Laboration 1, M0043M, HT14

Laborationsuppgifter skall lämnas in senast 21 november 2014.

Introduktion till MATLAB[1]

Förberedelseuppgifter

1. Gör dig bekant med Matlab-manualen – finns för nedladdning på Fronter.
2. På internet finns en instruktiv film `http://www.mathworks.se/videos/getting-started-with-matlab-68985.html`.
3. Läs igenom laborationens teoridel, avsnitt 1-5 nedan. Kör teoridelens exempel.

Teoridel

1 Starta och avsluta MATLAB

MATLAB startas genom att klicka på tillhörande ikon.

Kommandon i MATLAB ges efter promptern i kommandofönstret

```
>>
```

och utförs när return-tangenten trycks ned.

För att få en kort demonstration av vad MATLAB kan göra skriver man

```
>> demo
```

följt av return.

För att avsluta MATLAB skrivs `quit` följt av return

```
>> quit
```

I det följande undviker vi att skriva ut promptern och returnkommandot eftersom detta ger svårläst text.

Om man vill avbryta en pågående beräkning i MATLAB håller man ctrl-tangenten nedtryckt samtidigt som man trycker på ”c”-tangenten.

[1] Bearbetning av Jönsson, P. *Laborationer i* MATLAB, 2001.

2 Räkning med tal

MATLAB kan användas som en miniräknare. För att multiplicera talen 12 och 13 skriver vi

```
12*13
```

Resultatet läggs i en variabel *ans* som skrivs ut på skärmen

```
ans =
156
```

Upphöjt till markeras med

```
^
```

För att beräkna 3^4 skriver vi

```
3^4
```

varvid MATLAB returnerar

```
ans =
81
```

Normalt placerar man resultat i egendefinierade variabler.

```
x=pi
x =
3.1416
y=x/2
y =
1.5708
```

Första kommandot tilldelar x värdet π medan andra kommandot tilldelar y värdet $x/2 = \pi/2 \approx 1.5708$.

Ett stort antal funktioner är definierde i MATLAB, se avsnitt 3.5. Om vi skriver

```
z=sin(y)
```

returnerar MATLAB

```
z =
1
```

Variabeln z har tilldelats värdet $\sin(y)$ vilket är lika med 1 då $y = \pi/2$.

3 Vektorer

Antag att vi vill plotta funktionen $y = f(x) = \sin(2\pi x)$ i intervallet $[0, 1]$. För att göra detta måste man:

1. definiera ett antal x-värden i intervallet

$$0 = x_1 < x_2 < \ldots < x_n = 1$$

2. beräkna funktionsvärdet för varje x-värde

$$y_k = f(x_k), \quad k = 1, \ldots, n.$$

3. rita polygonlinjen som förbinder punkterna $(x_1, y_1), \ldots, (x_n, y_n)$.

Ovanstående exempel visar att det är viktigt att kunna generera följder av x- och y-värden. Vi börjar med att diskutera hur detta kan göras i MATLAB.

3.1 Generering av vektorer

En vektor x är en samling av reella tal

$$x = (x_1, x_2, \ldots, x_n).$$

I MATLAB är längden av vektorn är lika med antalet element. För att generera vektorn

$$\mathbf{x} = \begin{bmatrix} 1.1 \\ 2.2 \\ 3.3 \end{bmatrix}$$

och bestämma antalet element i vektorn skriver man

```
>> x=[1.1 2.2 3.3]

x =

    1.1000    2.2000    3.3000

>> length(x)

ans =

     3
```

Observera att `length(x)` inte är lika med $|x|$.

De tre elementen är nu lagrade i vektorn med namnet `x`.

För att inspektera exempelvis element två skriver man

```
x(2)
```

MATLAB svarar då

```
ans =
2.2000
```

vilket är värdet av elementet. Om vi vill ändra värdet av element två till ett nytt värde, säg 4.2, skriver vi

```
x(2)=4.2
```

varvid MATLAB returnerar den nya vektorn

```
x =
1.1000 4.2000 3.3000
```

I många fall, speciellt när man jobbar med stora vektorer, är det önskvärt att kunna stänga av MATLABs svarsutskrifter på skärmen. Det görs genom att avsluta ett kommando med semikolon. Till exempel genererar kommandot

```
y=[pi 0 3];
```

en vektor $y = (\pi, 0, 3)$ utan att resultatet visas på skärmen.

3.2 Kolon-notation

Det finns ett flertal användbara kommandon för att generera vektorer. Kommandot

```
x=20:1:22
```

är ekvivalent med

```
x=[20 21 22]
```

Avståndet mellan elementen i vektorn kallas *steglängden*. Vektorn ovan har steglängd ett. För att generera en vektor med steglängd 0:5 skriver man

```
x=20:0.5:22
```

vilket är ekvivalent med

```
x=[20.0 20.5 21.0 21.5 22.0]
```

Den allmänna formen av kolon-notationen är

```
x=a:h:b
```

där a är startvärdet, h steglängden och$b$ slutvärdet. Notera att även negativa steglängder är tillåtna. Till exempel är

```
x=10:-2:0
```

ekvivalent med

```
x=[10 8 6 4 2 0]
```

3.3 Linspace

För att generera vektorer med ett bestämt antal element finns flera möjligheter. Kommandot

```
x=linspace(a,b,n)
```

ger en vektor med n element jämnt fördelade i intervallet $[a, b]$. Till exempel är

```
x=linspace(0,1,11)
```

ekvivalent med

```
x=[0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]
```

3.4 Aritmetiska operationer på vektorer

Antag att vi har en vektor $\mathbf{t} = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$. Kommandona

```
u=3*t
v=t+2
```

ger $\mathbf{u} = \begin{bmatrix} -3 \\ 0 \\ 6 \end{bmatrix}$ och $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$. I första fallet har alla elementen i $\mathbf{t}$ har multiplicerats med tre, i andra fallet har talet två adderats till alla elementen i $\mathbf{t}$.

Antag nu att vi har två vektorer som innehåller lika många element, t. ex. $\mathbf{s} = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}$ och $\mathbf{t} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$.

Kommandona

```
u=s+t
v=s.*t
w=s./t
z=t.^2
```

ger $\mathbf{u} = \begin{bmatrix} 1 \\ 4 \\ 6 \end{bmatrix}$, $\mathbf{v} = \begin{bmatrix} -2 \\ 3 \\ 8 \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} -1/2 \\ 1/3 \\ 1/2 \end{bmatrix}$ och $\mathbf{z} = \begin{bmatrix} 4 \\ 9 \\ 16 \end{bmatrix}$.

I första fallet har vektorerna $\mathbf{s}$ och $\mathbf{t}$ adderats elementvis. I andra och tredje fallet har vektorerna multiplicerats respektive dividerats elementvis. I det sista fallet har samtliga element i $\mathbf{t}$ kvadrerats.

Provkör ovanstående exempel på egen hand som nyttig övning.

Observera att man vid elementvis multiplikation, division och kvadrering av vektorer måste använda punktnotation.

Till exempel ger kommandot

```
v=s*t
```

upphov till följande felmeddelande från MATLAB

```
??? Error using ==> *
Inner matrix dimensions must agree.
```

Att man har glömt punkten vid elementvis multiplikation, division eller kvadrering av vektorer är ett av de allra vanligaste felen vid arbete med MATLAB.

Man skall också vara medveten om att det inte går att addera, multiplicera eller dividera vektorer med olika längd.

3.4.1 Skalär- och vektorprodukt, längd

MATLAB har ett antal inbyggda funktioner, knutna till vektorer. Vi ska titta närmare på några av dem i nästkommande laboration. Låt oss exemplifiera.

- Funktionen `dot(u,v)` returnerar skalärprodukten $\mathbf{u} \bullet \mathbf{v}$,
- Funktionen `cross(u,v)` returnerar vektorprodukten $\mathbf{u} \times \mathbf{v}$,
- Funktionen `norm(u)` returnerar vektorlängden $|\mathbf{u}|$.

Prova dessa funktioner på vektorerna $\mathbf{s}$ och $\mathbf{t}$ ovan.

3.5 Funktionsevaluering

MATLAB har ett stort antal inbyggda matematiska funktioner. En del av dessa funktioner är listade nedan (för en mera komplett lista se t ex Jönsson s. 30 resp s. 68.)

`abs(x)`	ger absolutbeloppet av x, dvs x.
`sqrt(x)`	ger kvadratroten av x, dvs $\sqrt{x}$.
`exp(x)`	ger exponentialfunktionen av x, dvs e^x.
`log(x)`	ger naturliga logaritmen av x, dvs $\ln x$.
`log10(x)`	ger 10-logaritmen av x, dvs $\lg x$.
`sin(x)`	ger $\sin x$ där x är i radianer.
`cos(x)`	ger $\cos x$ där x är i radianer.
`tan(x)`	ger $\tan x$ där x är i radianer.
`cot(x)`	ger $\cot x$ där x är i radianer.
`asin(x)`	ger $\arcsin x$, dvs $\sin^{-1} x$.
`acos(x)`	ger $\arccos x$, dvs $\cos^{-1} x$.
`atan(x)`	ger $\arctan x$, dvs $\tan^{-1} x$.

Funktionerna i MATLAB accepterar förutom tal även vektorer som argument. Om vi till exempel definierar en vektor

```
x=1:1:10
```

och skriver

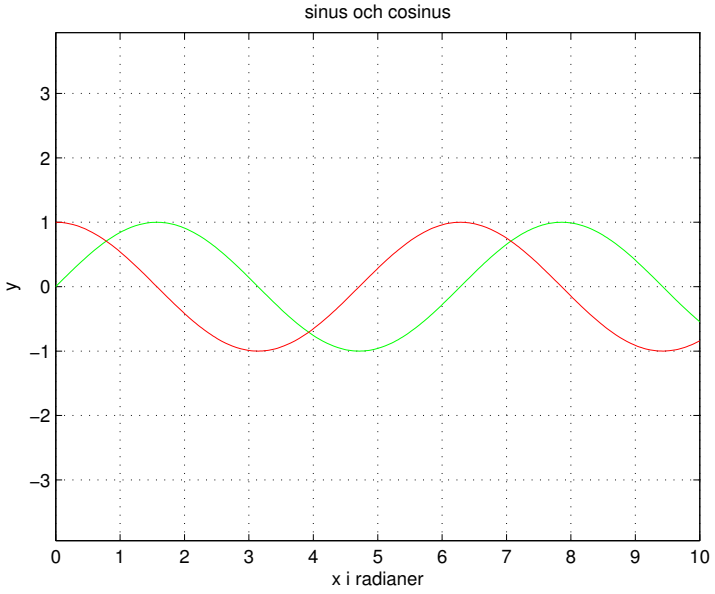
```
y=sqrt(x)
```

får vi en vektor y vars element består av kvadratrötterna av heltalen från ett till tio.

4 Plottning

För att plotta funktionerna $y = \sin x$ och $y = \cos x$ i intervallet $0 \leq x \leq 10$ med heldragna linjer med olika färg skriver man

```
>> x=linspace(0,10,300);
>> f=sin(x);
>> g=cos(x);
>> plot(x,f,'g',x,g,'r')
>> xlabel('x i radianer')
>> ylabel('y')
>> title('sinus och cosinus')
>> axis equal
>> grid on
```



Figur 1: Funktionen $y = \sin x$ ritad med heldragen grön linje, tillsammans med $y = \cos x$ ritad med heldragen röd linje.

Första kommandot genererar en vektor x med 300 element jämnt fördelade från 0 till 10. Andra och tredje kommandot genererar vektorer f och g med motsvarande funktionsvärden. Tredje kommandot ritar ut talparen (x_i, f_i) respektive (x_i, g_i) och förbinder dem med heldragna linjer. De sista kommandona lägger in rubrik, skriver text på axlarna, använder samma axelskala samt lägger på ett rutnät1, varvid man slutligen får Figur 1. Strängen `'g'` genererar grön färg, medan `'r'` genererar röd färg.

Den allmänna formen av plotkommandot är

```
plot(x,y,'str')
```

där `str` är en teckensträng som talar om vilken linjetyp, punkttyp eller färg man önskar på kurvan. De olika tillvalen är listade i nedanstående tabell.

Punkttyper		Linjetyper	
`.`	punkt	`-`	heldragen linje
`*`	asterisk	`--`	streckad linje
`square`	fyrkant	`-.`	punkt-streckad linje
`diamond`	ruta	`:`	prickad linje
`hexagram`	sexuddig stjärna	**Färgtyper**	
`o`	ringar	`g`	grön
`+`	plustecken	`m`	magenta
`x`	kryss	`b`	blå
`<`	vänsterpekande triangel	`c`	cyan
`>`	högerpekande triangel	`k`	svart
`^`	uppåtpekande triangel	`y`	gul
`v`	nedåtpekande triangel	`r`	röd

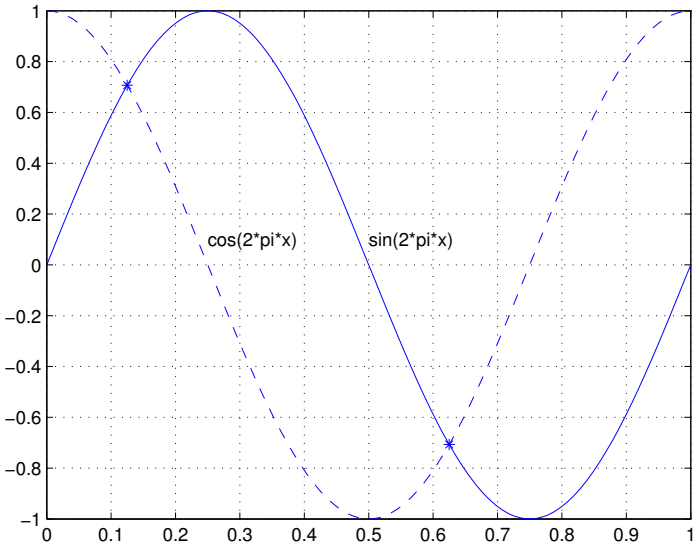
Förutom att kontrollera punkt- linje- och färgtyp kan man också skriva text på axlar och i graffönstret. Vidare kan man skriva en ruta med förklaringar till kurvorna och lägga in ett rutnät i graffönstret. Kommandona som styr detta är beskrivna i nedanstående tabell.

`hold on`	håller kvar graffönstret så att man kan rita flera funktioner i samma fönster
`hold off`	avslutar kvarhållningen av graffönstret
`grid on`	ritar ett rutnät i graffönstret
`grid off`	tar bort rutnätet från graffönstret
`title(txt)`	skriver ut teckensträngen `txt` överst i graffönstret
`xlabel(txt)`	skriver ut teckensträngen `txt` under x-axeln
`ylabel(txt)`	skriver ut teckensträngen `txt` under y-axeln
`text(x,y,txt)`	skriver teckensträngen `txt` i position (x, y) på skärmen
`gtext(txt)`	ger användaren möjlighet att placera text i graffönstret med hjälp av musen
`legend(txt)`	skriver en ruta med förklaringar till kurvorna i graffönstret
`axis`	ger användaren möjlighet att välja skalning på axlarna

I exempelvis Jönsson, kap 5, står mer att läsa om plottning och grafik.

Exempel. I följande exempel visas hur man kan använda en del av ovanstående kommandon.

```
x=linspace(0,1,100);
y1=sin(2*pi*x); y2=cos(2*pi*x);
plot(x,y1)
hold on
plot(x,y2,'--')
plot(1/8,1/sqrt(2),'*')
plot(5/8,-1/sqrt(2),'*')
grid on
text(0.25,0.1,'cos(2*pi*x)')
text(0.5,0.1,'sin(2*pi*x)')
```



Figur 2: Funktionerna $y = \sin(2\pi x)$ och $y = \cos(2\pi x)$ ritade med olika linjetyper.

Kommandot `plot(x,y1)` använder en heldragen linje för att rita $y = \sin(2\pi x)$. Kommandot `hold on` låser graffönstret och tillåter fler plottar i samma figur. Kommandot `plot(x,y2,'--')` använder en streckad linje för att rita $y = \cos(2\pi x)$. Kommandona `plot(1/8,1/sqrt(2),'*')` och `plot(5/8,-1/sqrt(2),'*')` placerar asterisker vid skärningspunkterna $(1/8, 1/\sqrt{2})$ och $(5/8, 11/\sqrt{2})$. De två sista kommandona placerar förklarande textsträngar i positionerna $(0.25, 0.1)$ och $(0.5, 0.1)$ så att vi slutligen erhåller Figur 2.

5 M-filer och funktionsfiler

Det är ofta bekvämt att skriva kommandon i en *scriptfil*. Genom att skriva namnet på scriptfilen vid kommandopromtern kommer kommandona i filen att utföras i ordning. I MATLAB har scriptfiler alltid suffixet (extension) `.m` och kallas därför M-filer. M filer skapas i tre steg:

1. aktivera MATLABs *Editor/Debugger* genom att gå upp överst i MATLABs kommandofönster och klicka på figuren som föreställer ett vitt pappersark.
2. skriv in kommandona i editorn.
3. spara filen genom att gå upp överst i editor-fönstret och klicka på ikonen som föreställer en diskett.

Observera att namn på scriptfiler ej får börja med siffror. Namn får ej heller innehålla punkter annat än för att martkera suffix. Tex är `1a.m` och `uppgift1.1.m` otillåtna namn på M-filer. Man ska också se till att man inte ger sin M-fil ett fördefinierat namn. Det är alltså olämpligt att döpa en M-fil till `sin.m` eftersom `sin` står för den fördefinierade standardfunktionen `sin(x)`.

5.1 Funktionsfiler

Funktionsfiler är en speciell typ av M-filer som definierar en funktion av en eller flera variabler. Som ett exempel kan vi ta funktionen $f(x) = x^2 \sin(x)$. Om vi refererar till funktionen som `fun1` så ges denna av funktionsfilen

```
function f=fun1(x)
f=x.^2.*sin(x);
```

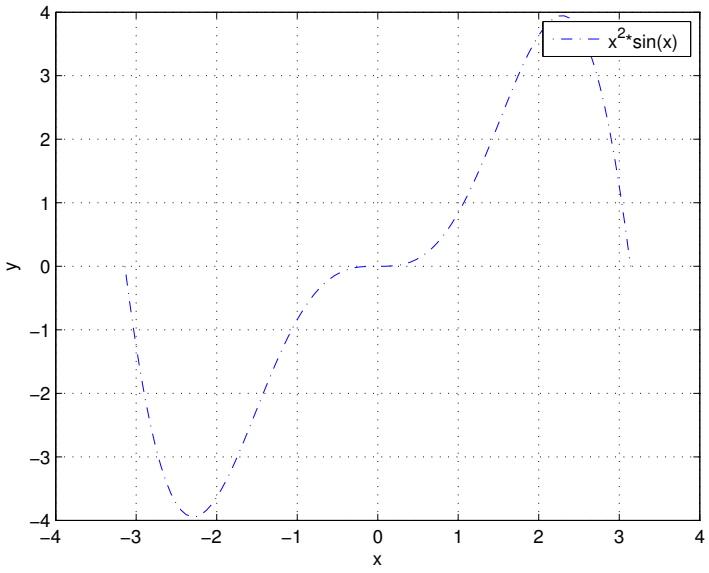
Filen skall alltid ges samma namn som funktionen. I detta fallet döper vi filen till `fun1.m`

Funktioner definierade i funktionsfiler kan anropas av andra M-filer. Följande M-fil (med namnet `exempel.m`) ritar kurvan över funktionen `fun1` i intervallet$[-\pi, \pi]$.

```
x=linspace(-pi,pi);
y=fun1(x);
plot(x,y,'b-.')
grid on
xlabel('x')
ylabel('y')
legend('x^2*sin(x)')
```

I kommandofönstret gör vi anropet

```
>> exempel
```



Figur 3: Funktionen $y = x^2 \sin(x)$ ritad med blå punkt-streckad linje.

Kommandot `legend('x^2*sin(x)')` skriver en ruta med förklarande text till den aktuella kurvan.

6 Uppgiftsdel

Anvisningar

- Följ anvisningarna i "Labb-PM, VT14", som du kan ladda ner från Fronter.
- Lämna in en så enkel rapport som möljigt, utan – **detta är viktigt** – att utelämna matlab-kod, plottar och körningsresultat.
- Rapporten ska vara ett pdf-dokument (Konvertera till pdf från lämpligt ordbehandlingsprogram).
- OBS Viktigt Glöm inte namn på gruppmedlemmar och gärna epostadresser.
- Inlämning sker därefter i Fronter, under "Inlämning".

  Namnge dokumentet så att identifiering lätt kan ske.
- Uppgifterna 1-3 görs på egen hand och skall ej inlämnas.
- Uppgifterna 4-6 är obligatoriska och ska göras i form av M-filer, enligt metoden i avsn. 5.1 ovan.

Laborationsuppgifter–valfria

1. Det är en bra vana att utforska alla optioner för ett kommando via `doc`. Testa följande

   ```
   doc plot
   doc hold
   doc title
   doc axis
   ```

2. Ge kommandot

   ```
   x=-1:0.1:1
   ```

   och exekvera sedan följande kommandon och observera vad som händer.

   ```
   x',  x.^2,  plot(x,x.^3),  x^2,  x'*x,  x*x',  x*x,  [4 12 20]./[2 3 4]
   ```

3. Ge kommandot

   ```
   A=[1, 5, 8; 18, 21, 33; 7, 9, 10]
   ```

   och exekvera sedan följande kommandon och observera vad som händer.

   ```
   A(2,1),  A(3,3),  A(2,:), A(:,3), A(1:2,2:3), A(:), A(:,[1 2 1]),
   A([2 1 2], [3 1 2])
   ```

Laborationsuppgifter–obligatoriska

4. I en kommande matematikkurs får vi lära oss något om hur man approximerar en funktion $f(x)$ med speceiella polynom, s.k. Taylorpolynom.

   Plotta funktionen $f(x) = \sin x$ tillsammans med de tre första Taylorpolynomen

   $$P_1(x) = x, \quad P_3(x) = x - \frac{x^3}{6}, \quad P_5(x) = x - \frac{x^3}{6} + \frac{x^5}{120}$$

   för $-\pi \leq x \leq \pi$.

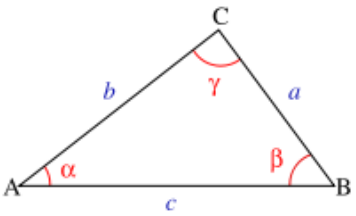
   Föreskrifter: Funktionen $f(x)$ ska vara heldragen i blå, medan $P_1(x)$ ska vara heldragen i rött, $P_3(x)$ ska vara heldragen i grönt och $P_5(x)$ ska vara heldragen i magenta, Gör plotten av de 4 kurvorna tydlig genom att använda lämplig förklaring som titel, 'legends' etc.

   Samla kommandona i en M-fil med namnet `uppg4.m` och kör M-filen genom att anropa den från kommandofönstret.

5. Den grekiske matematikern Heron upptäckte en algoritm för att kunna bestämma arean A av en triangel. Herons algoritm beskrivs av följande formel

   $$A = \frac{1}{4}\sqrt{4a^2b^2 - (a^2 + b^2 - c^2)^2}$$

   där a, b, c är triangelns tre sidor enligt nedanstående figur.

   Skriv en funktionsfil `area.m` som bestämmer arean av en triangel med Herons formel. Funktionsfilen anropas i kommandofönstret.

```
function z=area(a,b,c)
         % Herons algoritm
```

   Använd funktionsfilen för att bestämma arean av triangeln med sidorna

   (a) $a = 6, b = 8, c = 10$

   (b) $a = 9, b = 9, c = 9$.

   Triangelns sidlängder ges i kommandofönstret, exempelvis

```
>> a=7;
>> b=15;
>> c=20;
>> A=area(a,b,c)
```

6. Bestäm med Matlab vinkeln mellan vektorerna

$$\mathbf{u} = \begin{bmatrix} -1 \\ -2 \\ 1 \end{bmatrix} \quad \text{och} \quad \mathbf{v} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

Observera: Svaret skall anges i *grader*.

Samla kommandona i en M-fil med namnet `uppg6.m` och kör M-filen genom att anropa den från kommandofönstret.