\boldsymbol{a})^T (\boldsymbol{x} - \boldsymbol{a})$$

eller med matrisbeteckningar

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

där $\boldsymbol{L}(\boldsymbol{x}) = (L_1(\boldsymbol{x}), L_2(\boldsymbol{x}))$ och $\boldsymbol{D}\boldsymbol{f}(\boldsymbol{x})$ är Jacobimatrisen

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

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

$$\frac{\partial f_1}{\partial x_1}(\boldsymbol{x}) = 1 + x_2^2, \quad \frac{\partial f_1}{\partial x_2}(\boldsymbol{x}) = 2x_1 x_2$$

$$\frac{\partial f_2}{\partial x_1}(\boldsymbol{x}) = 2x_1 x_2, \quad \frac{\partial f_2}{\partial x_2}(\boldsymbol{x}) = 1 + x_1^2$$

och därmed

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

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

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

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

$$\boldsymbol{Df}(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(\boldsymbol{x}) & \ldots & \dfrac{\partial f_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial x_1}(\boldsymbol{x}) & \ldots & \dfrac{\partial f_m}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}$$

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

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

3.2 Newtons metod

Nu skall vi lösa system av n ekvationer i n obekanta

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

Om vi låter $\boldsymbol{x} = (x_1,\cdots,x_n)$, $\boldsymbol{f} = (f_1,\cdots,f_n)$ och $\boldsymbol{0} = (0,\cdots,0)$ så har vi $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, och vi kan skriva systemet på formen

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

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

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

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

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

$$\boldsymbol{Df}(\boldsymbol{x}_k)\boldsymbol{d} = -\boldsymbol{f}(\boldsymbol{x}_k)$$

Detta är ett linjärt ekvationssystem, Jacobimatrisen $\boldsymbol{Df}(\boldsymbol{x}_k)$ är en $n \times n$-matris, som vi löser med avseende på $\boldsymbol{d}$.

Nu bildar vi nästa approximation av lösningen till $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$:

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

Som stoppvillkor för iterationen tar vi $\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \| \leq tol$. Det betyder att vi accepterar $\boldsymbol{x}_{k+1}$ om ändringen i sista iteration är mindre än toleransen tol. Vi tillåter maximalt k_{max} iterationer ($k_{max} = 10$ är rimligt).

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

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

och skall lösa $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Funktionen $\boldsymbol{f}$ har vi redan beskrivit i MATLAB, återstår att beskriva Jacobimatrisen

```
>> Df=@(x)[1+x(2)^2    2*x(1)*x(2)
           2*x(1)*x(2) 1+x(1)^2   ];
```

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

```
>> x=[0.25;2];
>> kmax=10; tol=0.5e-8;
>> for k=1:kmax
       d=-Df(x)\f(x);
       x=x+d;
       disp([x' norm(d)])
       if norm(d)<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
```

Vi ser att approximationerna konvergerar snabbt.

Uppgift 2. 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.

3.3 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)

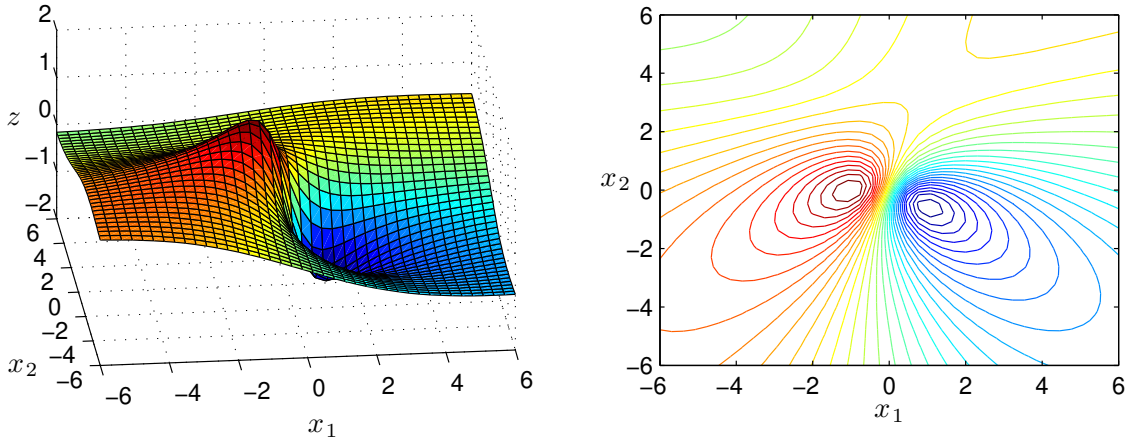
```
>> f=@(x)[x(1)*(1+x(2)^2)-1
          x(2)*(1+x(1)^2)-2];
>> x0=[0.25;2];
>> x=fsolve(f,x0)
x =
   0.214829232694215
   1.911768811990538
```

4 Optimering utan bivillkor

Vi skall bestämma max- och minpunkter 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 minpunkt, en lokal maxpunkt och en sadelpunkt.

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

I tidigare avsnitt 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 Hessematrisen av andraderivator

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2}(\boldsymbol{x}) & \dfrac{\partial^2 f}{\partial x_1 x_2}(\boldsymbol{x}) \\ \dfrac{\partial^2 f}{\partial x_2 x_1}(\boldsymbol{x}) & \dfrac{\partial^2 f}{\partial x_2^2}(\boldsymbol{x}) \end{bmatrix}$$

Eftersom $\boldsymbol{a}$ 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. Är egenvärdena positiva har vi en minpunkt, är de negativa har vi en maxpunkt och har de olika tecken har vi en sadelpunkt.

Eftersom Hessematrisen är en symmetrisk matris (vi har ju $\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_1}$) kan $\boldsymbol{H}(\boldsymbol{a})$ diagonaliseras av en ortogonalmatris $\boldsymbol{V}$

$$\boldsymbol{V}^T \boldsymbol{H}(\boldsymbol{a}) \boldsymbol{V} = \boldsymbol{D}$$

där $\boldsymbol{D}$ är en diagonalmatris med egenvärdena och kolonnerna i $\boldsymbol{V}$ är motsvarande egenvektorer.

Inför vi variabelbytet $\boldsymbol{x} = \boldsymbol{a} + \boldsymbol{V}\boldsymbol{y}$ så övergår den kvadratiska termen i Taylorutvecklingen till diagonalform

$$(\boldsymbol{x}-\boldsymbol{a})^T \boldsymbol{H}(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}) = (\boldsymbol{V}\boldsymbol{y})^T \boldsymbol{H}(\boldsymbol{a}) \boldsymbol{V}\boldsymbol{y} = \boldsymbol{y}^T \boldsymbol{V}^T \boldsymbol{H}(\boldsymbol{a}) \boldsymbol{V}\boldsymbol{y} =$$

$$= \boldsymbol{y}^T \boldsymbol{D}\boldsymbol{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2$$

Vi ser att tecknen på egenvärdena avgör ytans utseende, dvs. avgör typen av stationär punkt.

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

4.1 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:

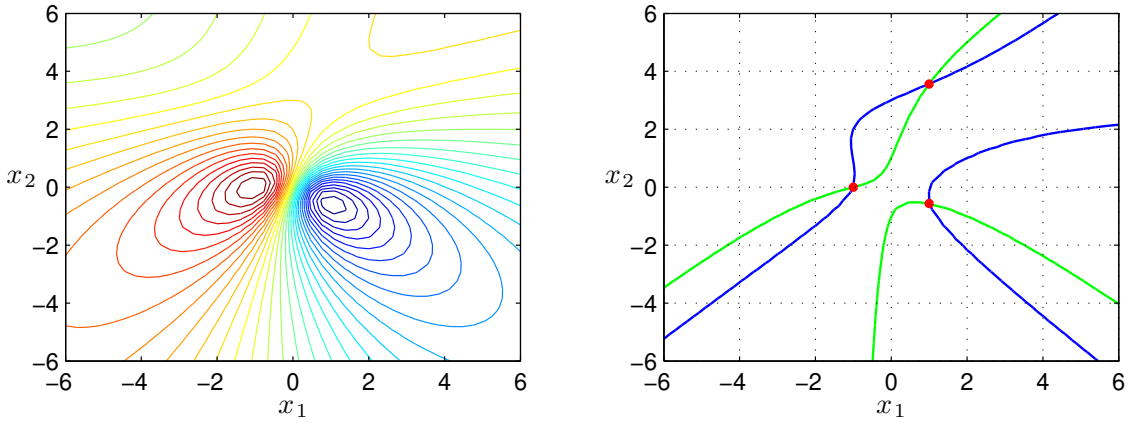
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h}$$

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 $\boldsymbol{D}\boldsymbol{g}(\boldsymbol{x}_k)$ är en $n \times n$-matris. (Jacobimatrisen till $\boldsymbol{g}$ är faktiskt Hessematrisen 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 approximationer av de stationära punkterna och bestämma dem noggrant med Newtons metod.

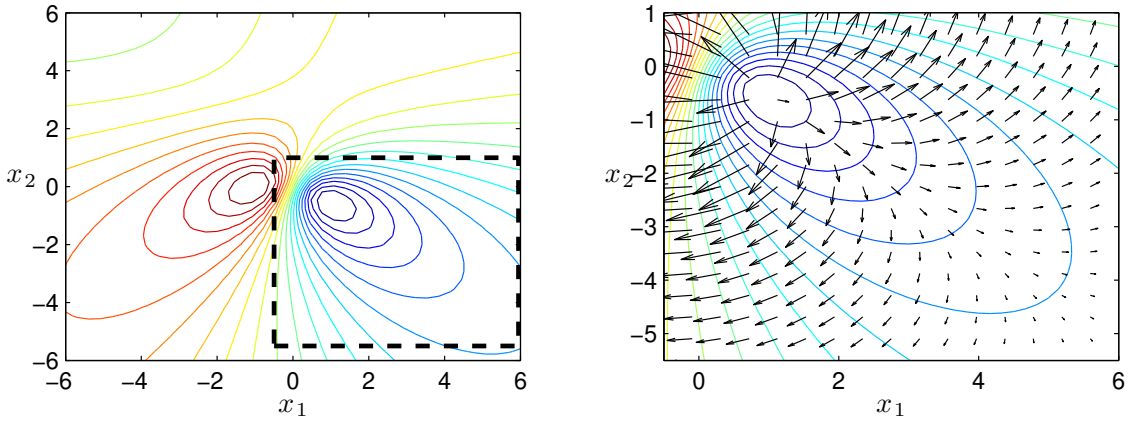


Uppgift 4. Åter till funktionen $f(x_1, x_2) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$ från uppgift 3. Beräkna alla stationära punkter noggrant med Newtons metod, dvs. lös ekvationen $\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$. Avgör sedan vilken typ av stationära punkter vi med hjälp av egenvärdena till Hessematrisen $\boldsymbol{H}(\boldsymbol{x})$.

4.2 Steepest descentmetoden

En funktion växer som ni vet snabbast i gradientens riktning och minskar snabbast i negativa gradientens riktning. Vi skall söka efter minpunkter 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 minpunkten 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 minpunkt, bilda 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 $-\boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{x}_k)$.

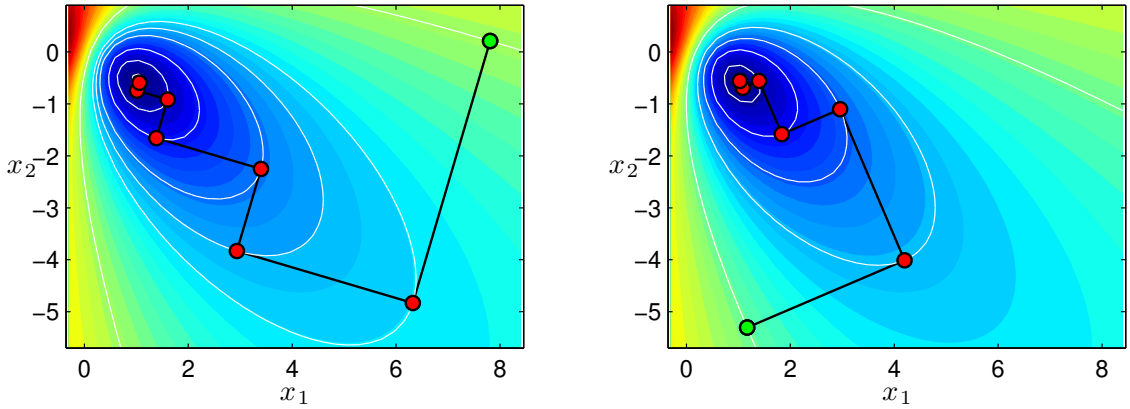
Vi väljer s_k så att $f(\boldsymbol{x}_k - s_k \boldsymbol{\nabla} f(\boldsymbol{x}_k)) < f(\boldsymbol{x}_k)$, vilket garanterar att funktionsvärdet minskas.

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 minpunkten. 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 som går 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 maxpunkt går man antingen i positiv gradientriktning (steepest ascent) eller så använder man steepest descent på $-f(\boldsymbol{x})$.

Uppgift 5. 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 max- och minpunkter samt sadelpunkter blir synliga. Använd sedan steepest descentmetoden för att beräkna lokala max- och minpunkter. Välj steglängd genom att minimera $g(s) = f(\boldsymbol{x}_k - s\boldsymbol{\nabla} f(\boldsymbol{x}_k))$ med `fminbnd`. Läs först hjälptexten för `fminbnd`.

4.3 Färdiga program 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 5 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 6. Betrakta funktionen $f(x, y) = x^3 + 3xy^2 - 15x + y^2 - 15y$. Bestäm alla stationära punkter till f och avgör deras typ. Använd `fsolve` och `eig`. Bestäm också eventuella min- eller maxpunkten med `fminunc` på lämpligt sätt.

5 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.

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

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

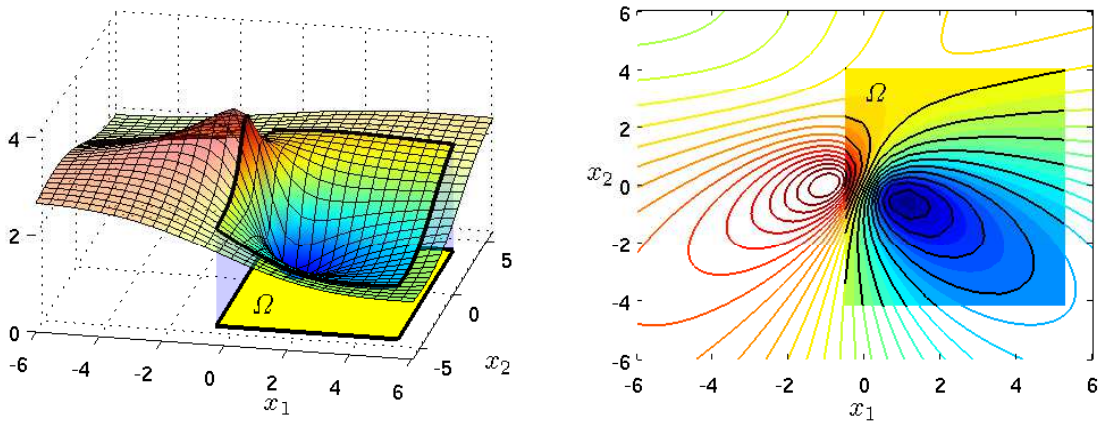
Här kallas f ofta för *objektfunktion* och Ω för *tillåtna mängden*. Bivillkoret $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}$ kallas för *olikhetsbivillkor* och $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$ kallas för *likhetsbivillkor*.

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 minpunkt i det inre av Ω och en maxpunkt på randen av Ω.

5.1 Färdiga program i MATLAB

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

Objektfunktionen kan beskrivas som en anonym funktion eller med en funktionsfil.

Icke-linjära bivillkor måste beskrivas med en funktionsfil. Bivillkor som är linjära beskrivs med matriser och vektorer, $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}$ för olikhetsbivillkor och $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{d}$ för likhetsbivillkor. Med $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ ges enkla gränser på variablerna. Skulle en variabel t.ex. x_i inte vara begränsad nedåt (uppåt) så låter vi $l_i = -\infty$ ($u_i = +\infty$).

Vi måste också ge en startapproximation $\boldsymbol{x}_0$ och `fmincon` används enligt

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

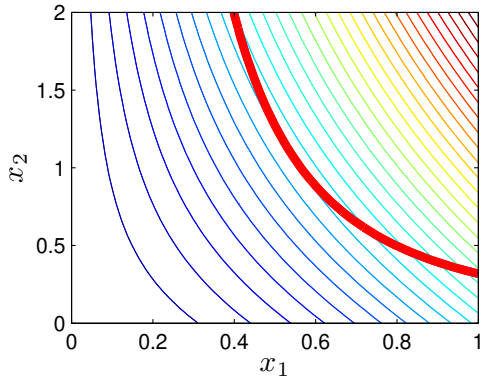
Som exempel tar vi: Minimera plåtåtgången vid tillverkningen av en burk med radien x_1 och höjden x_2, givet att den skall rymma volymen $V = 1$.

Arean av burkens yta är $A = 2\pi x_1 x_2 + 2\pi x_1^2$ och dess volym är $V = \pi x_1^2 x_2$ så vi skall lösa

$$\min_{h(\boldsymbol{x})=0} f(\boldsymbol{x}), \quad \text{där} \quad f(\boldsymbol{x}) = 2\pi x_1 x_2 + 2\pi x_1^2 \quad \text{och} \quad h(\boldsymbol{x}) = \pi x_1^2 x_2 - 1$$

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)
```



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
```

Bivillkor som inte finns med i vårt aktuella problem ersattes med tomma mängder.

6 Dubbelintegral

Enkelintegralen $\int_a^b f(x)\,dx$ över ett intervall betraktade vi i en tidigare laboration. 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 $\Delta x = \frac{b-a}{n}$.

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

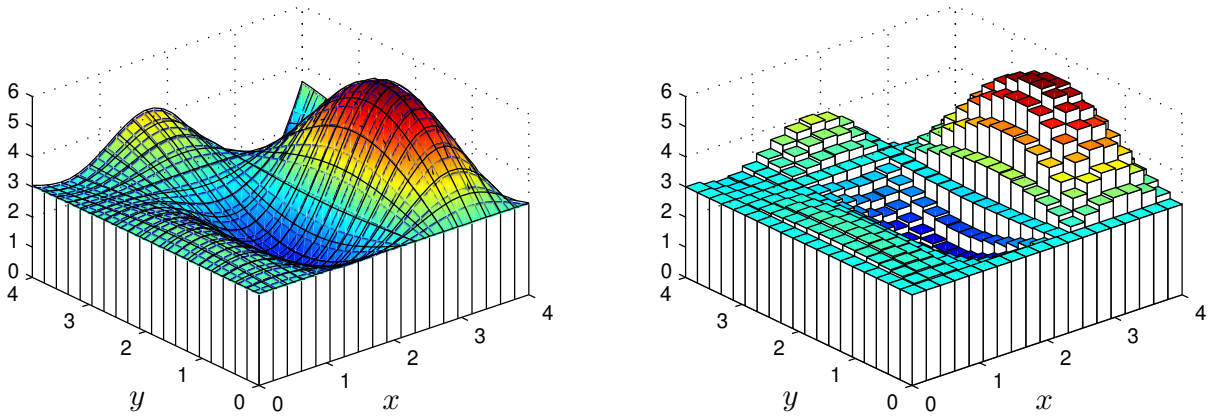
$$c = y_0 < y_1 < y_2 < \cdots < y_{m-1} < y_m = d$$

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

Vi får därmed en indelning av området R i nm stycken likformiga små rektanglar $R_{i,j} = \{(x,y)\colon\ x_{i-1} \le x \le x_i,\ y_{j-1} \le y \le y_j\}$ som har arean $\Delta A = \Delta x \Delta y$ var och en.

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

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



Vi approximerar varje liten delintegral med volymen av ett rätblock vars bas har arean $\Delta x \Delta y$ 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 7. 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`.)

6.1 Färdiga program i MATLAB

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

Vi skall som exempel beräkna dubbelintegralen

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

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

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

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

Som ytterligare ett exempel tar vi

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

där $D=\{(x,y)\,:\,|x|+|y|\le 1\,\}$.

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.

Uppgift 8. Vi skall beräkna integralen

$$\iint_R x \exp(xy)\, dA$$

där $R = \{(x, y)\colon\ 0 \leq x \leq 2,\ 0 \leq y \leq 1\}$.

Rita först funktionsytan till integranden över aktuellt område. Använd sedan `integral2` för att beräkna integralen. Jämför gärna med exakt värde som du i så fall räknar ut för hand och jämför gärna med vad rektangel- och trapetsreglerna ger.

Uppgift 9. Beräkna integralen

$$\iint_D (a - x + y)\, dA$$

där $D = \{\, (x, y)\colon 0 \leq y \leq a - \frac{x^2}{a} \,\}$, med `integral2`. Jämför med exakt svar $\frac{28}{15}a^3$.