Lunds Universitet
Ekonomihögskolan
Statistiska institutionen

STAN42

Kort introduktion till MATLAB

1 Syfte

Denna korta självstudiekurs är endast avsedd som en kort introduktion för att hjälpa dig att komma igång med MATLAB.

MATLAB har idag blivit en innehållsrik nummerisk plattform och det är idag mer eller mindre nödvändigt att ha tillgång till utförliga och välskrivna handledningar för att få en överblick över och till fullo kunna utnyttja programmet.

MATLAB finns på de flesta datorer vid LU och LTH. till fördelarna med programmet hör att det ser i stort sett likadant ut oberoende av på vilken sorts dator man använder. Kompatibiliteten mellan olika datorsytem är också smärtfri. Det går också att förstärka MATLAB med rutiner skrivna i programmeringsspråk som tex Phyton, R, C samt C++.

2 Bakgrund

MATLAB är ett interaktivt program för numeriska beräkningar. Det är också ett programmeringsspråk med en instruktionsrepertoar liknande den som finns i C++ och C.

Det grundläggande matematiska objektet i MATLAB är matrisen, och många funktioner kan ha matriser som argument (vektorer och skalärer betraktas som matriser). I MATLAB behandlas dock alltid en skalär (en 1×1-matris) lite okventionellt. Om man till exempel adderar en skalär λ med en matris A av godtycklig dimension $(n \times k)$ tolkas skalären som en matris av motsvarande dimension.

Vi utför alltså i själva verket operationen $A + \lambda \cdot E$, där E representerar en matris med dimensionen $n \times k$, där alla elementen har värdet 1. Detta underlättar räknandet på ett praktiskt sätt, kanske inte alltid så strikt enligt matrisalgebrans regler men ingenjörsmässigt acceptabelt.

3 In- och utloggning

Börja med att logga in på ditt vanliga konto. På Unixdatorer startar man MATLAB genom att skriva `matlab` i ett terminalfönster.

På datorer som kör MS Windows kan man starta MATLAB i startmenyn under "Programs" → "MATLAB R2017b ", alternativt genom att dubbelklicka på ikonen **MAT-LAB R2017b**. Då öppnas ett fönster med rubriken **MATLAB**.

Du vet att du har kommit in i MATLAB när du ser prompten
`>>`
i kommandofönstret.

Du måste eventuellt också ge ett speciellt initieringskommando för att MATLAB skall kunna hitta de filer som du behöver under datorlabbarna. Detta kan göras med hjälp av kommandot `path`. För mer information se `help path`. Observera att du ska använda varianten med två inparametrar, dvs. `path(path,'nyväg')`, för att inte tappa bort tidigare inställda sökvägar.

- Passa på att notera hur det går till vid den dator och i det nätverksystemet som du kommer att använda MATLAB i.

MATLAB-sessionen avslutas med kommandot `exit`, `quit` eller genom att välja **Exit** från menyn.

Windows-sessionen avslutas lämpligast genom att välja **Avsluta** från den meny som kommer upp när du klickar på **Start**-knappen i nedre vänstra hörnet. Ett fönster med rubriken **Avsluta Windows** öppnas. Bekräfta att du vill avsluta genom att klicka på **OK**.

Denna utloggningsprocedur kan tyckas något omständlig, men på detta vis ser du till att alla filer som du är rädd om sparats på betryggande sätt.

4 MATLAB's grunder

När det gäller det praktiska användatet av MATLAB finns det några saker som det är bra att ha med sig från första början

- MATLAB arbetar med matriser. Det gäller att tillägna sig en erfarenhet av att använda och formulera problemställningarna med hjälp av matriser. Olika loop-konstruktioner t.ex. som är så vanligt i andra programeringsspråk kan oftast ersättas med hjälp av en eller några få enkla matrisoperationer i MATLAB.

- MATLAB är interaktivt. För att undvika problemet med den interaktiva miljön bör man konsekvent arbeta med `m-filer`. Istället för att skriva om alla de kommandon som är korrekta vid något fel ändrar man bara felet i `m-filen`, vilket är ytterst tidsbesparande. MATLAB's `m-filer` motsvaras av funktioner och procedurer i andra programmeringsspråk som C och Pascal.

- Det finns en väl fungerande manual inbyggd i MATLAB. Den kommer man åt med hjälpkommandot `help` och kan alltid användas i nöd, kommandot `lookfor` har man också stor nytta av. Om du redan har startat MATLAB kan du prova kommandot `help help` och `help lookfor` så får du mer information om hjälpfunktionerna.

- Obs! Om programmet tar lång tid eller verkar ha fastnat så kan man alltid avbryta med `CONTROL-C` i Unix miljö, och i Windows miljö med `CONTROL-C` eller `CONTROL-BREAK`. Det kan emellertid ställa till en massa problem när man tvingar ett program att avbrytas på detta brutala sätt. Så försök i görligaste mån undvika denna utväg.

- Decimaltecknet i Matlab utgörs av punkt, `.`, inte komma, `,`.

- Potenser med basen 10 i uttryck av typen $3{,}2 \cdot 10^{-6}$ förkortas med `e`, exempelvis `3.2e-6` för $3{,}2 \cdot 10^{-6}$.

- Alla funktionsanrop, som `sqrt`, `exp`, `log`, `sin`, `asin`, `cos` och `tan`, kräver parenteser om argumentet, exempelvis `sin(2.29)` för $\sin 2{,}29$.
- Talet $\pi = 3{,}1415\ldots$ finns fördefinierat som `pi`.
- Addition, subtraktion, multiplikation, division och potens skrivs respektive `+`, `-` (bindestreck), `*`, `/` och `^`.
- Prioriteringsreglerna vid addition och multiplikation är de vanliga. Matlab-uttrycket `3+4*5` betyder sålunda $3 + 4 \cdot 5 = 23$, inte $(3 + 4) \cdot 5 = 35$. Parenteser används på vanligt sätt för att ändra prioriteringarna.
- Negation anges med förställt minustecken (bindestreck), exempelvis `-2.34` för $-2{,}34$.
- Flera kommandon kan samsas på en och samma kommandorad om de åtskiljs av kommatecken, `,`, eller semikolon, `;` (se nedan!).

4.1 Start

I princip fungerar MATLAB som en avancerad räknedosa, där man utnyttjar likhetstecknet, `=`, som tilldelningstecken. Vi arbetar alltså i princip med den generella formen

```
>> variabelnamn = uttryck  % följt av ‘‘enter’’ eller ‘‘vagnretur’’
```

Exempelvis

```
>> y = 2*exp(-5)
```

eller ännu enklare bara ett uttryck utan någon tilldelning, i det här fallet

```
>> 2*exp(-5)
```

Inmatningen avslutas med “vagnretur”. Det är först då uttrycket beräknas. MATLAB svarar i det här fallet med resultatet

```
y =
   0.0135
>>
```

respektive

```
ans =
     0.0135
>>
```

Om man avslutat uttrycket med ett semikolon `;` undertrycks utskriften, men värdet finns så klart lagrat i variabeln `y`. Med kommandot `clear y` raderas den ur arbetsminnet. Med kommandot `whos` får du reda på vilka variabler som finns i arbetsminnet. Prova gärna detta. Mer om inmatning och kommandon finns i kap. 2.2-2.4 i Pärt-Enander m.fl. [12].

De vanligaste funktionerna som finns på en mer avancerad räknedosa typ $\exp(\cdot)$, $\sin(\cdot)$, $\cos(\cdot)$, $\log(\cdot)$, $\log10(\cdot)$, ... finns så klart också i MATLAB. En lista på några av de vanligaste inbyggda matematiska funktionerna finns i kap. 2.5 Pärt-Enander m.fl. [12].

4.1.1 Övningsuppgifter med svar direkt

- Beräkna $e^2/\sin(\frac{\pi}{9})$.
- Utnyttja piltangenterna och ändra i föregående uttryck och beräkna istället $e^{-4}/\sin(\frac{\pi}{9})$.
- Beräkna $\sqrt{1.19^2 - 1} + \sin(\frac{\pi}{2}) + e^2$.
- Svar: 21.6042, 0.0536 respektive 9.0341.

Övningar

1. Beräkna $\dfrac{2{,}9 + \sin\frac{3}{2{,}56}}{120 + 6{,}3 \cdot 10^2}$.
2. Beräkna $3{,}28^{-1{,}067 \cdot 1{,}067}$.
3. Beräkna $3{,}28^{-1{,}067} \cdot 1{,}067$.
4. Beräkna $3{,}28^{(-1{,}067)^2}$.
5. Beräkna $(-2{,}34)^{-5}$.
6. Beräkna $\mathrm{e}^{2 \cdot 0{,}075}$.
7. Beräkna $\sqrt[5]{13}$.
8. Vad betyder `3.28^-3.2^-1e-0`? Detta visar att man ibland för tydlighets skull bör använda parenteser vid potensangivelser.
9. Beräkna de två lösningarna till ekvationen $x^2 + 3{,}29x - 10 = 0$.
10. Beräkna $\arcsin 0{,}47$.
11. Funktionen `sin` tar argument angivna i radianer. Beräkna $\sin 14{,}6°$.

4.2 Editering och ändring på kommandoraden

Med de två piltangenterna ↑ och ↓ kan man bläddra bland tidigare givna kommandon. Tangenten ↑ stegar bakåt och tangenten ↓ framåt i följden av tidigare givna kommandon. Har man redan skrivit något på kommandoraden, exempelvis `12+`, då man börjar med piltangenterna ↑ och ↓, kommer man att bläddra endast bland de tidigare givna kommandon som inleds med (i detta fall) `12+`.

Kommandoraden rensas med Ctrl+u (håll Ctrl-tangenten nedtryckt, tryck på u-tangenten, släpp upp bägge tangenterna).

På kommandoraden flyttar man markören ett steg åt höger med piltangenten → och ett steg åt vänster med piltangenten ← (inte backstegningstangenten!). Man flyttar markören till början av kommandoraden med Ctrl+a (alternativt Home-tangenten) och till slutet av kommandoraden med Ctrl+e (alternativt End-tangenten). Ctrl+k tager bort alla tecken på kommandoraden som står till höger om markören. Ett enskilt tecken på kommandoraden borttags genom att markören placeras till höger om tecknet, varefter backstegstangenten nedtrycks. Flera på varandra följande tecken kan borttagas i ett svep genom att backstegstangenten hålls nedtryckt tills önskat antal tecken har försvunnit. Inom kommandoraden kan markören flyttas, förutom med piltangenterna ← och →, också med hjälp av musen: man flyttar helt enkelt mus-cursorn till önskad plats för kommandoradsmarkören och klickar (gäller endast under Windows).

4.2.1 Namngivning och variabeltilldelning

MATLAB kommandon ges alltid med små bokstäver. Däremot kan man använda både små och stora bokstäver när man själv definierar variabler. Att tänka på är att MATLAB som regel skiljer små och stora bokstäver, (dock inte i namn på `m-filer` på PC). Specialtecken kan inte användas, däremot siffror såvida de inte inleder variabelnamnet. Exempelvis fungerar `Kalle2`, däremot inte `2kalle` som variabelnamn.

Om man råkar definiera en variabel eller `m-fil` med samma namn som någon av MATLAB's fördefinierade funktioner (`definierat_namn`) så har det namnet man själv definierat högre prioritet. Vill man då återfå den inbyggda funktionens namn raderar man bara den nya med hjälp av `clear definierat_namn`.

Övningar

12. Varför är det olämpligt att använda exempelvis `pi`, `ans`, `sin` eller `quit` som namn på variabler som man har deklarerat själv?

13. Ibland kan de till synes oförargliga och nära till hands liggande variabelnamnen `i` och `j` vara farliga om man inte deklarerar dem på rätt sätt. Varför? Ledning: Tryck

    ```
    >> i*i, j*j, help i
    ```

4.3 Några sätt att påverka skärmutskrifterna

Exekvera och jämför resultaten från följande två snarlika kommandosekvenser.

```
>> A=3
>> f=440, t=2.18
>> phi=pi/6
>> y=A*sin(2*pi*f*t+phi)
```

och

```
>> A=3;
>> f=440; t=2.18;
>> phi=pi/6;
>> y=A*sin(2*pi*f*t+phi)
```

Av detta förstår vi att om ett kommando avslutas med ett semikolon, `;`, skrivs resultatet inte ut på skärmen. När man har långa uträkningar med många ointressanta delresultat, är det bra att kunna undertrycka onödiga utskrifter. Hädanefter kommer alla delresultat att vara försedda med semikolon.

Svar till beräkningarna visas på skärmen med ungefär fem siffror. Vill man ha fler siffror, ca femton stycken, skriver man

```
>> format long
```

och gör om beräkningarna. Återgång till fem siffror sker med

```
>> format short
```

Här måste sägas någonting viktigt: Antalet siffror som visas på skärmen har ingenting alls att göra med noggrannheten i resultaten av de numeriska beräkningarna.

Övningar

14. Kör och begrunda följande kommandosekvens.

```
>> pi, format long, pi, format short, pi
```

15. Vilka övriga ”format” (`format`) finns det?

16. Vilken inverkan har `format compact`, respektive `format loose`, på utskrifternas utseende?

Denna handledning förutsätter från och med nu att du har valt följande.

```
>> format long, format compact
```

Det kan hända att man behöver skriva någonting på kommandoraden som man inte vill skall exekveras. Det kan vara en minnesanteckning, en förklaring av ett kommando eller bara en grafisk markör. Sådana så kallade kommentarer inleder man med ett procenttecken, `%`. Resten av raden till höger om procenttecknet reserveras för kommentaren och exekveras inte.

Övning

17. Studera nedanstående Matlab-körning försedd med kommentarer. Skriv sedan in och exekvera, om du tycker att du behöver. (I Matlab 6.5 kan bokstäverna Å, Ä och Ö inte skrivas in på skärmen, inte ens i kommentarer.)

```
>> A=3;                    % A=amplituden
>> f=440, t=2.18;          % f=440 Hz (ettstrukna a)
f =
   440
>> phi=pi/6;               % phi är fasvinkeln enligt Holgersson et al.
>> y=A*sin(2*pi*f*t+phi)  % Varför blir det utskrift på skärmen här?
y =
   2.93444280220093
>> % ***** NYTT FÖRSÖK 2003-01-20-20:10 *****
>> A=3;
>> f=466.2; t=2.18;     % f=466,2 Hz (ettstrukna aiss, liksvävande)
>> % och så vidare ...
```

Ibland måste man skriva in så långa kommandon att kommandoraden inte räcker till. Då kan man bryta raden genom att skriva tre punkter, `...`, följda av vagnretur. Sedan kan man skriva kommandot färdigt på nästa rad. Man kan bryta nästan var man vill. Ett undantag är inuti eller omedelbart efter ett reellt tal. Ett annat undantag är inuti ett variabelnamn (kan dock ibland fungera, men varför skulle man vilja det?). Ett tredje är inne i en kommentar inledd av procenttecken, `%`.

Man kan också i viss mån få mera plats på kommandoraden genom att förstora Matlab-fönstret på bredden.

Övning*

18. Pröva följande kommandon.

```
>> 6*...
3
```

Blir det fel nu i nästa kommando nedan? I så fall varför?

```
>> 6...
*3
```

Fungerar detta?:

```
>> 4*(3+...
9)/6/...
2
```

Och hur fungerar det här?:

```
>> nagonting=...
sin(log(4.8)-sqrt((12+3*sin(45*pi/180))/(pi^2))...
/6)
>> nagonting...
=sin(log(4.8)-sqrt((12+3*sin(45*pi/180))/(pi^2))...
/6)
```

4.4 Matriser

Cleve Moler, matematikern som grundade The Mathworks och skrev programmet MATLAB, sa en gång "MATLAB is a highly typed language. It has one type. The matrix!" Det är inte sant numera eftersom MATLAB har utvecklats väldigt sedan den första versionen började säljas för pengar 1984, men Matlab är en sammandragning av orden "Matrix laboratory" och grunden är just matrisberäkningar.

I det här avsnittet skall vi bara tala om vad en matris är och hur man kan indexera den i MATLAB. Matrisoperationer och kommandon för att enkelt bygga matriser finns det mer om i kap. 3 och 4 i [12].

En matris är en ruta med tal. Man använder hakparenteser i MATLAB för att bilda matriser. Vi börjar med att bilda en 2×3-matris `x`, alltså en matris med 2 rader och 3 kolonner. För att skilja tal på samma rad används mellanslag eller kommatecken. För att ange ny rad används vagnretur eller semikolon. Man bör konsekvent använda semikolon för att markera ny rad, detta framförallt för att undvika problem och misstag i `m-filer`. Lämpligast sätt att skriva in matrisen är alltså följande.

```
>> x = [8 3 4; 6 9 17]

x =
     8     3     4
     6     9    17

>> size(x)

ans =
     2     3
```

Kommandot `size` bestämmer matrisens storlek, `length` är också användbart.

Man kan använda hakparenteser för att bygga matriser av matriser. I följande exempel utnyttjas funktionerna `zeros`, `ones` och `eye` som ger en matris av nollor, en matris av ettor och en enhetsmatris, för att bygga matrisen `z`.

```
>> z = [eye(2) zeros(2,3); ones(2,3) [3 5; 2 7]]

z =
     1     0     0     0     0
     0     1     0     0     0
     1     1     1     3     5
     1     1     1     2     7
```

En *radvektor* är en rad med tal, d v s en matris med bara en rad. En *kolonnvektor* är en kolonn med tal, d v s en matris med bara en kolonn. En vektor är en radvektor eller en kolonnvektor.

Punkt tillsammans med aprostrof `.'` används som transponatoperator, den gör om en matris så att rad 1 blir kolonn 1 i stället, rad 2 blir kolonn 2 och så vidare. En radvektor görs då om till en kolonnvektor eller vice versa. Om man bara använder apostrof `'` så får man både transponering och konjugering av matrisen. Detta har betydelse då man arbetar med matriser som kan innehålla imaginära tal.

Notationen `a:b:c` betyder en radvektor från `a` till `c` i steg om `b`. I notationen `a:c` förutsätts steget vara 1.

```
>> 5:9
ans =
     5     6     7     8     9
>> y = [5:0.5:6].'
y =
    5.0000
    5.5000
    6.0000
```

Övning

19. Prova nu själv att skapa en vektor `x` som startar i 0 och slutar i 10 i steg om 0.05.

Man kan plocka ut ett element ur en matris genom att ange rad och kolonn inom vanliga parenteser. Man kan ta flera rader eller kolonner också genom att indexera med en vektor. Man kan plocka ut hela rader eller kolonner genom ett ensamt kolontecken.

Övning

20. Skriv in en valfri 2×4-matris med namnet `z`. Pröva därefter följande operationer. Försök lista ut resultaten innan du utför kommandona.

    ```
    z(2,1)
    z(1,[1 3])
    z(2,2:4)
    ```

```
z(:,1)
z(2,:)
```

Det fungerar också att ändra en del element i matrisen genom att indexera i tilldelningssatser:

```
>> x(1,3) = 0
x =
     8     3     0
     6     9    17
>> x(1,[1 2]) = [99 98]
x =
    99    98     0
     6     9    17
```

Övning

21. Bilda en 7×5-matris med namn `z` som bara innehåller nollor (använd funktionen `zeros`). Sätt därefter det mittersta elementet och dess 8 närmaste grannar till 1.

4.5 Logisk indexering och relationsoperatorer

I MATLAB kodas falskt som 0 och sant som 1. Relationsoperatorerna i MATLAB är `==` (lika med), `~=` (ej lika med), `<`, `>`, `<=` och `>=`. Logiska operatorer är tex. `&` (logiskt och), `|` (logiskt eller), det finns fler. En lista på operatorer som MATLAB använder kan du komma åt genom att tex. skriva `help >`.

Antag att vi vill simulera 4 stycken slumptal mellan noll och ett och sedan plocka ut de som blev mindre än 0.4. Det verkar vettigt att skriva följande där `rand` är en funktion som väljer slumptal mellan 0 och 1 med lika stor sannolikhet.

```
>> x = rand(1,4)
x =
    0.7975    0.6220    0.1516    0.1031
>> y=x(x<0.4)
y =
    0.1516    0.1031
```

Till vår stora förvåning fungerade det. Men betrakta följande.

```
>> k = x<0.4
k =
     0     0     1     1
```

Resultatet är en vektor av samma längd som `x`. Men den innehåller ettor och nollor, och noll är inte något nummer på ett element i `x`, som ju indexeras från 1 till 4. Anledningen till att det fungerar ändå är att det finns en dold flagga som gör att vektorn minns att den är resultatet från en logisk operation. Funktionen `islogical` avslöjar detta:

```
>> islogical(x)
ans =
     0
```

```
>> islogical(k)
ans =
     1
```

Ett annat exempel är om vi vill ta bort alla element i en vektor `v` som är negativa. Då kan vi indexera med en logisk vektor på vänstersidan i en tilldelningssats och ha en tom matris på högersidan:

```
>> v(v<0) = [];
```

Övning*

22. Skapa en vektor `v` som innehåller både negativa och positiva tal och prova ovanstående. Prova också vad som händer när man skriver `w=v(v<0)`. Hur ska man göra för att plocka bort alla positiva tal i vektorn `v`?

4.6 Operatorerna – och, eller samt inte

Och-operatorn heter `&` och eller-operatorn heter `|`. De fungerar elementvis på vektorer och matriser. `&` har högre prioritet än `|`. Vilket är nytt.

En märklig sak är att de två operatorerna har haft samma prioritet i MATLAB, till skillnad från alla andra programmeringsspråk. MATLAB varnar i version 5 med en lång varningstext om man skriver `a|b&c`, som MATLAB 5 tolkar från vänster till höger som `(a|b)&c`, men som ändrats till `a|(b&c)`, vilket är brukligt i alla andra programmeringsspråk och i matematisk notation.

Icke-operatorn heter `~`. Uttrycket `~x` ger ett för varje element i `x` som är noll, annars ger uttrycket noll. Exempelvis är uttrycket `~~x` samma som uttrycket `x~=0`.

Se även avsnitt 7.3 om `all` och `any`.

4.7 Komplexa tal*

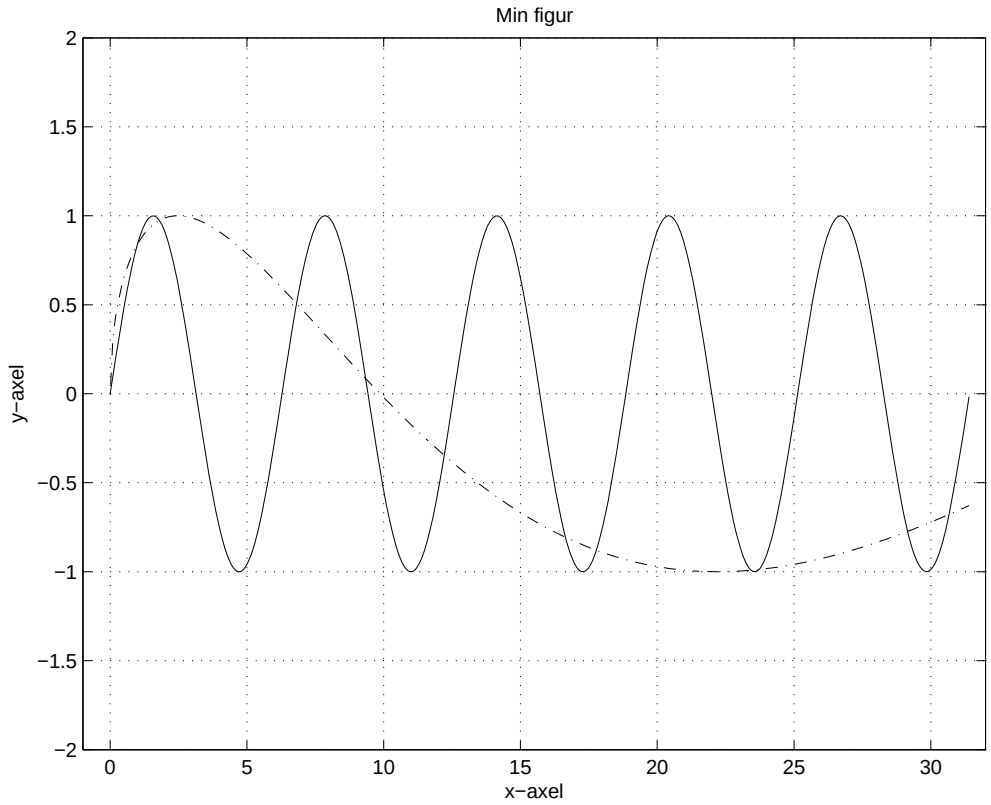
Komplexa tal erbjuder inga problem. Följande exempel illustrerar komplexa tal.

```
>> i = sqrt(-1)
i =
        0 + 1.0000i
>> z = 5 + 7*i
z =
   5.0000 + 7.0000i
>> real(z)
ans =
     5
>> imag(z)
ans =
     7
>> conj(z)
ans =
   5.0000 - 7.0000i
>> q = z*conj(z)
q =
    74
```

4.8 Enkel grafik

MATLAB innehåller numera ett mycket avancerat grafiksystem, men här skall vi bara gå igenom det allra enklaste. Vi börjar med ett exempel. Figur 1 är skapad på följande sätt. Först ritas den heldragna kurvan:

```
>> figure  % för att skapa ett nytt figurfönster
>> x = 0:0.1:2000*pi;
>> f = sin(x);
>> plot(x,f)
```



Figur 1: Två kurvor

Det som hände var att vektorn `x` skapades som en vektor från 0 till 2000π i steg om 0.1. Man kan också använda funktionen `linspace` för samma ändamål. När man skall plotta en kurva så gäller det att ta lagom många punkter, inte för få så att det blir hackigt, och inte för många så att det tar för mycket minnesutrymme och tid. I vektorn `f` lagras sedan sinus av motsvarande element i `x`. Funktionen `plot` ritar en kurva.

I nästa steg ritas den punktstreckade kurvan:

```
>> g = sin(sqrt(x));
>> hold on
>> plot(x,g,'-.')
```

Funktionen `hold` används för att inte den heldragna kurvan skall försvinna när vi ritar den streckade. Funktionen `plot` ges ett tredje argument, teckensträngen `'-.'`,

som talar om hur kurvan skall ritas, här punktstreckad. Andra möjligheter är att rita med olika färger och olika plotsymboler, t ex optionen `'o'` för små ringar, i stället för en kurva (se hjälpfunktionen till `plot`).

```
>> axis([-1 32 -2 2])
>> grid on
```

Axlarna justerades, med `axis`, och hjälplinjer lades till, med `grid`.

```
>> xlabel('x-axel')
>> ylabel('y-axel')
>> title('Min bild')
```

Man skall ha texter på axlarna och figuren skall ha en titel, det ordnades med `xlabel`, `ylabel` och `titel`. Nu är bilden klar. Prova nu följande kommandon, (fungerar dock bara om det finns et ljudkort installerat);

```
>> sound(f);
>> sound(g);
>> sound(f.*g); % punkten . är till för elementvis multiplikation!
>> sound(f+g);
```

Övningar

23. Hur ser ljuden `f.*g` samt `f+g` ut? Kolla själv genom att komplettera figuren.

24. Hur ändrar man frekvensen på ljudet? Prova också att lägga till lite fler toner.

4.8.1 Peka och zooma

Man kan peka med musen i en bild och få koordinaterna som svar, genom att använda funktionen `ginput`. Vidare kan man också lägga till text på något lämpligt ställe i bilden, använd `gtext`.

Övning

25. Rita kurvan $\sin(x) \cdot x$ med följande kommandon.

```
x = 0:0.01:pi; y = sin(x).*x; plot(x,y)
```

Observera punkten . framför *. Den talar om för MATLAB att vi ska multiplicera ihop talen i de två vektorerna elementvis med varandra, och inte utföra en vanlig matrismultiplikation. Klicka någonstans i bilden efter att du skrivit följande kommando.

`>> p = ginput(1)` Du fick x- och y-koordinaten för punkten du pekade på. I anropet av funktionen `ginput(1)`användes 1 som värde av inparametern. Vad händer om du använder något annat heltal tex. 3 istället?

Lägg nu till text genom att först skriva kommandot

```
>> gtext('Be in love with yr life, cite J. K.')
```

och därefter klicka någonstans i bilden. Man kan också zooma i en figur genom kommandot `zoom`, eller med hjälp av verktygen i figurfönstret.

Övning

26. Ge kommandot `zoom`. Klicka därefter med musen och håll ner musknappen och dra den en bit och släpp knappen sedan. Se till att få med något intressant i rutan som syns. Pröva att zooma ut med högerknappen också.

4.8.2 Subplot

Det är lätt att plotta flera olika grafer i figurfönstret. Man använder funktionen `subplot(m,n,p)` som delar in figurfönstret i `m`×`n` grafer och sätter aktuell graf till nummer `p`. Man kan strunta i kommatecknet också och skriva t ex `subplot(313)` om man vill dela in figurfönstret i tre avlånga grafer och rita i den nedersta.

Övning

27. Pröva följande kommandon och försök lista ut vad som händer i förväg.

```
>> x = 0:50;
>> y = sqrt(x);
>> z = rand(1,51);
>> subplot(211)
>> plot(x,y)
>> subplot(223)
>> plot(x,z,'o')
>> subplot(224)
>> plot(x,y+z,'.-r')
>> grid on
>> grid off
```

Man kan ha flera figurfönster igång samtidigt också, se hjälptexten till `figure`.

4.8.3 Papperskopior

Det finns två sätt att skriva ut en figur direkt till skrivaren. Det ena är kommandot `print` och det andra är att klicka med musen på menyn "File" i ovankanten av figurfönstret, och sedan klicka på "Print".

4.8.4 Att spara undan och skriva ut figurer

Om man har plottat diagram i ett figurfönster, kan man med kommandot `saveas` spara hela figurfönstret i en extern fil, vilken får extensionen `.fig`. Skriver man exempelvis in

```
>> saveas(2,'myplot1','fig')
```

på skärmen, sparar man undan fönster 2 (`figure(2)`) i en fil `myplot1.fig`. Man återkallar en `fig`-fil med kommandot `open`:

```
>> open myplot1.fig
```

eller

```
>> open('myplot1.fig')
```

När man återkallar en `fig`-fil med `open`, öppnas ett nytt fönster. På så sätt undviker man att ett befintligt figurfönster överritas av misstag.

Med kommandot `print` kan man spara undan Matlab-figurer i filer av många olika format, se `help print`. Exempelvis sparar kommandot

```
>> print -f3 -deps mynewplot
```

undan figurfönster 3 (`figure(3)`) på EPS-format (Encapsulated PostScript) i en fil `mynewplot.eps`. Om man skriver

```
>> print -f2 -dmeta
```

under Windows, kopieras figuren till klippytan i Windows och kan klistras in exempelvis i ett MSWord-dokument. Om man i Matlab skriver

```
>> print -f4
```

skrivs figurfönster 4 (`figure(4)`) ut omedelbart på närmaste skrivare. Ibland händer det att datorn tillfälligt har ”glömt” vilken som är den närmaste skrivaren. Då måste man gå vägen över en utskriftshanterare och ange en skrivare. Hur får man fram en utskriftshanterare? Jo, man måste välja File → Print i figurfönstrets menybar eller trycka på den knapp i samma menyrad som bär en stiliserad bild av en skrivare. Då öppnas ett nytt fönster (utskrifts-hanteraren), i vilket man specificerar skrivaren: tex. The printernamn.

4.8.5 Spara figur som m-fil

Med kommandot `saveas(gcf,'figurnamn','m')` kan man spara en aktuell figur som en `m-fil`. Detta är speciellt bra om man håller på att arbeta med lite mer komplicerade utskrifter och figurer. Den stora fördelen med detta sättet att spara en figur är att man från Matlab kan öppna den som vilken `m-fil` som helst och därefter fortsätta att arbeta i den. Skriv

```
>> figurnamn
```

så får man tillbaks figuren.

4.9 Teckensträngar*

Teckensträngar anges inom apostroftecken, alltså `'`, och inte inom citattecken `''` som i andra programspråk:

```
>> s = 'kålrot';
```

Variabeln `s` är nu i själva verket en radvektor bestående av sex tal, koderna för 'k', 'å' o s v, men variabeln vet själv att den skall skrivas ut som bokstäver. För att se koderna kan man bara addera 0.

```
>> s
s =
kålrot
>> s+0
ans =
   107   229   108   114   111   116
```

Eftersom strängar bara är vanliga radvektorer så kan man bygga längre strängar med hakparenteser precis som man kan göra med numeriska vektorer:

```
>> t = ['En god ', s, 'ssoppa']
t =
En god kålrotssoppa
```

4.10 Övningsuppgifter

28. Skapa en vektor `x` som antar alla heltalsvärden mellan -10 och 10.

    Tips: Använd syntaxen `start:steg:slut` eller

    funktionen `linspace(start,slut,antalpunkter)`.

29. Beräkna värdet av funktionerna $y = x^2$, $z = (x-2)^2$ och $w = x^2 + 20$ för ovanstående x-värden.

30. Rita graferna av funktionerna y, z och w i samma figur men med olika färger. Tips: använd `plot` samt `hold`.

31. Bestäm grafiskt om följande funktioner är jämna eller udda

$$y = \sin(x)$$

$$y = \exp(-x^2)$$

32. Rita följande funktion $f(x)$ och dess inversa funktion $f^{-1}(\cdot)$ i samma figur

$$f(x) = \frac{x+1}{2}$$

33. Undersök grafiskt följande gränsvärden

$$\frac{\sin(x)}{x}, x \to 0$$

$$x \cdot \log(3 \cdot x), \to 0+$$

$$(1 + 1/n)^n, n \to \infty$$

34. Plotta $\sum(2/3)^k$ för $k = 1 \ldots 500$.

    Använd `axis` för att visa vad som sker tydligare.

    Vad blir seriens summa när $k \to \infty$?

35. Bestäm så noggrant som möjligt största och minsta värde av funktionen $f(x) = x \cdot \exp(-x)$ där x ligger i det slutna intervallet $[0, 2]$. Ange även x-koordinaterna för dessa värden. Tips: Använd funktionerna `min`, `max` och `find`, eller `fmin`.

36. Plotta följande funktioner i samma figur

$$x = \cos(t)$$

$$y = \sin(t)$$

$$z = t$$

37. Plotta den sammansatta funktionen $w = \sin(x) \cdot \cos(y)$.

38. Bestäm skärningen grafiskt för linjerna, (plotta de två linjerna i samma figur)

$$L_1 : \begin{cases} x &= 2 + 4t \\ y &= 4 - 8t \end{cases}$$

och

$$L_2 : \; 2x + y \;\; = \;\; 0.$$

5 Spara och läsa in

Man sparar alla sina variabler med kommandot `save`. Då skapas en fil som heter `matlab.mat` (om det finns en tidigare sparad fil med samma namn så skrivs den över). För att få tillbaka variablerna använder man `load`. Man kan också ange ett explicit filnamn, t ex `save minfil3` Då lägger MATLAB automatiskt till fyra tecken så att filens namn blir `minfil3.mat` Om man bara vill spara vissa variabler i filen så anger man dem efter filnamnet, exempelvis `save minfil5 a x`

5.1 Aktuell katalog

Var hamnar filen i filträdet? Jo, i MATLAB's aktuella katalog. Med kommandot `cd` med ett argument byter man aktuell katalog. Om man använder `cd` utan argument får man reda på aktuell katalog (observera här skillnaden mot Unix-kommandot `cd`). Kommandot `pwd` ger också aktuell katalog. Med kommandot `dir` eller `ls` kan man se vilka filer som finns. Tre av dessa kommandon heter likadant som i Unix och det fjärde heter likadant som i Ms DOS, vilket gör att man lättare kommer ihåg dem (På Unix-maskiner skiljer sig `dir` och `ls`, prova båda). Det finns till och med ett kommando `mkdir` som gör samma sak som i Unix, det vill säga skapar en ny katalog, även under MS Windows. I Matlab finns bland annat följande kommandon till hjälp

```
>> cd F:\fms065  % Gör 'F:\fms065' till aktuell katalog (om den finns)
>> cd ..         % Gör faderkatalogen till aktuell katalog
>> cd            % Ger namnet på aktuell katalog
>> dir           % Listar filerna i aktuell katalog
>> delete myfile % Tar bort filen 'myfile'
>> type myfile   % Skriver ut textfilen 'myfile' på skärmen
>> mkdir exjobb  % Skapar en underkatalog vid namn 'exjobb'
>> !rmdir fms065 % Tar bort en underkatalog vid namn 'fms065'
```

Övning

39. Ta reda på vad aktuell katalog heter. Skapa en underkatalog med namnet `matlab` i din hemkatalog. Gör så att den nyss skapade katalogen blir aktuell katalog. Titta efter vilka variabler som finns i arbetsminnet. Spara variablerna. Titta efter att en fil har skapats. Radera alla variabler och titta efter att arbetsminnet är tomt. Ladda tillbaka variablerna. Titta efter att de finns igen.

Man kan också spara en matris med läsbara siffror. Det gör man genom att ange filnamn, variabelnamn och optionen `-ascii -double` när man sparar. Betrakta följande exempel.

```
>> a = [1 2; 3 4]
a =
     1     2
     3     4
>> save afil.txt a -ascii -double
>> type afil.txt
   1.0000000e+00   2.0000000e+00
   3.0000000e+00   4.0000000e+00
>> clear all
>> load afil.txt
>> who
Your variables are:
a         afil
>> afil
afil =
     1     2
     3     4
```

MATLAB-kommandot `type` skriver ut en fil i kommandofönstret (det motsvarar alltså Unix-kommandona `cat` och `more`). För att inte långa texter bara skall svischa förbi kan man använda kommandot `more on`.

Notera att variabelnamnet efter inladdning av filen blev samma som filnamnet (med undantag av `.txt`).

På det viset kan man enkelt exportera och importera siffror till och från egna program, skrivna i till exempel Java.

Om man skall mata in många siffror för hand gör man det bäst i en text-fil med sin favoriteditor för att sedan läsa in den i MATLAB på det beskrivna sättet.

5.2 Excel-filer

Om man har data i Excel och sparat dem separerade med hjälp av `tab` kan de exporteras till MATLAB. Exempelvis kan ett datamaterial sparat som `Box1.txt` öppnas i MATLAB genom, kommandot `load Box1.txt -ascii`. När man är i MATLAB's kommandofönster refererar man till datamaterialet med `Box1`. För att detta ska fungera måste alla elementen i filen vara numeriska, alla rader måste innehålla lika många element och decimaltecknet måste följa rätt konvention, i allmänhet punkt inte kommatecken.

5.3 Loggbok

Om du vill kunna spara vad du har gjort under en MATLAB-session kan du ge kommandot `diary` då skapas en loggfil. Egenskaperna hos denna beskrivs i avsnitt 2.7 sist i *Användarhandledningen för MATLAB 6.5*, [12]. Denna loggfil kan du sedan använda, dels för egen del för att i efterhand kunna se vad du egentligen gjort, t.ex. i samband med felsökning, eller för att kunna verifiera vad du gjort när du skall redovisa detta för någon annan. Det finns dock ett problem med loggfilen, den sparar allt även utskriften av stora matriser. Detta gör att den kan bli ohanterligt stor.

6 Mer om matrisoperationer

Det finns tre MATLAB-operatorer som betecknas `*`, / och `^`. Operatorerna är så kallade matrisoperatorer. Det finns tre andra operatorer som har en punkt först och som kallas elementvisa operatorer. De är `.*`, `./` och `.^`

6.1 Multiplikation

Vi skapar två matriser för att illustrera skillnaden mellan matrismultiplikation och elementvis multiplikation.

```
>> A = [1 3 6; 2 3 8]
A =
     1     3     6
     2     3     8
>> B = [1 5; 4 2; 6 3]
B =
     1     5
     4     2
     6     3
```

Matrismultiplikation är inte symmetrisk, `A*B` är inte samma sak som `B*A`:

```
>> A*B
ans =
    49    29
    62    40
>> B*A
ans =
    11    18    46
     8    18    40
    12    27    60
```

Om man vill multiplicera varje element i matrisen `A` med motsvarande element i matrisen `B'` använder man operatorn `.*` så här:

```
>> A.*B'
ans =
     1    12    36
    10     6    24
```

För exponentoperatorn `^` gäller samma sak. Notationen `A^2` betyder samma sak som `A*A`, medan `A.^2` betyder att varje element i `A` skall kvadreras.

Det finns också en operator `.'` som behövs när man arbetar med komplexa matriser. Operatorn `'` betyder Hermitskt transponat, dvs. transponat av komplexkonjugatet, medan operatorn `.'` betyder transponat. Men för reella matriser så är det ingen skillnad.

6.2 Invers och division

Låt oss illustrera de olika divisionsoperatorerna med de tre matriserna A, B och C:

```
>> A = [1 2; 5 3]
A =
     1     2
     5     3
>> B = [5 2;3 1]
B =
     5     2
     3     1
>> C = A*B
C =
    11     4
    34    13
```

Invers heter i MATLAB `inv` och enhetsmatrisen heter `eye`. Om man vet att relationen $AB = C$ gäller och man känner C och B men inte A så kan man multiplicera med B^{-1} från höger för att få fram $A = CB^{-1}$, eller i matlabformulering:

```
>> C*inv(B)
ans =
    1.0000    2.0000
    5.0000    3.0000
```

Man kan också göra på följande sätt, vilket är bättre av numeriska skäl. Minnesregeln är att om $AB = C$ så borde ju $A = C/B$. Så får man inte skriva i matematisk text men i MATLAB bör man av beräkningstekniska skäl skriva precis så, (tolkas som C dividerat med B från höger):

```
>> C/B
ans =
    1.0000    2.0000
    5.0000    3.0000
```

Om C och A vore kända men B okänd så kan vi multiplicera relationen $AB = C$ med A^{-1} från vänster och få fram att $B = A^{-1}C$.

```
>> inv(A)*C
ans =
    5.0000    2.0000
    3.0000    1.0000
```

Men ännu hellre gör man som nyss. Eftersom matrismultiplikation inte är symmetrisk så behövs en ny operator \ för vänsterdivision. Minnesregeln är att om $AB = C$ så borde ju $B = A\backslash C$, som tolkas som C dividerat med A från vänster:

```
>> A\C
ans =
    5.0000    2.0000
    3.0000    1.0000
```

Att siffrorna presenteras med ett antal nollor beror på att avrundning under beräkningarna har gjort att den sista signifikanta siffran inte är noll. I MATLAB görs alla beräkningar med dubbel precision, vilket betyder att varje tal tar åtta byte av minnet och att antalet signifikanta siffror ungefär är femton. (För bildbehandling har man ofta behov av andra format, se `help datatypes`.)

6.3 MK-metoden

Antag att vi har ett ekvationsystem $XB = Y$ där det finns fler rader än kolonner i matrisen X (och ingen kolonn i X kan skrivas som en linjärkombination av de övriga). Då saknar ekvationssystemet i allmänhet lösning (om inte vektorn Y är så speciell att ekvationssystemet ändå går att lösa).

Istället vill man ofta ha minstakvadratlösningen, det B som minimerar avståndet mellan Y och XB (med avstånd menas här normen $||Y - XB||$). Lösningen ges av de så kallade normalekvationerna, vilka i matrisformulering fås genom att multiplicera på X^T från vänster, $X^T XB = X^T Y$. Lösningen blir $B = (X^T X)^{-1} X^T Y$. I MATLAB skulle koden för detta bli kort, men man skall inte skriva så. MATLAB's matrisdivisionsoperator levererar i stället minstakvadratlösningen direkt på ett numeriskt bättre sätt:

```
>> X = [1 2; 3 4; 5 0]}
X =
     1     2
     3     4
     5     0
>> Y = [0 1 2]'
Y =
     0
     1
     2
>> B = X\Y
B =
    0.4048
   -0.0833
>> Yhatt = X*B
Yhatt =
    0.2381
    0.8810
    2.0238
```

Observera att det var till vänster X stod så det var vänsterdivision \ som användes. Jämför också `Yhatt` med `Y`, kommentar?

Notationen i MATLAB att använda matrisdivisionsoperatorerna för både vanlig lösning och minstakvadratlösning är helt konsekvent eftersom den vanliga lösningen är ett specialfall av minstakvadratlösningen, när matrisen X är kvadratisk med full rang.

Övning

40. Anpassa en rak linje till följande punkter, $p_i(x, y)$.

    $p_1 = (0, 1)$, $p_2 = (1, 1)$, $p_3 = (2, 3)$, och $p_4 = (4, 5)$

    Börja med att plotta punkterna i en figur och låt ögat anpassa en rak linje till datamaterialet, ('fitted by eye'). Utnyttja t.ex. funktionen `ginput` för att lägga in en linje. Välj därefter ut punkterna parvis och beräkna a och b i ekvationen $y = a + b \cdot x$. Använd därefter samtliga punkter och bestäm värdena på a och b med hjälp av MK-metoden. Tips matrisformulering och \-operatorn är allt som behövs för beräkningarna. Plotta dina olika räta linjer i punktsvärmen.

Hur ska vi tolka MK-skattningen av linjen? Går t.ex.MK-skattningen genom punkten $p_4 = (4, 5)$? Beräkna kvadratsumman av det lodräta avståndet mellan de olika linjerna och punkterna. Vilken blir minst? Går det att hitta en linje där kvadratsumman är mindre?

6.4 Skalärprodukt och norm*

Skalärprodukten $u^T v$ mellan två kolonnvektorer, eller inre produkten som den ibland kallas, skrivs precis som väntat i MATLAB. Vektorprodukten (kryssprodukten)får man som $cross(u, v)$. För normen $||u|| = \sqrt{u^T u}$ finns funktionen `norm`.

```
>> u = [4 3 7]'
u =
     4
     3
     7
>> v = [5 2 1]'
v =
     5
     2
     1
>> u'*v
ans =
    33
>> u*v'
ans =
    20     8     4
    15     6     3
    35    14     7
>> norm(u)
ans =
    8.6023
```

Determinanten beräknas med `det` och rangen av en matris med `rank`. För egenvärdena finns funktionen `eig` och funktionen `diag` tar ut diagonalelementen ur en matris. Se i följande exempel hur man kan få ut fler än ett utargument från en funktion i MATLAB:

```
>> x = rand(3)
x =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
>> [v, e] = eig(x)
v =
    0.7510    0.8135   -0.3483
    0.6246   -0.5268   -0.6320
    0.2142   -0.2464    0.6922
e =
    1.5996         0         0
         0    0.2046         0
```

```
         0         0    0.7119
>> v*e*inv(v)
ans =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
>> diag(e)
ans =
    1.5996
    0.2046
    0.7119
```

Ordningen mellan egenvärdena blev inte i storleksordning, tyvärr, och vi hade tur som fick reella egenvärden. De hade kunnat bli komplexa eftersom matrisen med slumptal inte är symmetrisk.

6.5 Övningsuppgifter*

41. Beräkna skalärprodukten av $x = (1, 2)$ och $y = (3, -2)$
    Rita även vektorerna i en lämplig figur.

42. Beräkna skalärprodukten och vektorprodukten av $u = (1, 0, 1)$ och $v = (1, 2, 3)$
    Rita även vektorerna och vektorprodukten i en lämplig figur.

43. Lös följande ekvationssytem

$$1x - 2y + 1z = 1$$
$$2x - 6y + 6z = 2$$
$$-3x + 5y + 1z = 3$$

44. Lös ekvationssystemet

$$1x - 2y + 1z = 1$$
$$2x - 6y + 6z = 2$$
$$-3x + 5y - 1z = 3$$

    Vad hände här? Vad innebär detta för matrisen A?

6.6 Diverse: Sortera, summera, medelvärdesbilda, söka maximum och sådant

För radmatriser och kolonnmatriser (vektorer) finns några särskilda funktioner:

A. `length(v)` ger antalet element i vektorn `v`; funktionen påminner alltså om `size`.

B. `sum(v)` ger summan av elementen i vektorn `v`.

C. `prod(v)` ger produkten av elementen i vektorn `v`.

D. `max(v)` ger det största av elementen i vektorn `v`.

E. `min(v)` ger det minsta av elementen i vektorn `v`.

F. `sort(v)` sorterar elementen i vektorn `v` i stigande ordning (det minsta värdet först).

G. `norm(v)` ger euklidiska normen av vektorn `v`.

H. `mean(v)` ger aritmetiska medelvärdet av elementen i vektorn `v`.

I. `std(v,1)` ger stickprovsstandardavvikelsen för elementen i vektorn `v`.

J. `std(v)` ger en skattning av populationsstandardavvikelsen för elementen i vektorn `v`.

Övningar

45. Beräkna summan av alla positiva, udda heltal lägre än 100.

46. Generera 50 rektangelfördelade slumptal mellan 0 och 1. Ange det största och det minsta värdet bland dem. (Utnyttja funktionen `rand`.)

47. Generera 50 standard-normalfördelade slumptal. Ange 1) det största värdet, 2) det minsta värdet, 3) variationsbredden (avståndet mellan det största och det minsta värdet). (Utnyttja funktionen `normrnd` för att generera slumptalen.)

48. Sortera slumptalen i uppgift 47 i stigande ordning och i fallande ordning.

49. Matlab-funktionerna `length`, `sum`, `prod`, `max`, `min`, `sort`, `mean` och `std` fungerar faktiskt även om argumentet är en matris. Tag reda på vad det betyder.

7 Kort om programmering i MATLAB

De reserverade orden `if`, `else`, `while`, `for`, `end`, `break`, `switch`, `case` och `otherwise` styr programflödet. MATLAB har inte något reseverat ord `begin` som man har i till exempel Pascal. Till varje `while`, `if` och `for` hör dock ett `end` som kommer senare.

7.1 Repetitionssatser (for-satser, while-satser)

Ibland måste man upprepa en kommandosekvens ett förutbestämt antal gånger. Då används en for-sats.
Exempel. Låt $x_1 = 0$, $x_2 = \cos(x_1)$, $x_3 = \cos(x_2)$, $x_4 = \cos(x_3)$, ..., $x_{100} = \cos(x_{99})$ vara en serie om 100 element. Generera den i Matlab. Lösning (skriv in och exekvera!):

```
>> x=zeros(100,1);
>> for i=2:100, x(i)=cos(x(i-1)); end
>> x
```

Plotta gärna resultatet x mot ordningsnumret. Konvergens? Mot vad?

Ibland måste man däremot upprepa en kommandosekvens tills ett visst bestämt villkor är uppfyllt, alltså ett obestämt antal gånger. Då används en while-sats.
Exempel. Följande kommandosekvens finner lösningen till ekvationen $x = \cos(x)$ med fixpunktsmetoden.

```
>> xOld=0; xNew=cos(xOld); i=1;  % i är antalet funktionsberäkningar
>> while abs(xNew-xOld)>eps*abs(xNew); xOld=xNew; xNew=cos(xNew); i=i+1, end
>> xNew
```

Skriv in och exekvera! Vi har låtit variabeln i skrivas ut på skärmen efter hand. Hur många iterationer krävdes?

7.2 Villkorssatser (if-satser)

Det händer att man måste tillgripa villkorssatser. En villkorssats är av typen

```
>> if logiskt-uttryck, satser, end
```

eller

```
>> if logiskt-uttryck, satser, else, satser, end
```

eller

```
>> if logiskt-uttryck-1, satser
>> elseif logiskt-uttryck-2, satser
>> else, satser
>> end
```

Exempel. Följande kommandosekvens läser först in ett tal från kommandoraden. Sedan skriver den ut

```
Haha, talet var negativt!
```

på skärmen om talet var strikt negativt, annars skrivs kvadratroten av talet ut.

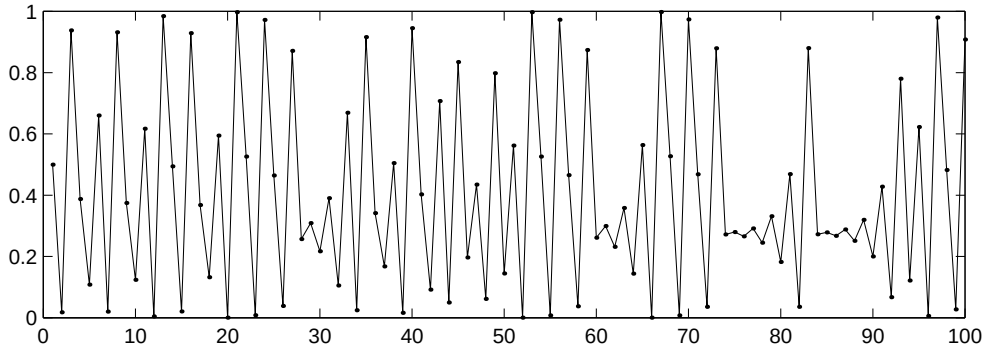
```
>> x=input('Ange ett tal här: ');
>> if x<0, disp('Haha, talet var negativt!'), else, sqrt(x), end
```

Följande är ett exempel på en kombinerad `while`- och `if`-konstruktion.

```
>>   k = 3;
>>   while k>1
>>      if rem(k,2) == 0
>>         k = k/2;
>>      else
>>         k = 3*k+1;
>>      end
>>      disp(k)
>>   end
    10
     5
    16
     8
     4
     2
     1
```

Ett så långt exempel som det ovan skrivs egentligen bäst i en fil med en editor, men ibland är det bra att skriva korta programslingor direkt i kommandofönstret. För att kunna editera lätt skriver man då flera satser på samma rad, som separeras med kommatecken eller semikolon. Följande kommandon genererade figur 2.

```
>> n = 100;
>> a = 1.73;
>> x = 0.5;
>> for i=2:n, x(i)=(1-a*x(i-1))^2; end
>> subplot(211)
>> plot(x,'.-')
```



Figur 2: Kaos

Man kan bryta en `for` eller `while`-slinga med `break` precis som i Java eller C. Däremot saknas konstigt nog motsvarande `continue`-instruktion.

En flervägs hoppinstruktion finns också, se hjälptexten till `switch`.

7.3 Ingen och alla

I en `if`- eller `while`-konstruktion vill man ofta testa om alla (eller minst ett) element i en vektor har en viss egenskap. Använd då `all` och `any`.

```
>> all(x<100)
ans =
     1
```

Ja, tydligen var alla element i vektorn `x` mindre än 100.

För en matris med minst två rader fungerar de två funktionerna på varje kolonn för sig och lämnar en radvektor som svar. Om man vill testa om det finns något element i en matris `A` som är negativt så skriver man `any(any(A<0))`.

Om villkoret i en `if`- eller `while`-sats har någon annan storlek än en 1×1-matris, dvs en skalär, så kan man fråga sig vad som händer. Svaret är att villkoret betraktas som sant bara om alla elementen är icke-nollor. Man kan alltså testa om två matriser är lika genom att skriva så här:

```
>> if A==B, disp('Yes equal'), end
```

Däremot kan man inte använda

```
>> if A~=B, disp('Fooled! Ha ha!'), end
```

för att testa att de två matriserna inte är lika, eftersom testet som utförs kräver att *alla* element är olika för att skriva ut teckensträngen. Man menade nog

```
>> if any(any(A~=B)), disp('Not equal'), end
```

eller ekvivalent

```
>> if A==B, else, disp('Not equal'), end
```

8 Egna Matlab program – m-filer

MATLAB-program är filer med MATLAB-kod. Namnet skall sluta med punkt och bokstaven m, t ex `mittprog.m`, och de kallas därför `m-filer`. Det finns två varianter av `m-filer`: `script` och `function`.

`script`-filer är helt enkelt MATLAB-kommandon som utförs som om man hade skrivit dem direkt i kommandofönstret.

`function`'s har in- och utargument och de börjar alltid med det reserverade ordet `function`. Möjligheten till in- och utargument ger en bättre struktur och användbarhet än de enkla `script`-filerna. Betrakta följande egna funktion som finns i en fil med namnet `randint.m`.

```
function x = randint(m,n,a,b)
%RANDINT Random integer matrix.
%
%        randint(m,n,a,b)
%
%        Returns an m-by-n matrix with
%        entries between a and b (default
%        is 0 and 9).
if nargin==0, m=1; end
if nargin<2, n=m; end
if nargin<3, a=0; b=9; end
x = floor((b-a+1).*rand(m,n)) + a;
```

Funktionen `floor` avrundar nedåt, till närmast lägre heltal. Några kommentarer om koden i funktionen följer.

- Variabeln `nargin` ger antalet inargument vid anropet. Det kan vara färre än antalet deklarerade argument, men inte fler.
- Koden visar hur man använder `nargin` för att ge värden till inargument som anroparen inte har angivit. Programmet ovan är skrivet för 0, 1, 2 eller 4 argument.
- Det finns ingen instruktion som måste avsluta funktionen, men man kan använda det reserverade ordet `return` om man vill avsluta på något speciellt ställe i koden.
- Utargumentet, `x` i exemplet, måste tilldelas ett värde.
- Hjälptexten till funktionen – den text som visas med kommandot `help` – är det första blocket med kommentartext. Som kommentartext räknas allt som följer efter ett procenttecken på varje rad. Om man börjar hjälptexten med att ta en kopia av första raden i `m-filen` får man automatiskt reda på hur anropet av funktionen skall se ut när man skriver `help funktionsnamn`.
- Funktionsnamnet bör stämma överens med filnamnet – i själva verket bryr sig MATLAB bara om filnamnet men man skall ju inte förvirra sig själv eller andra med olika namn.

Ett av de bästa sätten att lära sig hur man skriver bra matlabprogram är att studera kod som finns. Kommandot `type` skriver ut hela programtexten på skärmen. En annan fördel med att studera programtexten är det ibland är först då man förstår hur och `vad` som programmet beräknar. Tycker man att resultaten inte alltid stämmer ska man inte dra sig för att bena upp hur programmet räknar.

- Övningsuppgift: Skriv ut hjälptexten till funktionen `mean` på skärmen. Skriv sedan ut hela programkoden.

För att editera programkod använder man sin favoriteditor. Det finns också en inbyggd editor i MATLAB som startas med kommandot `edit minfil.m`, men i Unix-miljö är den så usel så att alla använder Emacs eller någon annan editor istället. Det finns också styrfiler för matlabkod till Emacs att tillgå som gör automatisk indragning och använder tjusiga färger för teckensträngar, kommentarer och liknande.

8.1 Varning och fel

När man skriver snygga program som andra skall använda så skall man kontrollera indata och ge varningar eller avbryta om något är orimligt. Man använder då funktionerna `warning` för att varna och `error` för att avbryta, på exempelvis det här sättet:

```
if any(any(x<0))
   error('Illegal x')
end
```

8.2 Sökvägar

Hur kör man programmet då? Finns m-filen i aktuell katalog så är det bara att köra. Skriv kommandot som om det vore inbyggt i MATLAB, i det här fallet `randint`. I de flesta andra programmeringsmiljöer måste man kompilera eller ladda in eller göra något speciellt men inte i MATLAB. Varje gång man ger ett kommando så kollar MATLAB med diverse sofistikerade metoder att programkoden inte har ändrats. Det kan dock hända om man har otur att MATLAB har en gammal kopia i minnet och inte upptäcker att man har ändrat i programfilen. MATLAB kan till och med köra en gammal version samtidigt som den glatt skriver ut den nya koden när man använder kommandot `type`. För att vara bombsäker begär man då `clear functions` så laddar den in *alla* programfiler på nytt när man kör dem.

Om man vill att MATLAB skall köra program från flera olika underkataloger använder man funktionerna `path` och `addpath` för att ta reda på och sätta alla sökvägar.

8.3 Övningsuppgifter

50. Skriv en `m-fil` vars första rader ser ut på följande sätt

    ```
    function T = plotta(a,b)
    % T = plotta(a,b)
    % Plottar sinus och cosinus i intervallet [a,b] samt
    % lagrar funktionsvärdena i matrisen T.
    ```

51. Gamla kvadratrötter.
    Använd följande mycket gamla metod för att implementera en kvadratrotslösare i form av en `m-fil`. Syntaxen i m-filen ska vara `y=minsqrt(x,ystart,precision)`. Algoritmen är klar när skillnaden mellan två konsekutiva värden är mindre än `precision`. Låt y beteckna kvadratrotten av x, alltså $y^2 = x$. Om y_0 är en gissning av kvadratroten då är medelvärdet av y_0 och x/y_0 en bättre gissning.

    $$y_1 = \frac{y_0 + x/y_0}{2}$$

8.4 Egna funktioner

Ibland behöver man använda en funktion som inte finns definierad i MATLAB. Detta problem löser man genom att definiera en egen funktion i form av en `m`-fil med lämpligt namn. Funktionen $sincos(x) = \sin(x) \cdot \cos(x)$ är ett sådant exempel. Vi definierar funktionen med namnet `sincos` och lagrar den på `m`-filen `sincos.m`.

```
function y = sincos(x)
     y=sin(x).*cos(x);
```

Ett anrop med svar av funktionen kan se ut på följande sätt:

```
>> y1=sincos([0 pi/4 pi/2 2 pi])
   y1=
       0 0.5000 0.000 -0.3748 -0.0000
```

Lös nu följande övningsuppgifter:

52. Skriv en egen funktion `exp2(x)` som beräknar $e^{(x^2)}$.

53. Skriv en egen funktion `expexp(x)` som beräknar $e^{-e^{-(x-b)/a}}$.

54. Skriv en egen funktion `h(x)` som beräknar `1-expexp(x)`. Där `expexp(x)` är definierad i föregående övningsuppgift.

9 Kontrollfrågor

Några kontrollfrågor för att testa dig själv

55. Beräkna skalärprodukten av vektorerna $(1, 2, ..., 100)$ och $(1, 1, ..., 1)$.

56. Skapa en vektor, Z, som innehåller 1000 slumpvis valda heltal. Där varje heltal kan anta ett värde mellan 0 och 1000. Leta upp det största och minsta heltalet samt deras position i vektorn Z. Beräkna också summan av alla heltalen som är mindre än 15.

57. Rita följande två funktioner i samma graf

$$x(t) = \cos(\frac{-11}{4} \cdot t) + 7\cos(t)$$

$$x(t) = \sin(\frac{-11}{4} \cdot t) + 7\sin(t)$$

    där $0 < t < 8\pi$.

58. Hur inverterar man en matris i MATLAB?

- Lös ekvationssystemet `A*x=b`, där `A=[6 7;4 9]` och `b=[6;7]`. Kontrollera sedan lösningen genom att beräkna `A*x`.

59. Plotta sinusfunktionen i intervallet $[0, 2\pi]$ med en **röd** kurva. Plotta därefter cosinusfunktionen i samma intervall med en **blå** kurva i en egen figur. Avsluta med att plotta de två funktionerna med var sin färg i samma figur. Tips: använd kommandot `hold` och utnyttja `subplot` funktionen på lämpligt sätt så kan du få de olika delfigurena i samma figurfönster.

60. Skriv en funktion, (`m-fil`), som beräknar det n'te Fibonacci talet $fib(n)$ som definieras som

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n) &= fib(n-1) + fib(n-2). \end{aligned}$$

61. Skapa en slumpvis sekvens av bokstäver med följande kommando:

    ```
    >> s=setstr(floor(26*rand(1,20000)+65));
    ```

    Hur många gånger erhölls bokstaven A och P respektive teckensträngen AP och PA? Hur många gånger borde de uppträtt?

10 Inför kursen

10.1 M-fil – en egen rutin för ett normerat histogram

För att kunna göra jämförelser mellan olika datamaterial och även mellan datamaterialen och motsvarande teoretiska sannolikhetsfördelningar måste man använda normerade histogram.

Det här laborationsmomentet går ut på att du själv skall skriva en grafisk rutin som skapar ett normerat histogram. Funktionen skall sparas som en `M-fil` så att du senare under kursen vid behov kan använda den.

Uppgift: Innan du börjar skriva rutinen tänk igenom och svara på följande frågor: Varför skall man normera histogrammet så att den sammanlagda ytan av staplarna blir ett? Hur skall man gå tillväga för att göra detta i *Matlab*? Vilka inparametrar är lämpligt att använda?

Ett exempel på hur man gör en egen `m-fil` finns i ovanstående avsnitt. Tekniken är i grunden rätt enkel, man skriver i stort sett bara de vanliga MATLAB-kommandona i en särskild s.k. `M-fil`. Börja med att öppna en editor: ur rullgardinsmenyn **File** kan du välja **New** och sedan **M-file**. Ett fönster med rubriken **Matlab editor/Debugger** öppnas, och där kan du skriva in din programkod.

För att kunna skapa en funktion som producerar normerade histogram, kan du ha stor nytta av följande standardfunktioner i `MATLAB`: `hist, bar, size och range`. Börja med att kontrollera hur dessa funktioner fungerar, framförallt vilka möjligheter de har.

```
>> help hist
>> help histc
>> help bar
>> help range
```

Spara sedan filen under t. ex. filnamnet `mitthisto.m` (suffixet `.m` är nödvändigt för att MATLAB skall kunna hitta filen). Du kan spara en fil genom att välja **Save** i menyn **File**. Ett fönster med rubriken **Save as** öppnas då, och där skall du skriva namnet på din fil i rutan **Filname** och sedan klicka på **OK**. Om du loggat in med lösenord kommer din fil automatiskt att hamna i din hemkatalog, annars får du ange även detta när du sparar filen. När detta är gjort kan du ikonifiera editorn genom att klicka på lämplig ruta i fönstrets övre högra hörn.

Ibland fungerar inte den filhanteraren som håller rätt på aktuell `M-fil`. När man sparar den korrigerade `M-filen` ligger pekaren kvar på den gamla. Om så skulle vara fallet kan man separat i MATLAB's kommandofönster innan man provar den korrigerade `M-filen` skriva:

```
>> clear funktionsnamn.m % i det här fallet clear mitthisto.m
```

Nu kan du gå tillbaka till MATLAB-fönstret och använda `mitthisto` på samma sätt som alla andra funktioner (om du t.ex. ger kommandot `help mitthisto`, så kommer de kommentarer du själv skrivit efter %-tecknen fram). Provkör programmet och kolla att det fungerar.

För att kunna provköra rutinen behöver du ett datamaterial, enklast är om du använder en uppsättning rektangelfördelade slumptal. Dessa kan du generera på följande sätt:

```
>> A = rand(1,1000);
```

Svar till vissa övningar

1. 0,005 095 307 350 95. Skriv `(2.9+sin(3/2.56))/(120+6.3e2)`
2. 0,258 632 622 514 24. Skriv `3.28^(-1.067*1.067)`
3. 0,300 418 745 459 29. Skriv `3.28^(-1.067)*1.067` eller `3.28^-1.067*1.067`
4. 3,866 488 265 396 41. Skriv `3.28^((-1.067)^2)`
5. −0,014 253 473 510 50. Skriv `(-2.34)^-5`
6. 1,161 834 242 728 28. Skriv `exp(2*0.075)`
7. 1,670 277 652 334 81. Skriv `13^(1/5)`
8. Det betyder $(3{,}28^{-3{,}2})^{-1\cdot 10^{-0}}$, vilket bör skrivas `(3.28^(-3.2))^(1e-0)`
9. Lösningarna till $x^2 + px + q = 0$ ges av $x_{1;2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$. Skriv alltså

   ```
   >> -3.29/2-sqrt((3.29/2)^2+10), -3.29/2+sqrt((3.29/2)^2+10)
   ```

   Matlab svarar −5,209 551 163 891 47 och 1,919 551 163 891 47.
10. 0,489 290 778 014 12. Skriv `asin(0.47)`
11. 0,252 069 358 243 11. Skriv `sin(14.6*pi/180)`
12. Dessa ord har andra förutbestämda betydelser.
13. `i` och `j` används i Matlab för den imaginära enheten om man vill räkna med komplexa tal.
19. x=0:0.05:10;
20. A=[1 2 3 4; 8 7 6 5];
21. z=zeros(7,5), z(3:5,2:4)=1
22. v=[-1 2 -3 4 6 -7]; v(v>0)=[]; Plockar bort alla positiva tal.
24. Dubbla frekvensen: Skriv `f2=sin(2*x);` För att få ett dovare ljud skriv `f3=sin(x/1.1);`
28. Skriv `x=linspace(-10,10,21);`
29. Använd punktvis multiplikation. Skriv `y=x.^2; z=(x-2).^2; w=x.^2+20;`
30. Skriv `plot(x,y,'-b'), hold on, grid on, plot(x,z,'-r')` etc.
31. `sin(x)` är en udda funktion, `exp(-x^2)` är en jämn funktion.
32. Skriv: `x=-5:0.1:5; f=(x+1)/2;plot(x,f,'-b'), hold on, plot(f,x,'-r')`
33. Jämför med de riktiga gränsvärdena: $\lim_{x\to 0} \frac{\sin(x)}{x} = 1$. $\lim_{x\to 0+} x\cdot\log(3\cdot x) = 0$. $\lim_{n\to\infty}(1+1/n)^n = e$.
34. Summan konvergerar mot värdet 2.

35. Minsta värdet av funktionen i intervallet är 0 då $x = 0$. Största värdet är $\exp(-1) = 0.3679\ldots$.

37. Obs! Använd funktionerna `x` och `y` från uppgift 36. Plotta tex. funktionen w mot x och y.

38. Linjerna är parallella.

39. Skriv `cd` Så får du reda på den aktuella katalogen.

40. Skriv `Xpkt=[0 1 2 4]'; Y=[1 1 3 5]'; X=[ones(4,1) Xpkt];`. Beräkna `B=X\Y;` Då får man att $a = 0.6$ samt $b = 1.0857$. Vill man plotta punkterna och den beräknade linjen i samma figur kan man skriva:

    `plot(Xpkt,Y,'*'), hold on, grid on, plot(X,X*B,'-r').`

    MK-skattningen ger såklart minsta kvadratsumman.

41. -1. Skriv `x*y'`

42. Skalärprodukten = 4. Vekorprodukten: `cross(u,v)` $= [-2\ -2\ 2]$.

43. Använd matrisformulering: `XB=Y`.

    Där: `X=[1 -2 1;2 -6 6; -3 5 1]; Y=[1 2 3]'; B=X\Y`. Resultat: `B=[10 6 3]'`.

44. Använd matrisformulering `A*B=Y`: `A=[1 -2 1;2 -6 6; -3 5 1]; Y=[1 2 3]'; B=A\Y=` något mycket stort. Determinanten för matrisen `A = 0`. I detta fall är matrisen `A` ej inverterbar.

45. Svar: 2500. Skriv: `k=1:2:100; summan=sum(k)`, eller ännu enklare `sum(1:2:99)`

46. Skapa först 50 rektangelfördelade slumptal. Skriv `A=rand(50,1). max(A), min(A)`

47. Skapa först 50 standardiserade normalfördelade slumptal. Skriv `A=normrnd(0,1,50,1); max(A), min(A), max(A)-min(A)`

48. Skriv `B=sort(A)`, respektive `C=-sort(-A)`. Det finns fler sätt kolla `sort`.

49. Prova tex med matrisen `A=normrnd(0,1,10,10)`.

50. ```
    function T= plotta(a,b);
    % kommentarer
    x=[a:0.01:b]';
    T=[x sin(x) cos(x)],
    plot(T(:,1),T(:,2),'-b')
    hold on, grid on
    plot(T(:,1),T(:,2),'-m')
    xlabel('x'),ylabel('sin(x) cos(x)')
    title('sin(x) blue line, cos(x) red line')
    ```

51. Implementera `m`-filen själv och kolla resultatet med `sqrt`-funktionen.

52. ```
    function y=exp2(x);
    \% Anrop: y=exp2(x);
    y=exp(x.^2);
    ```

53.
```
function y=expexp(x);
      \% Anrop: y=expexp(x);
      \% a=2 b= 10 y=exp(-exp((x-b)/a));
      b=10;
      a=2;
      z=(x-b)/a;
      y=exp(-exp(-z));
```

54.
```
function y=h(x);
\% Anrop: y=h(x);
y=1-expexp(x);
```

55. – 61. Dessa uppgifter får ni själv hitta på lösningar till.