Matlabövning

Inledning

Denna datorövning ger en introduktion till Matlab. Systemet används här som en avancerad räknedosa med inbyggda matrisoperationer och grafik. Ha den Matlabmanual tillgänglig som du är van vid.

Övningarna är tänkta att genomföras på egen hand. Om du stöter på problem fråga då någon av lärarna på kursen.

Matlab och Funktionsteori

Matlab är från början ett skal kring ett programbibliotek som innehåller rutiner från lineär algebra. Några fördelar med Matlab som gör systemet lämpligt för experimentell matematik i vår kurs och i andra kurser är:

- Matlab har en enkel syntax och många av er har använt programmet tidigare. Inga deklarationer behövs.
- I Matlab kan man direkt räkna med komplexa tal.
- Komplicerade operationer på vektorer och matriser, t ex matrisprodukt och matrisinvers, är direkt tillgängliga med ett enda kommando (`A*B` respektive `inv(A)`).
- Matlab har inbyggt kraftfulla grafiska kommandon, som gör det lätt att åskådliggöra resultat av beräkningar.
- Matlab innehåller ett programmeringsspråk, som gör det möjligt att bygga ut systemet med egna nya funktioner och kommandon.

Matlab används i Lund flitigt av forskare i många områden, matematik, fysik, numerisk analys, reglerteknik, teletransmissionsteori . . . , och du kommer att få använda systemet mera i kommande kurser.

Vi skall längre fram använda ett annat matematiskt program, nämligen Maple. Detta är speciellt duktigt på symboliska (i motsats till numeriska) räkningar.

Nedan ges en kort översikt över (en del av) de delar av Matlab som har direkt anknytning till Funktionsteori. För en mer allsidig introduktion hänvisas till Matlabs systemmanualer, som i senare versioner finns tillgängliga från Matlabsystemet, t ex via kommandot `helpdesk`.

Den grundläggande datastrukturen i Matlab är matriserna. Vi skall här nästan enbart använda *vektorer*, närmare bestämt radvektorer, för att representera *följder*. Följden $\langle 2, 1, 4, -5 + i, 8 - 0.3i \rangle$ representeras i Matlab som radvektorn `[2 1 4 -5+i 8-0.3*i]`.

De flesta följdoperationer har direkta motsvarigheter i Matlab. Nedan följer en liten tabell med några exempel på de viktigaste:

följd	vektor
$\langle 2, 1, 4, -5+i, 8-0.3i \rangle$	[2 1 4 -5+i 8-0.3*i]
$\langle 1, 2, 3, 4, 5, 6 \rangle$	[1 2 3 4 5 6]
Sätt $\mathrm{a} = \langle 1, 2, 3, 4, 5, 6 \rangle$	a = [1 2 3 4 5 6]
Sätt $\mathrm{b} = \langle 1, 1, 1, 1, 1, 1 \rangle$	b = [1 1 1 1 1 1]
$7\mathrm{a} - 3\mathrm{b}$	7*a - 3*b
$\mathrm{a} \cdot \mathrm{b}$ [1]	a.*b
$\langle 1, -2, 3 \rangle \cdot \langle 4, 5, 6 \rangle \; (= \langle 4, -10, 18 \rangle)$	[1 -2 3].*[4 5 6]
$\Sigma \mathrm{a}$	cumsum(a)
$\Pi \mathrm{a}$	cumprod(a)
$\mathrm{x} = \langle k \rangle \,,\; 0 \leq k \leq 10$	x = 0:10
$\langle a + kd \rangle \,,\; 0 \leq k \leq n$	a:d:a+n*d
$\langle ar^k \rangle \,,\; 0 \leq k \leq n$	a*r.^(0:n)

En snitslad väg genom Matlabdjungeln

Var man startar

0.1 Första gången du vill köra Matlab kan det vara klokt att tillverka en katalog /matlab omedelbart under din rotkatalog. I denna kan du eventuellt lägga en fil startup.m med matlabkommandon, som körs varje gång du startar Matlab. Efter hand kommer du att hämta hem m-filer från kursens hemsida. Dessa kan förslagsvis placeras i katalogen /matlab.

0.2 Start och avslutning av Matlab på Linuxdator.

1. Klicka med vänster musknapp på en TV-liknande ikon på skärmens översta rad. Ett terminalfönster öppnas. Skriv där setxkbmap -variant nodeadkeys. Detta behövs för att du i Matlab ska ha tillgång till alla tecken. (Utan kommandot kan du exempelvis inte skriva 2^3.)
2. Klicka med vänster musknapp på Applications i övre vänstra hörnet på skärmen.
3. Välj i fönstret som kommer upp Math och sedan Matlab7.4
4. Skriv i Matlab

[1] Elementvis multiplikation.

```
cd matlab
```

så hittar Matlab de program du kan ha lagrat i katalogen ~/matlab.

5. Avbryt en pågående räkning i Matlab med kommandot Ctrl-c. För att helt stänga Matlab ger du kommandot exit eller quit.

6. Om du vill återställa datorn i det skick den var före Matlabkörningen kan du öppna ett terminalfönster och i det skriva kommandot setxkbmap -variant basic.

7. Då du är klar så klicka på ikonen Desktop på övre raden och välj log out.

Start och avslutning av Matlab på Windows-dator

1. Gå till Start följt av AllPrograms och leta upp MatlabR2007a.

2. Skriv i Matlab

```
cd matlab
```

så hittar Matlab de program du kan ha lagrat i katalogen ~/matlab.

3. Bryt en pågående räkning i Matlab med kommandot Ctrl-c. För att helt stänga Matlab ger du kommandot exit eller quit.

0.3 Bekanta dig med Matlabs syntax. Matlab har en utvidgad mängd av aritmetiska operationer:

+	addition
-	subtraktion
*	multiplikation
/	högerdivision (vanlig division)
\	vänsterdivision
^	potens

Kontrollera att systemet kan lägga ihop 2 och 2 med rätt svar. Vad blir i^2? Lägg speciellt märke till möjligheten att kalla tillbaka gamla kommandon med pil-upp tangenten. Det går också att göra rättelser i dem.

0.4 Matlab har många inbyggda matematiska funktioner. Här följer en lista på några av dem.

Funktion	**Beskrivning med matetematisk terminologi**
abs(x)	$\lvert x \rvert$
sqrt(x)	$\sqrt{x}$
exp(x)	e^x
log(x)	$\ln x$ (Naturliga logaritmen)
log10(x)	$\lg x$ (10-logaritmen)
sin(x)	$\sin x$
cos(x)	$\cos x$
tan(x)	$\tan x$
cot(x)	$\cot x$
asin(x)	$\arcsin x$
acos(x)	$\arccos x$
atan(x)	$\arctan x$
atan2(x,y)	$\arg(x + iy)$, där argumentet väljs i intervallet $(-\pi, \pi]$
sinh(x)	$\sinh x$
cosh(x)	$\cosh x$

0.5 Prova Matlabs hjälpfunktion genom att först ge kommandot help och sedan välja något speciellt ämne t ex help abs eller help help. Skriver man bara help får man en (lång) lista på det mesta man kan be om hjälp om. Använd denna hjälpfunktion varje gång du stöter på något i texten som du inte förstår. Testa nu help inv, help : (help colon) och help . (help arith). Öppna rullgardinen Help och studera där Full, Product, Family Help eller MATLAB Help.

Följder och grafik

0.6 Följden $\left\langle 0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \ldots, 2\pi \right\rangle$ kan i Matlab åstadkommas med

```
t=0:pi/6:2*pi
```

Om du vill beräkna $\sin s$ då $s = 0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \ldots, 2\pi$ så kan alla dessa värden beräknas på en gång med kommandot

```
y=sin(t)
```

Genom

```
plot(t,y)
```

sammanbinds punkter av formen (t_k, y_k) med (t_{k+1}, y_{k+1}) med en rät linje. Här är $t_k = k\frac{\pi}{6}$ och $y_k = \sin(t_k)$ och $k = 0, 1, 2, \ldots, 11$. Om du vill ha en ”vackrare” sinuskurva kan du göra steglängden mindre. Exempelvis kan du skriva

```
t=0:pi/100:2*pi;
plot(t,sin(t))
```

Vad blir effekten av semikolon?

0.7 Rita upp kurvan $\sin(t)$, $-5 \leq t \leq 15$ genom kommandona.

```
t = -5:0.1:15
plot(t,sin(t))
```

Se efter vad sin(t) egentligen är. För att slippa utskrifter av resultaten kan man lägga till ett semikolon på slutet av kommandot, alltså t ex t = -5:0.1:15;.

Rita i stället kurvan $\sin(2t)$ på samma intervall. Prova sedan $\sin(\omega t)$ med $\omega = 50, 100, 200, 500, 501, 502, 503, 504$. Vad kan det vara frågan om? Kan man lita på datorritade figurer? Se läroboken sidorna 2-3 (exempel 1.1) och sidan 56!

0.8 Här följer några enkla manipulationer av följder. Utgå t ex från följden $\mathrm{x} = \langle 3, 7, 5, -2, 6 \rangle$. Om du vill öka alla termer med 4 kan du skriva

```
x=[3 7 5 -2 6]
x+4*ones(size(x))
```

där ones(size(x)) är en vektor med lika många element som hos x. Alla element är 1. Men det går lika bra med

```
x=[3 7 5 -2 6]
x + 4
```

Om du vill kvadrera alla elementen i följden kan du skriva x.*x eller x.^2. Observera punkten!

För att invertera alla elementen i x skriver du 1./x. Notera punkten igen.

Som sista exempel tänker vi oss att talet 2 ska upphöjas till vart och ett av elementen i x. Skriv då i Matlab 2.^x. För att få elementvis operation behövs även här en punkt.

0.9 En aritmetisk summa fås som medelvärdet av första och sista termen multiplicerat med antalet termer. Kontrollera detta för följden $\mathrm{n} = \langle 0, 1, 2, 3, \ldots, 30 \rangle$. Tillverka följderna $\mathrm{b} = \langle n(n+1)/2 \rangle$ och $\mathrm{s} = \Sigma \mathrm{n}$ genom att sätta

```
n=0:30
b=(n.*(n+1))/2
s = cumsum(n)
```

och jämför dem (t ex genom att bilda skillnaden).

0.10 Att förlänga en följd är lätt. Prova till exempel

```
a = [1 2 3 4]
a = [5 a]
a = [a 6]
```

Geometriska och andra summaföljder

0.11 Vi skall nu titta lite på geometriska följder. Skriv in

```
n = 15
k = 0:n
x = 0.8
geomf = x.^k
plot(geomf)
bar(geomf)
axis([0 20 0 1.5])
stairs(geomf)
geomsumma = cumsum(geomf)
plot(geomsumma)
axis([0 20 0 6]) % summan tar större plats i höjdled
bar(geomsumma)
axis([0 20 0 6])
stairs(geomsumma)
```

så får du följden och delsummorna utritade på olika sätt.

Om man vill göra om figurerna ovan med olika antal termer och olika värden på x så är det lämpligt att göra ett litet matlabprogram. Sådana program lägger man i filer med extensionen .m.

0.12 Tillverka en fil med namnet geomsum.m med innehållet

```
geomf = x.^k;
clf
subplot(211)
bar(geomf)
xlabel('foeljd a')
geomsumma = cumsum(geomf);
subplot(212)
stairs(geomsumma)
xlabel('summafoeljd s')
```

och lägg in den i ditt matlabbibliotek. Den kan nu köras med kommandot geomsum. Vill du ändra på värdet av x eller på antalet termer och köra om, så är det bara att skriva t ex

```
k = 0:30;
x = -0.9
geomsum
```

Prova detta och några andra x-värden. Studera sedan finesserna i m-filen. Återställ till slut grafikfönstret med

```
subplot(111), clf
```

0.13 De harmoniska talen utgör den diskreta motsvarigheten till logaritmfunktionen. Se läroboken sidan 17! De är lätta att bilda i Matlab:

```
k = 1:40;
H =  cumsum(1./k)
```

Rita upp dem och rita in logaritmfunktionen i samma diagram. Jämför sedan med logaritmfunktionen.

```
hold off
stairs(H)
hold on
plot(k,log(k))
hold off
plot(k,H-log(k))
```

Upprepa försöket med 40 ersatt med 1000. Tror du nu på Eulers konstant?

0.14 Följden $\left\langle \frac{1}{k!} \right\rangle$ har intressanta egenskaper. Vi skall försöka konstruera den, och börjar med att tillverka fakulteterna. De är ju lösningar till differensekvationen

$$x_n = nx_{n-1},\ x(0) = 1,$$

och kan alltså fås genom att bilda produktföljden till följden

$$\mathrm{a} = \langle 1, 1, 2, 3, 4, 5, 6, \ldots \rangle .$$

Lägg märke till den dubbla ettan i början. När vi sedan har fakulteterna så är det bara att dividera 1 med dem. I Matlab är det enkelt, skriv

```
a = [1 1:14]
fakul = cumprod(a)
invfakul = 1./fakul
invfakulsum = cumsum(invfakul)
```

Observera att Matlab använder samma beteckning för funktionsvärden och matriselement. Till exempel betyder `exp(1)` värdet e^1 av exponentialfunktionen medan `invfakulsum(15)` anger term nummer 15 i följden. Alla Matlabföljder indiceras med början från 1.

Plotta sedan elementen i `invfakul` och dess summaföljd, och se hur värdena av summorna

$$s_n = \sum_{k=0}^{n} \frac{1}{k!}$$

tycks närma sig ett gränsvärde. Vilket tror du? För att testa den hypotes som vi hoppas alla har ställt upp kan man jämföra t ex term 15 (den sista) i följden med e, genom kommandot

```
exp(1) - invfakulsum(15)
```

0.15 Den som vill spara och eventuellt skriva ut de figurer som Matlab ritat kan göra detta med kommandot print. Det finns en mängd optioner, som man får se med help print. Om man t ex vill spara sin figur i en Postscriptfil fil.eps som sedan skall inkluderas i en LaTeXfil (eller i något annat system), så ger man Matlabkommandot

```
print -deps fil
```

eller

```
print -depsc fil
```

om man vill trycka i färg. Vill man ha en fil som skall skickas ensam till en skrivare av någon typ så ger man i stället för optionen -deps i stället -dps, -dljet3 etc.

Rekursionsekvationer

0.16 Matlab har de vanliga kontrollstrukturerna med repetitionsslingor, for, while och logiska val, if. Repetitionsslingor skall man av effektivitetskäl undvika om det är möjligt, men här skall vi använda dem för att lösa en differensekvation. Låt oss börja med övning 2.16:

$$x_n = 1.077x_{n-1} - 200,\ x_0 = 10000.$$

Vi vill finna lösningen under de första 50 åren. Detta kan vi ställa upp på en rad och sedan plotta lösningen med nästa rad på följande sätt:

```
x(1)=10000; for n=2:50,  x(n) = 1.077*x(n-1)-200; end
stairs(0:49,x)
```

Plotsatsen ser ut som den gör, därför att Matlabvektorer nödvändigtvis indiceras med början på 1 medan vi startar vid år 0, och motsvarigheten mellan matematik och Matlab blir därför $x_n \leftrightarrow$ x(n+1).

Variera lite på begynnelseinsättningen och se vad som sker. Lägg märke till skalorna! Speciellt intressanta ting händer nära 2597 kronor. Vad har detta med den konstanta partikulärlösningen att göra?

0.17 Lös rekursionsekvationen

$$x_n = ax_{n-1} - x_{n-2},\ x_0 = 0,\ x_1 = 1$$

med a) a lite mindre än 2 och b) a lite större än 2. Det går att göra på en rad med

```
x(1)=0; x(2)=1; for n=3:50, x(n)=a*x(n-1)-x(n-2); end
```

Försök förklara resultatet. Se sidan 55 i löroboken! Beundra de vackra sinuskurvorna som du får i a) med plot(x).

0.18 (Om du hinner) Lös på motsvarande sätt en olineär andra ordningens differensekvation, t ex den i övning 2.22. Håll noga reda på vad n är, så att du inte kommer ett steg fel. Enklast är att beräkna x(2) för hand och starta iterationen med $n = 3$.

Matriser och lineära ekvationssystem

Matlabs specialitet är matrisräkning. Det finns i princip två typer av matrisoperationer i Matlab, elementvisa operationer (*array operations*) och egentliga matrisoperationer (*matrix operations*). De förra arbetar med en matris eller två av samma storlek, och utför operationen på en plats i taget utan att blanda in andra matriselement. De senare bildar nya element ur många gamla. Typiska exempel på egentliga matrisoperationer är matrisinversion och matrismultiplikation.

0.19 Prova skillnaden mellan t ex `A.*A` och `A*A` (för en kvadratisk matris `A`). Skriv in t ex

```
A = [1 2; 3 4]
```

vilket ger 2×2-matrisen

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Transponering av (*reella*) matriser sker med prim-operatorn, A^T erhålles med `A'`. Bilda A^TA och AA^T för matrisen ovan.

0.20 Man behöver ofta beräkna uttryck av typen $\langle a_1^n, a_2^n, \ldots, a_m^n \rangle$, om $\langle a_1, a_2, \ldots, a_m \rangle$ är givna. Detta är mycket enkelt, använd operationen `.^`. Försök t ex med `[1 2 3 4 5 6].^3`.

Observera att det inte finns någon operation `.+`, eftersom matrisaddition alltid sker elementvis.

Med tillgång till Matlab behöver man inte tycka att lineära ekvationsystem är jobbiga att lösa, åtminstone inte kvadratiska med entydig lösning. Matrisinversen A^{-1} erhålls med `inv(A)`. Ett sätt att lösa det lineära ekvationssystemet $Ax = b$ är alltså att använda formeln $x = A^{-1}b$, vilket översatt till Matlab blir `x=inv(A)*b`. (Detta är i allmänhet inte det mest effektiva sättet att lösa systemet, utan man skall hellre använda Matlabkommandot `x = A\`.)

0.21 Lös ekvationsystemet

$$\begin{cases} x + 2y &= 5, \\ 3x + 4y &= 6 \end{cases}$$

för hand och med Matlab, med båda de angivna metoderna.

0.22 Försök att lösa ekvationssystemet

$$\begin{cases} x + 2y &= 5, \\ 2x + 4y &= 6 \end{cases}$$

för hand och med Matlab, med båda de angivna metoderna.

0.23 Försök att lösa ekvationssystemet

$$\begin{cases} x + 2y &= 5, \\ 2x + 4y &= 10 \end{cases}$$

för hand och med Matlab, med båda de angivna metoderna.

Kan du dra några slutsatser av resultaten? Studera `help slash`.

Polynomekvationer

Med Matlab kan man lätt numeriskt lösa polynomekvationer. Operationen heter `roots`. För att ta ett exempel, får man lösningarna till polynomet $x^3 - 5x + 7$ genom operationen `roots([1 0 -5 7]`. Lägg märke till att en del av dem är komplexa. Man kan rita upp deras lägen i det komplexa talplanet genom följande operationer:

```
plot(roots([1 0 -5 7]),'*')
axis('equal')
```

(Den senare för att få samma skala på reella och imaginära axeln.) Försök nu samma sak med andra polynom, t ex $x^9 - 1$.

Lineära rekursionsekvationer

Det ovanstående kan nu kombineras för att skriva en lösare av begynnelsevärdesproblem för homogena lineära rekursionsekvationer, alltså (i specialfallet 2:a ordningen) av typen

$$x_n + ax_{n-1} + bx_{n-2} = 0, \qquad x_0 = \alpha, \quad x_1\beta.$$

Detta görs ju i följande steg (sid 36–37 i läroboken), åtminstone om det karakteristiska polynomet inte har multipelrötter:

1. Skriv upp den karakteristiska ekvationen.
2. Lös denna.
3. Skriv upp den allmänna lösningen till ekvationen.
4. Sätt in begynnelsevärdena i den allmänna lösningen. Det ger ett kvadratiskt lineärt ekvationssystem.
5. Bestäm konstanterna med hjälp av begynnelsevärdena, dvs lös det lineära ekvationssystemet.
6. Beräkna värdet av lösningen vid den önskade tidpunkten.

Alla dessa steg kan lätt utföras med hjälp av Matlab.

0.24 Lös på detta sätt begynnelsevärdesproblemet

$$x_n = x_{n-1} + x_{n-2}, \qquad x_0 = 1, \qquad x_1 = 1.$$

Bestäm x_5 och x_7 och jämför med direkt iteration.

0.25 (I mån av tid.) En lösning till rekursionsekvationen $x_n + ax_{n-1} + bx_{n-2} = \delta_n$, där $\delta_n = 0$ om $n \neq 0$ och $\delta_0 = 1$, kallas en fundamentallösning. En kausal sådan kan vi finna genom att kräva att

- $x_n = 0$ om $n < 0$
- $x_0 = 1$ (Sätt $n = 0$ i ekvationen!)
- $x_1 = -a$ (Sätt $n = 1$ i ekvationen.)
- $x_n + ax_{n-1} + bx_{n-2} = 0$ om $n > 1$ (H)

En lösning till den homogena ekvationen (H) kan anpassas till begynnelsevärdena $x_0 = 1, x_1 = -a$. Om h är en fundamentallösning är $h * w$, där $*$ betecknar faltning, en partikulärlösning till $x_n + ax_{n-1} + bx_{n-2} = w_n$. Testa dessa idéer i Matlab genom att köra programmet partsol.m, som du kan hämta från kursens hemsida. När du har programmet kan du först skriva help partsol och sedan type partsol. Med det senare kommandot visas hela programkoden. Högerledet w kan du hitta på själv eller välja slumpvis med hjälp av w=rand(1,n) där n är något positivt heltal. När du bestämt dig för w startar du programmet med kommandot partsol(a,b,w), där a och b ersätts med de värden du vill att de ska ha.

Avrundningsfel

De tal som Matlab räknar med är flyttal (enligt IEEE-standard). Detta innebär t ex att för tal av storleksordningen 1 räknar systemet endast(!) med ca 16 decimaler. Till exempel har Matlabs värde på π ett avrundningsfel av storleksordningen 10^{-16}. Detta kan ha märkbara konsekvenser, speciellt då man beräknar ting som borde vara lika med 0.

0.26 Vad är värdet av $\sin(2\pi k)$, om k är ett heltal? Testa detta i Matlab! Gör så här:

```
n = 100;
k = 0:n;
plot(k,sin(2*pi*k))
```

Blev det som du väntade? Sätt $n = 100000$ och gör om experimentet.

Försök också med ett riktigt stort heltal k. Vad blir t ex sin(2*pi*10^15)?

0.27 (Överkurs) Försök att förklara utseendet på figuren i föregående uppgift. (Det har bl a med talrepresentationen i datorn att göra.)

En lärdom man kan dra av experimentet ovan är att även om två beräknade flyttal borde vara lika, så skiljer de sig i regel lite åt. För att testa likhet av flyttal bör man därför i Matlab inte skriva (förstår du inte syntaxen så hjälper nog help if och och t ex help le)

```
if  x == y
  gör en sak
else
  gör en annan sak
end
```

utan något i stil med

```
tol = lämpligt valt litet tal
if abs(x-y)<= tol
  gör en sak
else
  gör en annan sak
end
```

0.28 Vid räkning med reella tal gäller som bekant att $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. I Matlab behöver det inte gälla. Skriv följande kommandon på en rad

```
a=(rand+i*rand)*1e5,b=(rand+i*rand)*1e5,(a*conj(a))*b-a*(conj(a)*b)
```

Kör dessa kommandon några gånger. Använd ”pil uppåt” för att återkalla raden med kommandon. Ganska snart hittar du ett fall då det sista resultatet inte blir 0.

3-D grafik

Repetera några olika sätt att presentera 3-D grafik. Pröva med kommandona `graf2d2` och `graf3d`

0.29 Vi ska nu grafiskt studera sinusfunktionen $w = \sin z$ där z är komplext. Skriv in

```
 x=0:0.1:13;
 y=0:0.1:2;
[X,Y]=meshgrid(x,y);
z=X+i*Y;
figure(1),surf(x,y,real(sin(z)))
figure(2),surf(x,y,imag(sin(z)))
figure(3),surf(x,y,abs(sin(z)))
figure(4),surf(x,y,abs(sin(z))-sqrt(sin(X).^2+sinh(Y).^2))
```

Ser grafen i första fönstret ut som du väntade dig då $y = 0$? Stämmer figuren i fönster 4 med övning 6.4? Kontrollera graderingen av axlarna. Hur stor kan $|\sin z|$ bli?

0.30 I denna övning är det logaritmfunktionen som ska studeras. Vi ritar den på ett cirkulärt område. Skriv in

```
 t=-pi:pi/30:pi;
 r=0.1:0.1:1;
 x=r'*cos(t);
 y=r'*sin(t);
 z=x+i*y;
 w=log(z);
 figure(1),surfc(x,y,real(w))
 figure(2),surfc(x,y,imag(w))
 figure(3),surfc(x,y,atan2(y,x))
```

Vilken gren av logaritmen använder Matlab? Notera att figurerna i fönster 2 och 3 är lika. Ser nivåkurvorna ut som du väntade dig? Använd funktionen `atan2` för att rita $\Im(\log(z))$ om log betyder naturliga grenen av logaritmfunktionen.