Flervariabelanalys och MATLAB
Kapitel 1

Thomas Wernstål
Matematiska Vetenskaper

29 augusti 2012

1 Kurvor och ytor

1.1 Funktionsytor

I detta kompendium kommer vi på olika sätt studera funktioner från $\mathbb{R}^n$ till $\mathbb{R}^m$, där n och m är $1, 2$ eller 3, och i kapitel 1 och 4 skall vi speciellt se hur sådana funktioner kan visualiseras geometriskt. Vi börjar i detta avsnitt med att studera funktioner från $\mathbb{R}^2$ till $\mathbb{R}$ dvs reellvärda funktioner av två variabler. Innan vi går in på olika plottkommandon så kan det vara praktiskt att titta på hur man skapar en anonym funktion i MATLAB, där funktionen beror av två variabler. Funktionen $f(x, y) = x \sin y^2$ skapar vi med kommandot

```
>> f=@(x,y) x.*sin(y.^2);
```

När man definierar funktioner så är det viktigt att tänka på att MATLAB arbetar med matriser och matrisoperationer. Vill man att funktionen skall klara elementvisa kalkyler (som i de flesta fall i denna kurs) måste man använda "punkterade operationer" (dvs. `.^` `.*` och `./` istället för bara `^` `*` och `/`). Om en viss funktion saknar punkter för att klara elementvisa kalkyler så kan man också be MATLAB att sätta ut dem på de rätta ställena med kommandot `vectorize`. Testa t.ex.

```
>> f=@(x,y) x*sin(y^2);
>> vectorize(f)
```

Att för hand rita funktionsytor är inte helt lätt. Låt oss därför titta på hur man kan använda MATLAB för att få en bild av grafen till en funktion av två variabler. Antag att vi vill plotta grafen till en funktion $f(x, y)$ över en rektangel $-1 \leq x \leq 2, 0 \leq y \leq 3$. Vi börjar då med att skapa koordinatmatriser med hjälp av kommandot `meshgrid`. Detta kommando ger oss en massa punkter (x_i, y_j) i xy-planet i vilket vi sedan kan beräkna funktionsvärdena $f(x_i, y_j)$. Kommandot

```
>> [X Y]=meshgrid(-1:0.5:2,0:0.4:3)
```

skapar t.ex. två matriser `X` och `Y` med x- resp. y-koordinater som är sådana att alla rader i `X` består av talen $-1, -0.5, 0, ..., 2$ (dvs. de som genereras av `-1:0.5:2`) och alla kolonner i `Y` består av talen $0, 0.4, 0.8, ..., 2.8$ (dvs. de som genereras av `0:0.4:3`).

$$X = \begin{bmatrix} -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\ -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 1.2 & 1.2 & 1.2 & 1.2 & 1.2 & 1.2 & 1.2 \\ 1.6 & 1.6 & 1.6 & 1.6 & 1.6 & 1.6 & 1.6 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2.4 & 2.4 & 2.4 & 2.4 & 2.4 & 2.4 & 2.4 \\ 2.8 & 2.8 & 2.8 & 2.8 & 2.8 & 2.8 & 2.8 \end{bmatrix}$$

Genom att ta ett tal ur matrisen `X` och motsvarande tal ur matrisen `Y` så får vi koordinaterna för en punkt i xy-planet. T.ex. får vi punkten (x_5, y_3) (dvs. $(1, 0.8)$ i det här fallet) med kommandot

```
>>[X(3,5) Y(3,5)]
```

Genom att låta `i` och `j` variera så bildar `[X(j,i),Y(j,i)]` $= (x_i, y_j)$ ett gitter av punkter i rektangeln $-1 \leq x \leq 2, 0 \leq y \leq 3$.

Om man har med för många punkter i gittret (dvs. om matriserna `X` och `Y` är för stora) så riskerar man att beräkningar med matriserna tar för lång tid för MATLAB att utföra och i värsta fall kan det ”hänga sig”. Var därför försiktig och börja med ett grövre gitter om du känner dig osäker på vad som kan vara lämpligt. För att t.ex. plotta ytor så har vi i detta kompendium valt att beräkna funktionsvärdena i ca. 400 punkter. Detta innebär att intervallen på x- resp. y-axeln indelas med ca 20 delningspunkter. Om vi har lika många punkter på de två axlarna så blir koordinatmatriserna kvadratiska. Då finns en risk att utelämnade punkter i uttrycket för beräkning av funktionsvärden inte upptäcks. Beräkningar av typen `x*y, x^3` är ju fullt möjliga att utföra om `x` och `y` är kvadratiska matriser. Vi väljer därför i fortsättningen att inte ha lika många punkter i x-led som i y-led. Följande kommandon ger koordinatmatriser för området $-1 \leq x \leq 2$, $0 \leq y \leq 3$ med 21 punkter på x-axeln och 20 punkter på y- axeln (matriserna `X` och `Y` består alltså av 420 element)

```
>> x=linspace(-1,2,21);y=linspace(0,3,20);
>> [X Y] = meshgrid(x,y);
```

Nu när vi skapat våra gitterpunkter så är vi redo att beräkna funktionsvärdena.

Kommandot

```
>> Z = f(X,Y);
```

ger en 20×21-matris `Z` där `Z(j,i)` $= f(x_i, y_j)$.

Vi kan nu rita funktionsytan med kommandot

```
>> mesh(X,Y,Z)
```

eller med kommandot

```
>> surf(X,Y,Z)
```

Om man går in på Tools och väljer Rotate 3D, i figurfönstrets menyer, så kan man rotera ytan och se den från olika håll (håll inne vänster musknapp och rör på musen).

Med kommandot `surfl` kan vi även styra belysningen på ytan. Till exempel kan vi plotta ytan, belyst med en ljuskälla placerad i punkten $(3, 4, 5)$

```
>> surfl(X,Y,Z,[3,4,5])
```

Se även kommandona `light` och `camlight`.

Man kan också ändra färgläggningen av ytorna. Om vi t.ex. vill jämna ut ytans färger så att de inte ändras så tvärt (från en meshruta till en annan) så kan vi använda kommandot

```
>> shading interp
```

Om inget annat anges så bestäms själva färgen på ytan av höjden över xy-planet (dvs. z-koordinatens värde). Vi kan dock själva bestämma den färg som skall kopplas till varje meshpunkt genom att i plottkommandot ange en matris (med samma storlek som koordinatmatriserna) som innehåller information om färg på respektive punkt. Pröva t.ex.

```
>> P=rand(size(X));
>> surf(X,Y,Z,P)
```

Man kan också ändra färgskalan. Pröva t.ex.

```
>> colormap copper
```

Färgsättningen på ytan kan också hämtas från ett fotografi (texture mapping). Pröva t.ex

```
>> P = imread('testpat1.png');
>> warp(X,Y,Z,P)
```

Fler kommandon som anger hur ytor presenteras får du med kommandot.

```
>> help graph3d
```

Vi skall nedan titta på några fler exempel som visar att funktionsytor inte alltid är så snälla och fina som den till funktionen ovan. Ibland kan det till och med vara svårt att se ur den plottade grafen om funktionen är kontinuerlig eller ej. **Resten av detta avsnitt kan betraktas som överbetygsnivå och kan hoppas över vid en första genomläsning**.

Exempel. Följande kommandon plottar grafen till funktionen $f(x,y) = x/\left(x^2+y^2\right)$.

```
>> f1=@(x,y) x./(x.^2+y.^2);
>> x=linspace(-1,1,21);y=linspace(-1,1,20);
>> [X Y] = meshgrid(x,y);Z=f1(X,Y);
>> surf(X,Y,Z)
```

Funktionen är ju inte definierad i punkten $(x,y) = (0,0)$ och det är oklart från figuren om det finns något gränsvärde då $(x,y) \to (0,0)$. Vad vi ser är att funktionsvärdena tycks variera mycket för (x,y) nära origo. Det är då skäl att bli lite misstänksam. I själva verket är det ju faktiskt så att $f(x,0) \to \pm\infty$ då $x \to 0$, så funktionen $f(x,y) = x/(x^2+y^2)$ saknar gränsvärde då $(x,y) \to (0,0)$. □

Låt oss titta på några exempel till.

Exempel. Om vi plottar grafen till funktionen $f(x,y) = (x^2+y^2)/(x+y)$

```
>> f2=@(x,y) (x.^2+y.^2)./(x+y);
>> x=linspace(-1,1,21);y=linspace(-1,1,20);
>> [X Y] = meshgrid(x,y);Z=f2(X,Y);
>> surf(X,Y,Z)
```

så ser vi att dess funktionsyta har höga toppar nära linjen $x+y=0$. Observera att funktionen inte är definierad i punkter (x,y) där $x+y=0$. Trots att ytan verkar ganska okej nära origo så saknar den gränsvärde där ty $f(x,0) \to 0$ och $f(x,x^2-x) \to 2$ då $x \to 0$ (visa det!). □

En funktion kan emellertid ha gränsvärde i en punkt utan att vara definierad där.

Exempel. Betrakta funktionen $f(x,y) = x^3/(x^2+y^2)$.

```
>> f3=@(x,y) x.^3./(x.^2+y.^2);
>> x=linspace(-1,1,21);y=linspace(-1,1,20);
>> [X Y] = meshgrid(x,y);Z=f3(X,Y);
>> surf(X,Y,Z)
```

Funktionen är inte definierad i origo men vi har

$$|f(x,y)-0| = \left|\frac{x^3}{x^2+y^2}\right| = \frac{|x|x^2}{x^2+y^2} \leq \frac{|x|(x^2+y^2)}{x^2+y^2} = |x| \to 0 \ \text{, då } (x,y) \to (0,0).$$

så $f(x,y) \to 0$ då $(x,y) \to (0,0)$. □

Om vi vill studera en funktion som inte är definierad i en viss punkt `(x0,y0)` så bör denna inte finnas med som en punkt i det rutnät som `meshgrid` genererar. Vi kan kontrollera om så är fallet med kommandot `find((X==x0)&(Y==y0))`. Svaret på detta kommando är det/de index som ger den kritiska punkten. Om detta är annat än `[ ]` så kan vi förskjuta rutnätet en liten sträcka i x-led med kommandot `x=x+eps;`. Om vi vill undvika division med noll så kan vi alternativt addera `eps` i nämnaren på funktionen. Värdet på `eps` är förinställt och är det minsta representerbara positiva talet i MATLAB.

1.2 Nivåkurvor

Ett annat vanligt sätt att åskådliggöra en funktion av två variabler, som inte kräver att man avbildar tredimensionellt, är att rita nivåkurvor. För olika värden på konstanten c så ger lösningsmängden till ekvationen $f(x,y) = c$ en nivåkurva i xy-planet. Nivåkurvan $f(x,y) = c$ är helt enkelt mängden av alla punkter (x,y) för vilket funktionsvärdet är c. Man får på detta sätt en tvådimensinell (topografisk) karta, med vars hjälp man kan utläsa funktionsytans utseende. T.ex. använder vanliga orienteringskartor detta sätt för att beskriva olika höjder i naturen. Tekniken används också för bland annat väderkartor. Där förekommer isobarer och isotermer som nivåkurvor till de funktioner som beskriver lufttrycket respektive temperaturen på olika platser.

Om vi t.ex. vill titta på några nivåkurvor till funktionen $f(x,y) = x\sin y^2$ så måste vi som förut börja med att beräkna funktionsvärdena i en massa punkter enl.

```
>> f=@(x,y) x.*sin(y.^2);
>> x=linspace(-1,2,21);y=linspace(0,3,20);
>> [X Y] = meshgrid(x,y);
>> Z=f(X,Y);
```

Sedan får vi nivåkurvor med t.ex. kommandot

```
>> contour(X,Y,Z)
```

eller något fylligare med kommandot

```
>> contourf(X,Y,Z)
```

Man kan även lyfta upp nivåkurvorna till respektive nivå i rummet med kommandot

```
>> contour3(X,Y,Z)
```

Det är också möjligt att rita både yta och nivåkurvor i samma figur med t.ex. kommandot

```
>> surfc(X,Y,Z)
```

och med följande kommandon ser man nivåkurvor inlagda på ytan.

```
>> mesh(X,Y,Z), hold, contour3(X,Y,Z), hold
```

Det går också bra att själv styra vilka nivåkurvor som skall ritas. Låt oss illustrera detta med ett exempel.

Exempel. Om vi vill plotta nivåkurvorna $yx^4 + 3xy^4 = a$, för $a = 1, 5, 10$, så ger vi kommandona

```
>> f=@(x,y) y.*x.^4+3*x.*y.^4;
>> x=linspace(-3,3,21);y=linspace(-3,3,20);
>> [X Y] = meshgrid(x,y);
>> Z=f(X,Y);
>> contour(X,Y,Z,[1,5,10])
```

Om vi bara vill plotta en nivåkurva så måste man ange nivåvärdet två gånger t.ex. plottar vi nivåkurvan $yx^4 + 3xy^4 = 6$ med kommandot

```
>> contour(X,Y,Z,[6,6])
```

□

1.3 Funktioner av tre variabler

Det är lite svårare att visualisera funktioner av tre variabler. Vi kan inte rita grafen på samma sätt som för funktioner av en eller två variabler eftersom avstånd bara går att illustrera geometriskt i högst tre dimensioner. Det finns emellertid andra sätt. I avsnitt 1.4 skall vi se hur funktioner av tre variabler kan illustreras genom att plotta några nivåytor. Ett annat sätt är att använda färgen (istället för avstånd) för att ange funktionsvärdena utefter några plan i rummet. Vi kan då använda kommandot `slice`. Följande kommandon illustrerar funktionen $f(x,y,z) = xy - z$ utefter planen $x = -1.2, x = 0.8, x = 2, y = 2, z = -2$ och $z = -0.2$

```
>> f=@(x,y,z) x.*y-z;
>> x=linspace(-2,2,21);
>> y=linspace(-2,2,20);
>> z=linspace(-2,2,20);
>> [X,Y,Z] = meshgrid(x,y,z);
>> slice(X,Y,Z,f(X,Y,Z),[-1.2 .8 2],2,[-2 -.2])
```

För att se vilket funktionsvärde som hör till respektive färg så kan vi lägga in en färgskala i figuren med kommandot

```
>> colorbar
```

Om man inte har någon ”slice” i t.ex. z-led så anges en tom matris `[ ]` på motsvarande plats i `slice`-kommandot. T.ex. illustrerar följande kommando funktionens värde utefter planen $x = -1.2, x = 0.8$ och $y = 2$

```
>> slice(X,Y,Z,f(X,Y,Z),[-1.2 .8],2,[])
```

Vi kan också illustrera funktionsvärdena utefter andra ytor än plan t.ex. ger följande kommandon funktionens värden utefter ytan $z = x \sin y + y \cos x$

```
>> g=@(u,v) u.*sin(v)+u.*cos(v);
>> u=linspace(-2,2,21);v=linspace(-2,2,20);
>> [U,V] = meshgrid(u,v);W=g(U,V);
>> surf(U,V,W,f(U,V,W)), colorbar
```

Färgen i varje punkt (x, y, z) på ytan anger alltså värdet på funktionen $f(x, y, z)$. Alternativt kan vi plotta nivåkurvorna till funktionen på ytan

```
>> shading interp
>> alpha(0.3)
>> contourslice(X,Y,Z,f(X,Y,Z),U,V,W)
```

Här använde vi bl.a. kommandot `alpha(0.3)`, som gör ytan delvis genomskinlig, för att nivåkurvorna skulle framträda lite tydigare (0 för helt transparant och 1 för helt solid).

1.4 Nivåytor

En nivåyta $f(x, y, z) = c$ är mängden av alla punkter (x, y, z) för vilket funktionsvärdet är c. Sådana nivåytor kan t.ex. användas för att beskriva temperaturnivåerna i ett tredimensionellt objekt. En funktionsyta $z = f(x, y)$ kan om man vill också betraktas som en nivåyta enl. $f(x, y) - z = 0$. För att plotta nivåytor i MATLAB använder vi kommandot `isosurface`.

Exempel. Sfären $x^2 + y^2 + z^2 = 0.5$ är exempel på en nivåyta och vi kan plotta den med följande kommandon

```
>> f=@(x,y,z) x.^2+y.^2+z.^2;
>> x=linspace(-2,2,21);
>> y=linspace(-2,2,20);
>> z=linspace(-2,2,20);
>> [X,Y,Z] = meshgrid(x,y,z);
>> isosurface(X,Y,Z,f(X,Y,Z),0.5)
```

Troligen ser det mer ut som en ellips än en cirkel när ni utför ovanstående kommandon. Skälet är naturligtvis att koordinataxlarna inte är dimensionerade på samma sätt. För att åstadkomma detta så ger vi kommandot

```
>> axis equal
```

Vi kan naturligtvis plotta nivåytor till andra funktioner i samma figur. Om vi t.ex. vill plotta nivåytan $x^2 + y^2 \sin z = 1$ i samma figur som sfären ovan så behöver vi inte beräkna om matriserna `X,Y` och `Z`

```
>> g=@(x,y,z) x.^2+y.^2.*sin(z);
>> T=g(X,Y,Z);
>> isosurface(X,Y,Z,T,1)
```

Vi kan plotta andra nivåytor till samma funktion utan att beräkna om matrisen `T`

```
>> isosurface(x,y,z,T,2)
```

Observera att föregående ytor ligger kvar i figuren när nästa nivåyta plottas, trots att man inte gett kommandot `hold on` (enklaste sättet att radera föregående plottar är att först stänga figurfönstret). Titta gärna på ytorna från lite olika vinklar

genom att rotera figurfönstret. Välj t.ex. Camera Toolbar ▶ Orbit Camera under View i menyraden överst i figurfönstret. □

Om man vill kan man också plotta normalvektorer till en nivåyta. Man kan då använda kommandona `isonormals` och `quiver3` (detta kommando plottar vektorfält som vi skall studera närmare i kapitel 4).

Exempel. Antag att vi vill plotta nivåytan $x^2 + y^2 \sin z = 1$ och ett antal normalvektorer till ytan. Med matriserna `X,Y,Z` och `T` från föregående exempel så åstadkommer vi detta med följande kommandon

```
>> [p,q]=isosurface(X,Y,Z,T,1);
>> isosurface(X,Y,Z,T,1), hold on
>> N=isonormals(X,Y,Z,T,q); N=-N;
>> quiver3(q(:,1),q(:,2),q(:,3),N(:,1),N(:,2),N(:,3))
>> axis equal, hold off
```

Istället för att använda kommandot `isonormals` så kan man naturligtvis även beräkna normalvektorer $\mathbf{N}$ till en nivåyta $f(x, y, z) = c$ med hjälp av gradienten, ty gradienten ger ju vektorer som pekar i den riktning som funktionen växer mest och är (därför) vinkelräta mot nivåytorna. Vi lämnar åt den intresserade att i detta exempel själv beräkna gradienten och fundera ut vilka kommandon som plottar normalerna. □

1.5 Parametriserade kurvor

Vi kan plotta parametriserade kurvor såväl i planet som i rummet.

Exempel. Om vi vill plotta kurvan $x = \sqrt{|\cos 2t\,|}\,\cos t\,,\ y = \sqrt{|\cos 2t\,|}\,\sin t\,,\ 0 \leq t \leq 2\pi$, i planet så ger vi följande kommandon

```
>> t=linspace(0,2*pi,200);
>> x=sqrt(abs(cos(2*t))).*cos(t);
>> y=sqrt(abs(cos(2*t))).*sin(t);
>> plot(x,y)
```
□

Exempel. Om vi vill plotta kurvan $x = e^t \cos 10t\,,\ y = e^t \sin 10t\,,\ z = t\,,\ -5 \leq t \leq 0$, i rummet så ger vi följande kommandon

```
>> t=linspace(-5,0,300);
>> x=exp(t).*cos(10*t);
>> y=exp(t).*sin(10*t);
>> z=t;
>> plot3(x,y,z)
```
□

1.6 Parametriserade ytor

Som tidigare noterats så kan en funktionsyta $z = f(x, y)$ betraktas som en nivåyta, men den kan också betraktas som en parametriserad yta. Det är då naturligt att använda x och y som parametrar enl. $x = u, y = v, z = f(u, v)$. Det är inte heller så stor skillnad på att plotta funktionsytor och att plotta parametriserade ytor. Principen är den samma, beräkna matriser `X,Y,Z` och rita ytan med t.ex. `surf(X,Y,Z)`.

Exempel. Antag att vi vill plotta den yta som ges av parametriseringen

$$\begin{cases} x = u + v \\ y = u^2 - v^2 \\ z = uv \end{cases} \qquad 0 \le u \le 1\ ,\ 0 \le v \le 1$$

För att rita ytan beräknar vi först matriser för parametrarna u, v, och sedan matriser för x, y, z.

```
>> u=linspace(0,1,20);v=linspace(0,1,21);
>> [U,V]=meshgrid(u,v);
>> X=U+V;
>> Y=U.^2-V.^2;
>> Z=U.*V;
>> surf(X,Y,Z)
```
□

Exempel. I avsnitt 1.4 såg vi hur man kan plotta en sfär genom att betrakta den som en nivåyta. Man kan också plotta en sfär genom att beskriva den som en parametriserad yta. Sfären $x^2 + y^2 + z^2 = 4$ kan t.ex. beskrivas med följande parametrisering

$$\begin{cases} x = 2\sin u \cdot \cos v \\ y = 2\sin u \cdot \sin v \\ z = 2\cos u \end{cases} \qquad 0 \le u \le \pi\ ,\ 0 \le v \le 2\pi$$

Sfären kan därför plottas med följande kommandon

```
>> u=linspace(0,pi,20);v=linspace(0,2*pi,21);
>> [U,V]=meshgrid(u,v);
```

```
>> X=2*sin(U).*cos(V);
>> Y=2*sin(U).*sin(V);
>> Z=2*cos(U);
>> surf(X,Y,Z)
```

Vi kan här ändra parameterområdet om vi bara vill plotta en viss del av sfären t.ex.

```
>> u=linspace(pi/4,3*pi/4,20);v=linspace(-3*pi/4,pi,21);
>> [U,V]=meshgrid(u,v);
>> X=2*sin(U).*cos(V);
>> Y=2*sin(U).*sin(V);
>> Z=2*cos(U);
>> surf(X,Y,Z)
```

I ovanstående parametrisering av sfären så är radien hela tiden lika med 2. Om vi låter radien variera med värdena på vinklarna u och v så får vi en annan sluten yta kring origo. Pröva t.ex. följande kommandon

```
>> u=linspace(0,pi,20);v=linspace(0,2*pi,21);
>> [U,V]=meshgrid(u,v);
>> R=(2+sin(2*U))./(2+cos(V));
>> X=R.*sin(U).*cos(V);
>> Y=R.*sin(U).*sin(V);
>> Z=R.*cos(U);
>> surf(X,Y,Z)
```

När man plottar ytor med hjälp av bl.a. `surf` så är det standard att ytan färgläggs med en färg som beror på höjden över xy-planet dvs. värdet på z i respektive punkt. Om man vill kan man också själv styra färgerna på varje liten ytbit. Följande kommando väljer t.ex. färgerna slumpvis

```
>> surf(X,Y,Z,rand(size(Z)))
```

Pröva även att jämna ut ytan och färgerna med

```
>> shading interp
```

□

Exempel. Antag att vi vill plotta nivåytan $x^4 + y^4 + z^4 = 1$. Ekvationen liknar sfärens ekvation så vi kan parametrisera den med något som påminner om sfärens parametrisering (kvadreras uttrycken nedan så erhålls parametrisering av sfären).

$$\begin{cases} x = \text{sign}(\sin(u) * \cos(v))\sqrt{|\sin(u) * \cos(v)|} \\ y = \text{sign}(\sin(u) * \sin(v))\sqrt{|\sin(u) * \sin(v)|} \\ z = \text{sign}(\cos(u))\sqrt{|\cos(u)|} \end{cases} \qquad 0 \le u \le \pi \ , \ 0 \le v \le 2\pi$$

För att plotta ytan ger vi därför följande kommandon i MATLAB.

```
>> u=linspace(0,pi,20);v=linspace(0,2*pi,21);
>> [U,V]=meshgrid(u,v);
>> X=sign(sin(U).*cos(V)).*sqrt(abs(sin(U).*cos(V)));
>> Y=sign(sin(U).*sin(V)).*sqrt(abs(sin(U).*sin(V)));
>> Z=sign(cos(U)).*sqrt(abs(cos(U)));
>> surf(X,Y,Z)
```
□

Det går naturligtvis även bra att plotta rotationsytor.

Exempel. Betrakta den kurva som ges av $x = t\sin t, y = t + \cos t, -3 \le t \le 4$. Följande kommandon plottar den yta som bildas då kurvan roterar kring y-axeln.

```
>> t=linspace(-3,4,20);v=linspace(0,2*pi,21);
>> [T,V]=meshgrid(t,v);
>> X=T.*sin(T).*cos(V);
>> Y=T+cos(T);
>> Z=T.*sin(T).*sin(V);
>> surf(X,Y,Z)
```
□

Om parameterområdet till en parametriserad yta inte är en axelparallell rektangel så kan man utöka området till en sådan och lägga in bilden av begränsningskurvorna i bilden av rektangeln.

Exempel. Låt oss rita den yta som ges av $x = u\cos v, y = u\sin v, z = uv, u^2 \le v \le 4$. Vi väljer en rektangel som omfattar området: $-2 \le u \le 2, \quad 0 \le v \le 4$ och ritar som ovan men beräknar och ritar samtidigt bilden av randkurvan där $v = u^2$.

```
>> u=linspace(-2,2,20);v=linspace(0,4,21);
>> [U,V]=meshgrid(u,v);
>> X=U.*cos(V);Y=U.*sin(V);Z=U.*V;
>> surf(X,Y,Z), hold on
>> T=linspace(-2,2,200);
>> Xr=T.*cos(T.^2);Yr=T.*sin(T.^2);Zr=T.^3;
>> plot3(Xr,Yr,Zr,'-k'), hold off
```

Om vi vill kan vi ta bort den del av ytan som motsvarar parametervärden som inte uppfyller villkoret $u^2 \leq v$. I så fall byter vi ut de värden i matrisen `Z` vars motsvarande element i matriserna `U` och `V` är sådana att $u^2 > v$, mot värdet `NaN` (Not a Number). När MATLAB plottar så utesluts nämligen linjer till sådana värden.

```
>> Z(find(U.^2>V))=NaN;
>> surf(X,Y,Z), hold on
>> plot3(Xr,Yr,Zr,'-k'), hold off
```

□

Om man vill kan man också plotta normalvektorer till en parametriserad yta. Man kan då använda kommandona `surfnorm` och `quiver3`.

Exempel. Följande kommandon plottar funktionsytan $z = xe^{-x^2-y^2}$ och ett antal normalvektorer till ytan.

```
>> x=linspace(-2,2,20);y=linspace(-1,1,21);
>> [X,Y]=meshgrid(x,y);Z=X.*exp(-X.^2-Y.^2);
>> [Nx,Ny,Nz]=surfnorm(X,Y,Z);
>> quiver3(X,Y,Z,Nx,Ny,Nz),hold on
>> surf(X,Y,Z), hold off
>> shading interp, axis equal
```

Istället för att använda kommandot `surfnorm` så kan man naturligtvis även beräkna normalvektorer $\mathbf{N}$ till en parametriserad yta $\mathbf{r} = \mathbf{r}(u, v)$, där $\mathbf{r}(u, v) = (x(u, v), y(u, v), z(u, v))$, med hjälp av formeln $\mathbf{N} = \mathbf{r_u} \times \mathbf{r_v}$. Men eftersom `cross` som beräknar vektorprodukter i MATLAB inte utför elementvisa operationer på ”vanligt” sätt så blir detta lite mer komplicerat. Vi lämnar därför detta till läsaren själv att fundera ut (i mån av tid). □