STATISTICAL METHODS FOR SAFETY ANALYSIS FMS065
2005

MATLAB — en kort introduktion

Tord Rikte

Innehåll

1 Allmänt

Detta är ett försök att skriva en mycket kort introduktion till Matlab för deltagarna i kursen FMS065 *Statistiska metoder för säkerhetsanalys*.
Matlab är ett interaktivt program för numeriska beräkningar. Det fungerar som en (mycket) avancerad räknedosa, och har stora möjligheter till lagring av data. Att programmet är interaktivt, innebär i detta fall att man styr beräkningarna genom att man skriver instruktioner vid en prompt allteftersom beräkningarna framskrider. En enskild instruktion kommer vi att kalla ett kommando, och raden som man skriver på kommer vi följaktligen att benämna kommandoraden. Långa beräkningar eller beräkningar som skall upprepas många gånger kan automatiseras genom att man sammanställer flera kommandon i särskilda (text-)filer (makron), så kallade m-filer.
Programmet Matlab finns till Windows och Unix. I kursen FMS065 finns sex datorövningar. Datorövningarna är i sal MH:231, som har Windows-datorer. Vid Matlab-introduktionen kan det emellertid hända att du blir hänvisad till någon av Linux-datorerna i datorsal MH:230.
Till de ordinarie datorövningarna kommer fasta elevkonton att upprättas på datorerna i MH:231. Ditt lösenord får du bestämma själv vid första inloggningen. De fasta elevkontona kommer att vara öppna så länge kursen pågår.
Till Matlab-introduktionen använder vi öppna besökskonton. Eftersom besökskontona är öppna för alla, ser workspacen ut som föregående användare lämnade den. Spara därför ingenting på dessa konton; näste användare får tillgång till det! De öppna besökskontona kommer efter Matlab-introduktionen bara att finnas tillgängliga på datorerna i MH:230.

1.1 Inloggning på Windows 2000-datorerna, Matlab-introduktion

Så här loggar man in på en Windows 2000-dator vid Matlab-introduktonen (öppna besökskonton):

1. Om inte någon dialogruta syns på skärmen, tryck Ctrl+Alt+Delete. En dialogruta visar sig nu på skärmen.
2. Skriv in ditt användarnamn och lösenord.
3. Välj domänen STUDENT i dialogrutan vid Log on to.
4. Vänta tills en vanlig Windows-workspace har stabiliserat sig.
5. Klicka bort eventuella pop-up-meddelanden.

Inloggning på Linux-datorerna

Vid inloggning på Linux-datorerna så får ni ett temporärt användarnamn och lösenord av handledarna.

1. Skriv in ditt användarnamn
2. Skriv in ditt lösenord i avsedd ruta.
3. Vänta tills en workspace har stabiliserat sig.
4. Klicka längst ner till vänster på symbolen med ett K, varvid en meny visar sig (“Start Applications”). Välj under Terminal Program alternativet System Konsole. Ett terminalfönster, i vilket kommandon skrivs, visar sig då på skärmen.

1.2 Start av Matlab

Så här startar man Matlab-programmet (Windows 2000): Om en ikon för Matlab 6.5 (Windows 2000) finns på workspacen, kan du dubbelklicka på den. Annars går du in på Programs → MATLAB 6.5 → MATLAB 6.5 (Windows 2000) på Start-menyn. Vänta tills ett Matlab-fönster öppnat sig och en prompt visar sig. I Matlab 6.5 ser prompten ut så här: >>. Programmet Matlab är nu i gång och redo att mottaga kommandon vid prompten. Matlab-fönstret är öppet under hela Matlab-sessionen.
I Matlab 6.5 finns en Java-interaktion som kan vara opålitlig. Stäng av den genom att i Matlab-fönstrets meny-bar välja View → Desktop Layout → Command Window Only.
På en Linux-dator, skriv helt enkelt i ett terminalfönster matlab. Programmet Matlab är nu redo att mottaga kommandon vid prompten. Matlab-fönstret är öppet under hela Matlab-sessionen..

1.3 Särskild instruktion för kursen FMS065

Nedanstående gäller för Windows 2000. För kursen FMS065 behöver fyra särskilda initieringsåtgärder vidtagas omedelbart efter varje uppstart av Matlab. För det första: skriv vid Matlab-prompten, >>,

```
>> mh_init fms065
```

Avsluta med vagnretur, ”return”. Kommandot gör så att Matlab hittar de datafiler som är specifika för kursen FMS065. För det andra: skriv in

```
>> initwafo
```

vid Matlab-prompten och avsluta igen med vagnretur. Kommandot gör att Matlab hittar det rutinbibliotek (s k *toolbox*) som bär akronymen WAFO. Mera om detta nedan. För det tredje: Om du arbetar med Matlab 6.5 eller Matlab 6.1 under Windows 2000, skriv

```
>> cd F:\
```

vid Matlab-prompten och avsluta med vagnretur. För det fjärde: skriv in

```
>> path(cd,path)
```

vid Matlab-prompten och avsluta med vagnretur. Kommandot gör så att Matlab alltid hittar filer i din hemkatalog oavsett var i filträdet du befinner dig. Det är huvudsakligen bara i hemkatalogen som du kan spara undan filer.

1.4 Att avsluta Matlab

Matlab avslutar man genom att man skriver

```
>> exit
```

eller

```
>> quit
```

vid Matlab-prompten, >>. Avsluta med vagnretur, ”return”. Man kan också väja Exit MATLAB ur File-menyn i Matlab-fönstret.

1.5 Utloggning från Windows 2000-datorerna

Utloggning från Windows 2000-datorerna går till så här:

1. Spara allt som skall sparas. Avsluta alla program som Matlab, MSWord och MSExcel.
2. Välj Shut Down under Start-menyn. Invänta en dialogruta.
3. Välj Log off euler23 (exempel) i dialogrutan.
4. Klicka OK i dialogrutan.
5. Vänta tills utloggningen fullföljts. Stäng INTE av datorn.

1.6 Utloggning från Linux-datorerna

Utloggning från Linux-datorerna går till så här:

1. Spara allt som skall sparas. Avsluta alla program som t.ex. Matlab.
2. Klicka med höger musknapp och välj logout längst ned i den meny som visar sig.
3. Vänta tills utloggningen fullföljts. Stäng INTE av datorn.

2 Vi börjar så smått med Matlab

Logga in på datorn och starta Matlab enligt ovan, om du inte redan har gjort det. Glöm inte att utföra de särskilda instruktionerna i avsnitt 1.3 ovan.
Vid Matlab-prompten, `>>`, skriver man in vad man vill att Matlab skall beräkna. Vi kommer alltså att kalla detta att man skriver in ett kommando. En markör visar var på kommandoraden nästa tecken kommer att hamna. Avsluta varje inskrift med vagnretur, ”return”; först då exekveras kommandot.

2.1 Matlab i stället för räknedosa

Matlab kan man använda i stället för räknedosa. Syntaxen är nu i början självinstruerande:
Exempel. Beräkna $3{,}54 + 4 \cdot 5{,}59^{-1{,}36}$. Lösning: Skriv

```
>> 3.54+4*5.59^-1.36
```

följt av vagnretur. På skärmen står nu

```
>> 3.54+4*5.59^-1.36

ans =

    3.9251

>>
```

Svaret blev alltså 3,9251 (approximativt). Den nedre prompten visar att kommandot är utfört, och att programmet väntar på att ett nytt kommando skall skrivas in.
Exempel. Beräkna $\sin\frac{4\pi+1}{3}$. Lösning:

```
>> sin((4*pi+1)/3)
```

Exempel. Beräkna $9{,}2876 \cdot 10^{-2} + 7{,}124 \cdot 10^{-3}$. Lösning:

```
>> 9.2876e-2+7.124e-3
```

Exempel. Beräkna $\sqrt{1{,}19^2 - 1} - \sin\frac{\pi}{2} + e^2$. Lösning:

```
>> sqrt(1.19^2-1)-sin(pi/2)+exp(2)
```

Två eller flera kommandon kan sammanföras på en och samma kommandorad om de åtskiljs av kommatecken.
Exempel. Beräkna de tre uttrycken $9{,}2876 \cdot 10^{-2} + 7{,}124 \cdot 10^{-3}$, $\sqrt{1{,}19^2 - 1} - \sin\frac{\pi}{2} + e^2$ och $2 - 3$. Lösning med svarsutskrift:

```
>> 9.2876e-2+7.124e-3, sqrt(1.19^2-1)-sin(pi/2)+exp(2), 2-3

ans =

    0.1000


ans =

    7.0341


ans =

    -1

>>
```

Svaren ställs upp i den ordning i vilken motsvarande kommandon skrevs in på kommandoraden.
Skriver man ett tecken fel, används backstegningsknappen ovanför vagnreturknappen som deleaturknapp. Knappen märkt Delete är opålitlig. Det är lätt att skriva fel när det är många värdesiffror. Längre ned kommer vi att lära oss hur man kan ge (lämpliga) namn åt numeriska värden, så kallad variabeltilldelning.
Från exemplen ovan lär vi oss följande.

- 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!).

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 $\sin 14{,}6^\circ$.

2.2 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 (=tryck ned Ctrl-tangenten och håll den 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).

Övningar

12. Hur många kommandon har du givit sedan du startade Matlab? Stega bakåt och räkna!

13. Leta tillbaka till det kommando som var det första att inledas med `3`.

14. Skriv åter in och låt exekvera din lösning till övning 2. Lös sedan övningarna 3 och 4 genom att bläddra och ändra med piltangenterna, Ctrl-kommandona och backstegstangenten enligt föregående stycke.

2.3 Om man behöver omedelbar hjälp

Om man behöver hjälp med innebörden av, eller syntaxen för, någon Matlab-funktion eller något Matlab-kommando, exempelvis kvadratrotsfunktionen `sqrt`, kan man skriva

```
>> help sqrt
```

och en liten hjälptext skrivs ut på skärmen. I hjälptexten finns ofta, men inte alltid, en hänvisning till besläktade eller liknande funktioner/kommandon. Gör det till en vana att alltid anropa `help` då du gör bekantskap med en ny funktion eller ett nytt kommando i Matlab.

Övningar

15. Tag reda på vilken bas som används av logaritmfunktionen `log`.

16. Beräkna $\ln 5{,}33$.

17. Finns någon funktion för 10-logaritmer? Beräkna $\lg 5{,}33 = \log_{10} 5{,}33$.

18. Vad heter inversfunktionen $\arctan$ till $\tan$ i Matlab?

Ett annat bra sökhjälpmedel är `lookfor`-kommandot. Första raden i hjälptexten till ett Matlab-kommando eller en Matlab-funktion brukar utgöra en ytterst kort sammanfattning av kommandot/funktionen i fråga, där några viktiga nyckelord om möjligt har medtagits. Raden läses av lookfor-rutinen. Om man exempelvis anger kommandot

```
>> lookfor hyperbolic
```

genomsöks alla hjälptexters första rad efter ordet hyperbolic. De kommandon/funktioner där ordet förekommer skrivs ut på skärmen. På så sätt kan man med hjälp av nyckelord kanske finna en lämplig, tidigare obekant Matlab-rutin. Det fungerar endast med ett enda sökord åt gången.

Övningar

19. Finns det någon funkton för beräkning av fakulteter i Matlab? *Fakultet* heter *factorial* på engelska.
20. Fakulteter kan beräknas med hjälp av gammafunktionen Γ. Hur? Använd `lookfor` och `help`.
21. Beräkna 20! (fakultet).
22. Vad finns det i Matlab för funktioner för avrundning till heltal? Ledning: *Avrundning* heter på engelska *rounding*. Det är emellertid bättre att skriva `lookfor round` än `lookfor rounding` eftersom man då får med alla formerna *round*, *rounds*, *rounded* och *rounding*.

När man är trött och undrar varför datorn gjorde som den gjorde, får man (nästan) alltid kärnfulla förklaringar med kommandot `why`.

Övning

23. Vaffor gjorde datorhel. . . tet på detta viset?

2.4 Om Matlab ”hänger sig”

Som du kanske märkt tager kommandot `lookfor` rätt så lång tid att exekvera. Ibland tycker man det går alldeles för långsamt och gitter inte vänta på något svar. Då kan man avbryta exekveringen i förtid med Ctrl+c. Ctrl+c används vid alla tillfällen då man önskar avbryta ett kommando eller ett funktionsanrop i förtid, alltså inte bara vid `lookfor`. Detta är särskilt användbart då Matlab-kommandon hakar upp sig, ”hänger sig”.

Övning

24. Kör

    ```
    >> lookfor premiepensionsvalet
    ```

    och avbryt efter ett tag.

2.5 Variabeltilldelning

Du har kanske märkt också att alla uträkningar ovan avslutades med att datorn skrev ordet `ans` (answer), följt av ett likhetstecken och därefter svaret, exempelvis

```
>> 3.28^(-1.067*1.067)

ans =

    0.2586

>>
```

(Skriv in och exekvera!) Man säger att man har tilldelat variabeln `ans` värdet av `3.28^(-1.067*1.067)`. Det innebär att svaret finns lagrat och åtkomligt i variabeln `ans`. Verifiera följande.

```
>> ans

ans =

    0.2586

>> ans+3

ans =

    3.2586

>>
```

Även andra variabler kan man tilldela värden. Medan variabeln `ans` får sitt värde automatiskt, måste man tilldela andra variabler deras värden uttryckligen på kommandoraden, varvid likhetstecknet fungerar som tilldelningstecken.
Exempel. Några tilldelningar och ty åtföljande beräkningar:

```
>> massa=81.4

massa =

   81.4000

>> langd=1.2, bredd=0.35, tjocklek=24e-3

langd =

    1.2000


bredd =

    0.3500


tjocklek =

    0.0240

>> volym=langd*bredd*tjocklek

volym =

    0.0101

>> densitet=massa/volym

densitet =

  8.0754e+003

>>
```

(Skriv in och exekvera!) Om variabelnamnen, som här, har valts med litet eftertanke, blir beräkningarna överskådligare och lättfattligare, varvid fel lättare upptäcks. Det är väl inte svårt att förstå vad ovanstående beräkning handlar om?
Lägg märke till att vid tilldelningarna i föregående exempel ändras inte variabeln `ans`.
I variabelnamn kan endast stora bokstäver (`A-Z`), små bokstäver (`a-z`), siffertecken (`0-9`) och understrykningstecknet (`_`) användas. Intet variabelnamn får heller inledas med ett siffertecken eller med ett understrykningstecken; `densitet2` är alltså ett tillåtet variabelnamn, medan `2densitet` inte kan fungera som variabelnamn.

Övningar

25. Kontrollera vilket värde du har på variabeln `ans` för tillfället.

26. Ovan lade vi märke till att tilldelningskommandon inte ändrar värdet på variabeln `ans`. Om man bara återkallar en variabel, ändras inte heller värdet på `ans`. Verifiera detta med följande kommandosekvens.

```
>> 3+2
>> ans
>> london=4
>> ans
>> london
>> ans
>> london+0
>> ans
```

27. Vilket värde får variabeln `ans` om man giver följande fyra kommandon? Tänk först och kontrollera sedan.

```
>> 3+2
>> k=3*ans
>> k-ans^(ans-4)
>> ans*ans
```

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

29. 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
```

Med kommandot `who` får man reda på vilka variabler man har deklarerat. Kommandot `whos` (överkurs) är en utförligare form av `who`.

Övningar

30. Vilka variabler har du deklarerat fram till nu? Finns variabeln `ans` med på listan? Finns variabeln `pi` med på listan?

31. Med kommandot `clear` borttager man en variabel (eller flera variabler) ur datorns minne. Tag reda på hur `clear` fungerar och radera några valfria variabler, exempelvis variablerna `tjocklek` och `k`.

32. Hur borttager man enkelt *alla* egendeklarerade variabler?

2.6 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

33. Kör och begrunda följande kommandosekvens.

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

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

35. 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

36. Studera nedanstående Matlab-körning försedd med kommentarer. Skriv sedan in och exekvera, om du tycker att du behöver.

```
>> 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

37. 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)
```

3 Matriser

Namnet Matlab är en hopdragning av **Mat**rix **lab**oratory. Programmet Matlab, skrivet av matematikern Cleve Moler, framtogs just för att bli ett smidigt matrishanteringsprogram. Fortfarande är den bärande datastrukturen just matrisen, och Matlabs styrka ligger i att notationen är enkel och kompakt.
En matris är ett rektangulärt schema av tal, exempelvis är $\begin{pmatrix} 1{,}95 & -2{,}65 \cdot 10^{-2} & \sin 5^\circ \\ \tan(\ln 4{,}87) & 0 & 3/7 \end{pmatrix}$ en matris av storleken 2×3 (2 rader och 3 kolonner). Matriser kan förstås vara av olika storlek. En matris som har lika många rader som kolonner ($n \times n$), sägs vara kvadratisk.

3.1 Att bilda matriser i Matlab

I Matlab skrivs matrisen $\begin{pmatrix} 1{,}95 & -2{,}65 \cdot 10^{-2} & \sin 5^\circ \\ \tan(\ln 4{,}87) & 0 & 3/7 \end{pmatrix}$ ovan in så här (låt oss kalla matrisen `A`):

```
>> A=[1.95 -2.65e-2 sin(5*pi/180); tan(log(4.87)) 0 3/7]
```

Skriv in och exekvera! På skärmen står nu

```
>> A=[1.95 -2.65e-2 sin(5*pi/180); tan(log(4.87)) 0 3/7]
A =
   1.95000000000000  -0.02650000000000   0.08715574274766
 -81.31251232751785                   0   0.42857142857143
>>
```

Vill man att matrisen inte skall skrivas ut på skärmen, avslutar man kommandoraden med ett semikolon, `;`, som vanligt. Innanför matrisens klamrar, `[]`, markerar dock ett semikolon läget för ett radbyte. Inom varje rad åtskiljs kolonnerna med ett blanksteg (”mellanrum”) eller ett komma, `,`. Det är viktigt att man får precis lika många kolonner i alla raderna, annars kan inte Matlab tolka det inskrivna.
Operationen

```
>> size(A)
ans =
     2     3
>>
```

ger att `A` är just en 2×3-matris. I datorövningarna kommer du att möta matriser som är mycket stora, kanske $10\,000 \times 3$ eller ännu större. Sådana matriser utgör stora statistiska datamaterial eller simuleringsresultat och finns lagrade på särskilda filer. Matlab är väl ägnat att hantera dem.

Övningar

38. Skriv in följande fyra matriser: `B` $= \begin{pmatrix} -2 & 4 \\ 3 & 0 \\ 2 & 5 \end{pmatrix}$; `C1` $= \begin{pmatrix} 1 & 0 & 1 & 3 \\ 0 & 2 & 0 & -1 \end{pmatrix}$; `C2` $= \begin{pmatrix} 6 & 1 \\ -2 & 4 \\ 7 & -1 \end{pmatrix}$; `D` $= \begin{pmatrix} 4 & 7 & 0 \end{pmatrix}$. Se till att du behåller dem i datorns minne; du behöver dem nedan.

39. Antag att du inte känner storlekarna på matriserna `B`, `C1`, `C2` och `D` från uppgift 38. Bestäm deras storlek med kommandot `size`.

Det finns många andra sätt att skapa nya matriser. Här är några (skriv in och exekvera följande fem exempel!):

- Enhetsmatriser skapas med kommandot `eye`. Enhetsmatriser är kvadratiska matriser med ettor på huvuddiagonalen och nollor överallt annars. Exempelvis ger operationen

```
>> eye(4)
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>>
```

enhetsmatrisen av storleken 4×4.

- Matriser av idel ettor skapas med kommandot **ones**. Exempelvis ger operationen

```
>> ones(2,5)
ans =
     1     1     1     1     1
     1     1     1     1     1
>>
```

en matris av storleken 2×5 bestående av idel ettor.

- Matriser av idel nollor skapas med kommandot **zeros**. Exempelvis ger operationen

```
>> zeros(3,2)
ans =
     0     0
     0     0
     0     0
>>
```

en matris av storleken 3×2 bestående av idel nollor.

- Ibland vill man generera slumptal. Ofta vill man ha slumptalen jämnt fördelade mellan 0 och 1. På statistikerspråk kallas sådana slumptal för rektangelfördelade slumptal mellan 0 och 1. Matriser av sådana slumptal erhålls i Matlab med kommandot **rand** (engelskans *random*, *randomise*). Exempelvis ger operationen

```
>> rand(2,3)
ans =
   0.95012928514718   0.60684258354179   0.89129896614890
   0.23113851357429   0.48598246870930   0.76209683302739
>>
```

ett exempel på en matris av storleken 2×3 bestående av idel rektangelfördelade slumptal. Önskas så kallade standard-normalfördelade slumptal, byts **rand** ut mot **randn**. Mera om olika slumptal kommer du att få lära dig under själva kursen FMS065.

- Diagonalmatriser erhålls med kommandot **diag**. Följande exempel förklarar hur kommandot fungerar.

```
>> diag([3 10 0 -4 2])
ans =
     3     0     0     0     0
     0    10     0     0     0
     0     0     0     0     0
     0     0     0    -4     0
     0     0     0     0     2
>>
```

Nya matriser kan ställas samman av tidigare bildade matriser. Försök förstå vad som händer i detta exempel: Med matriserna `B`, `C1`, `C2` och `D` från uppgift 38 kan vi bilda (skriv in och exekvera!)

```
>> E=[4 D -3; ones(3,1) B C2; C1 [0; 3]]
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     2     5     7    -1
     1     0     1     3     0
     0     2     0    -1     3
>>
```

Övningar

40. Bilda i Matlab matrisen

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 9 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Gå till väga på samma sätt som vid bildandet av matrisen `E` ovan. Kalla matrisen `mymatrix` och radera den inte. Hur stor är den? Tänk på att om kommandot blir för långt, kan man bryta raden med tre punkter, `...`.

41. Generera en 5×10-matris, vars första rad består av idel rektangelfördelade slumptal mellan 0 och 1, vars andra rad består av idel nollor, vars tredje rad ser exakt likadan ut som den första raden, och vars två sista rader består av idel ettor.

3.2 Transponat och Hermiteskt konjugat

När en matris transponeras, gör man om matrisen så att kolonn 1 blir rad 1, kolonn 2 blir rad 2, etc. Om ursprungsmatrisen A var en $m \times n$-matris, blir den transponerade matrisen A^{T} (transponatet) en $n \times m$-matris, exempelvis

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

I Matlab erhålls transponatet med efterställd punkt, omedelbart följd av apostrof, `.'`. Exempelvis är

```
>> A.'
```

transponatet av `A`. Så kallat Hermiteskt konjugat (komplexkonjugering + transponering) erhålls med en ensam, efterställd apostrof , `'`. Exempelvis är

```
>> A'
```

det Hermiteska konjugatet av `A`. I statistiska sammanhang räknar man oftast med endast reella tal (inte komplexa), och då sammanfaller transponering och Hermitesk konjugering. Därför kommer vi i fortsättningen att skriva `A'` när vi i egentligen menar `A.'`.

Övningar

42. Skriv ut transponaten av matriserna `B`, `C1`, `C2` och `D` från uppgift 38, matrisen `E` tidigare och matrisen `mymatrix` från övning 40.

43. Vad för resultat får följande kommandon? Blev det som väntat?

```
>> B'', (B')'
```

44. Vad för resultat får följande kommando?

```
>> eye(4)'
```

3.3 Kolon-notation

En matris som utgörs av endast en enda rad (storlek $1 \times n$), kallas en radmatris. En matris som utgörs av endast en enda kolonn (storlek $n \times 1$), kallas en kolonnmatris. Rad- och kolonnmatriser brukar sammanfattningsvis kallas *vektorer*. Det finns ett sätt att bilda särskilda rad- och kolonnmatriser: med konstruktionen `z1:delta:z2` får man en radvektor som löper från `z1` till `z2` (så nära `z2` som möjligt) i steg om `delta`. Den kortare konstruktionen `z1:z2` förutsätter att steget `delta` har värdet 1. Motsvarande kolonnmatriser erhålls medelst transponering. Vi låter fyra korta exempel illustrera (skriv in och exekvera!):
Exempel.

```
>> x1=3:2:11
x1 =
     3     5     7     9    11
>>
```

Exempel.

```
>> x2=(3:0.5:8)'
x2 =
   3.00000000000000
   3.50000000000000
   4.00000000000000
   4.50000000000000
   5.00000000000000
   5.50000000000000
   6.00000000000000
   6.50000000000000
   7.00000000000000
   7.50000000000000
   8.00000000000000
>>
```

Kolonavgränsarna tycks alltid ha absolut lägsta prioritet; därför behövs parenteserna i det senaste exemplet; prova

```
>> x2B=3:0.5:8'
```

och se: transponatet ansluter till det avslutande värdet `8` och inte till den sammantagna radmatrisen `3:0.5:8`.
Exempel.

```
>> x3=10:-1:4
x3 =
    10     9     8     7     6     5     4
>>
```

Exempel.

```
>> x4=9:15
x4 =
     9    10    11    12    13    14    15
>>
```

I detta exempel förutsätt alltså steget vara 1.

Övningar

45. Skriv in och exekvera följande kommando och begrunda resultatet. Fick sista elementet värdet `pi`?

    ```
    >> x5=(-pi):pi
    ```

46. Tag reda på hur Matlab-funktionen `linspace` fungerar och jämför resultaten av följande fyra kommandon.

```
>> linspace(0,1,10)
>> linspace(0,1,11)
>> 0:0.1:1
>> 0:(1/9):1
```

Förstår du vad som händer?

47. Om man skriver in och exekverar

```
>> x6=10:3
```

erhåller man en tom matris. Hur stor är den? Förvånad? Alla radmatriser, även tomma, har i Matlab storleken $1 \times n$, alltså även om $n = 0$. Vad händer om `x6` transponeras? En 0×0-matris (ja!) skrivs i Matlab

```
>> x7=[]
```

3.4 Att hämta element ur en matris

När man skall hämta ett bestämt element (eller flera) ur en matris anger man elementets plats (rad, kolonn). Man kan också tillgripa kolon-notation (`:`) och det reserverade ordet `end`. Vi blir förhoppningsvis klokare med ett exempel, där vi åter betraktar matrisen `E`:

```
>> E=[4 D -3; ones(3,1) B C2; C1 [0; 3]]      % Återskapa E
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     2     5     7    -1
     1     0     1     3     0
     0     2     0    -1     3
>> E(2,4)        % Elementet på rad 2 i kolumn 4
ans =
     6
>> z1=E(4,4)     % Fjärde diagonalelementet (rad 4, kolonn 4)
z1 =
     7
>> z2=E(end,2)   % Elementet på sista raden i kolonn 2
z2 =
     2
>> z3=E(end-1,end) % Elementet på näst sista raden i sista kolonnen
z3 =
     0
>> E(1,:)         % Hela rad 1
ans =
     4     4     7     0    -3
>> z4=E(:,3)      % Hela kolonn 3
z4 =
     7
     4
     0
     5
     1
     0
```

```
>> E(:,end-2)     % Hela näst näst sista kolonnen
ans =
     7
     4
     0
     5
     1
     0
>> E(3:5,4)      % Raderna 3--5 i kolonn 4
ans =
    -2
     7
     3
>> E(3:5,:)      % Raderna 3--5
ans =
     1     3     0    -2     4
     1     2     5     7    -1
     1     0     1     3     0

>>
```

Övningar

48. Plocka ut elementet på rad 3 i kolonn 2 ur matrisen `B` och multiplicera med elementet på rad 2 i kolonn 4 i matrisen `C1`.

49. Plocka ut hela andra raden i matrisen `C1` och lagra den i en variabel `quintus`.

50. Plocka ut det tredje elementet i `quintus` från uppgift 49 ovan. För rad- och kolonnmatriser finns en kortare notation

    ```
    >> quintus(3)
    ```

    som är viktig att känna till.

51. Vad ger följande kommandosekvens (använd `mymatrix` från uppgift 40) för slutresultat?

    ```
    >> m=size(mymatrix)
    >> mymatrix(1:m(1),1:m(2))
    ```

52. Förstår du detta?:

    ```
    >> E([1 3:end],(end-2):end)
    ans =
         7     0    -3
         0    -2     4
         5     7    -1
         1     3     0
         0    -1     3
    >>
    ```

53. Skriv

    ```
    >> help colon
    ```

    och se vad där står.

54. En ofta användbar notation är

    ```
    >> E(:)
    ```

    Vad innebär den?

55. Vad innebär detta?:

    ```
    >> E(end:-1:1,end:-1:1)
    ```

    Tänk först och exekvera sedan.

56. Vilket värde får `ans` efter följande kommando?

    ```
    >> E(:,ones(1,3))
    ```

3.5 Att ändra element i en matris

Med samma teknik som ovan kan man ändra bestämda element i redan givna matriser genom tilldelning. Försök förstå det exempel med matrisen `E` som visas här (Återskapa först `E` enligt ovan om det behövs, exekvera sedan):

```
>> E
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     2     5     7    -1
     1     0     1     3     0
     0     2     0    -1     3
>> size(E)
ans =
     6     5
>> E(4,3)=3*8
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     2    24     7    -1
     1     0     1     3     0
     0     2     0    -1     3
>> E(end,1)=44
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     2    24     7    -1
     1     0     1     3     0
    44     2     0    -1     3
>> E(4,:)=ones(1,5)
E =
     4     4     7     0    -3
     1    -2     4     6     1
     1     3     0    -2     4
     1     1     1     1     1
     1     0     1     3     0
    44     2     0    -1     3
```

```
>> E(:,1)=[3 -7 6 0 9 13]'  % Varför transponeras högerledet?
E =
     3     4     7     0    -3
    -7    -2     4     6     1
     6     3     0    -2     4
     0     1     1     1     1
     9     0     1     3     0
    13     2     0    -1     3
>> E(:,1)=[3;-7;6;0;9;13] % Varför transponeras inte högerledet nu?
E =
     3     4     7     0    -3
    -7    -2     4     6     1
     6     3     0    -2     4
     0     1     1     1     1
     9     0     1     3     0
    13     2     0    -1     3
>> E(3:5,4)=(1:3)'
E =
     3     4     7     0    -3
    -7    -2     4     6     1
     6     3     0     1     4
     0     1     1     2     1
     9     0     1     3     0
    13     2     0    -1     3
>> E(3:5,:)=zeros(3,5)
E =
     3     4     7     0    -3
    -7    -2     4     6     1
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
    13     2     0    -1     3
>>
```

Övningar

57. Spara undan det tredje elementet på rad 2 i matrisen `C1` i någon lämplig variabel som du väljer själv. Sätt det tredje elementet på rad 2 till -99 i matrisen `C1`. Återställ matrisen.

58. Betrakta åter matrisen `E` sådan den ter sig efter exemplet ovan. Gör så att raderna 3, 4 och 5 blir identiska med sista raden. Använd endast en tilldelningssats.

59. Betrakta åter matrisen `E` sådan den ter sig efter övning 58 ovan. Skifta ordningen på näst sista kolonnen i `E` så att det översta elementet i kolonnen blir det nedersta, det näst översta elementet blir det näst nedersta o.s.v. Det går att göra med en enda tilldelning.

60. I samband med att vi nu håller på med att stapla om i matriser på olika sätt, kan det vara bra att ta reda på vad kommandona `flipud` och `fliplr` gör.

61. Hur ser matrisen `G` ut efter följande kommandosekvens? Kan du förklara?

```
>> [m n]=size(B);  % Märk hur man kan spara undan matrisens storlek!
>> G=zeros(2*m+1,2*n+1);
>> G(2:2:end,2:2:end)=B
```

På matriser kan man i Matlab utföra flera operationer (i exemplen använder vi `B`, `C1`, `C2` och `D` från ovan):

A. *Negation.* En matris kan negeras, vilket innebär att alla element i matrisen negeras.

Exempel.

```
>> -B
ans =
     2    -4
    -3     0
    -2    -5
>> -D
ans =
    -4    -7     0
>> --D
ans =
     4     7     0
>>
```

B. *Addition, subtraktion.* Två matriser av samma storlek kan adderas och subtraheras. Additionen och subtraktionen sker elementvis, exempelvis

```
>> B+C2    % B och C2 är lika stora
ans =
     4     5
     1     4
     9     4
>>  C2-B
ans =
     8    -3
    -5     4
     5    -6
>>
```

C. *Multiplikation och division med skalär.* En matris kan multipliceras med ett reellt tal. Det innebär att man multiplicerar alla element i matrisen med det reella talet. Det reella talet kan multipliceras till matrisen både från vänster och från höger. På samma sätt kan man dela en matris med ett reellt tal. Divisionen sker vanligen från höger.
Exempel.

```
>> 2*C1
ans =
     2     0     2     6
     0     4     0    -2
>> C1*(-3)
ans =
    -3     0    -3    -9
     0    -6     0     3
>> C2/2-B*3
ans =
   9.00000000000000 -11.50000000000000
 -10.00000000000000   2.00000000000000
  -2.50000000000000 -15.50000000000000
>>
```

D. *Addition och subtraktion med skalär.* Ett reellt tal kan adderas till, och subtraheras från, en matris. Det innebär att det reella talet adderas till (subtraheras från) varje element i matrisen. Det reella talet kan adderas till matrisen både från vänster och från höger. Subtraktion sker endast från höger.
Exempel.

```
>> 2+C1
ans =
     3     2     3     5
     2     4     2     1
```

```
>> C1-3
ans =
    -2    -3    -2     0
    -3    -1    -3    -4
>> C1+0
ans =
     1     0     1     3
     0     2     0    -1
>>
```

E. *Matrismultiplikation.* Vanlig matrismultiplikation mellan en $m \times k$-matris och en $k \times n$-matris resulterar — som bekant — i en $m \times n$-matris. I Matlab skrivs vanlig matrismultiplikation med asterisk , `*`, precis som vanlig multiplikation mellan skalärer.
Exempel.

```
>> B*C1     % B är 3 x 2, och C1 är 2 x 4
ans =
    -2     8    -2   -10
     3     0     3     9
     2    10     2     1
>> B*C2'    % B är 3 x 2, och C2 är 3 x 2
ans =
    -8    20   -18
    18    -6    21
    17    16     9
>>
```

F. *Elementvis multiplikation och division av matriser.* Om man har två matriser som är lika stora, kan man multiplicera dem elementvis. Man kan också dividera dem elementvis. Elementvis multiplikation och division signaleras med `.*` (en punkt, omedelbart följd av en asterisk) och `./` (en punkt, omedelbart följd av ett snedstreck) respektive.
Exempel.

```
>> B.*C2    % B är 3 x 2, och C2 är 3 x 2
ans =
   -12     4
    -6     0
    14    -5
>> D./[-4 5 1]
ans =
  -1.00000000000000   1.40000000000000                   0
>> ans(3)
ans =
     0
>> [2 4 3; 1 4 6]./[5 8 5; -1 10 3]
ans =
   0.40000000000000   0.50000000000000   0.60000000000000
  -1.00000000000000   0.40000000000000   2.00000000000000
>>
```

Elementvis division kräver, förstås, att den matris som utgör nämnarna inte har något element som är noll.

De vanliga prioritetsreglerna för de skalära operatorerna gäller för matrisoperatorerna.

Övningar

62. Beräkna $\begin{pmatrix} 1 & 9 \\ 6 & 7 \\ 2 & 6 \\ -3 & 1 \\ 0 & 0 \end{pmatrix} + 2{,}45 \begin{pmatrix} 1{,}9 & 1{,}7 \\ 4{,}6 & 2{,}7 \\ -3{,}5 & 6{,}5 \\ 1{,}0 & 4{,}1 \\ 1{,}1 & 0 \end{pmatrix}$ i Matlab.

63. Beräkna $2{,}45 \begin{pmatrix} 1 & 9 \\ 6 & 7 \\ 2 & 6 \\ -2 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1{,}9 & 4{,}6 & -3{,}5 & 1{,}0 & 1{,}1 \\ 1{,}7 & 2{,}7 & 6{,}5 & 4{,}1 & 0 \end{pmatrix}$ i Matlab.

64. Beräkna $(1 \cdot 2, 2 \cdot 2, 3 \cdot 2, \ldots, 77 \cdot 2, 78 \cdot 2)$, alltså en radmatris, i Matlab.

65. Beräkna $(1 \cdot 2, 2 \cdot 4, 3 \cdot 6, \ldots, 77 \cdot 154, 78 \cdot 156)$, alltså en radmatris igen.

66. Beräkna $1 \cdot 2 + 2 \cdot 4 + 3 \cdot 6 + \ldots + 77 \cdot 154 + 78 \cdot 156$. Hint: Vilken storlek får produkten av en 1×78-matris och en 78×1-matris?

67. Man kan räkna ut potenser elementvis med notationen `a.^b`. Om `a` och `b` båda är matriser, måste de vara lika stora. Jämför följande tre kommandon.

```
>> [9 16 4 25 1].^(1/2)
>> 2.^[-1 0 1 2 3]
>> [9 16 4 25 1].^[-1 0 1 2 3]
```

68. Beräkna en 3×2-matris som på plats (i, j) har värdet $2 + 1{,}3\frac{(1-0{,}5B_{i,j})^3+6}{B_{i,j}-3{,}5}$, där $B_{i;j}$ är element $(i; j)$ i matrisen `B` som vi använt så mycket ovan.

69. Konstruktionen `a./b` fungerar också om `a` är en skalär och endast `b` är en matris. Hur beräknar man då $\begin{pmatrix} 1/6 & 1 \\ -1/2 & 1/4 \\ 1/7 & -1 \end{pmatrix}$ snabbt i Matlab om matrisen `C2` $= \begin{pmatrix} 6 & 1 \\ -2 & 4 \\ 7 & -1 \end{pmatrix}$ är given som vanligt.

70. Det finns även en konstruktion `a.\b` som ger precis samma svar som `b./a`. Illustrera detta med något exempel som du hittar på själv.

3.6 Standardfunktioner. Linjära ekvationssystem

De flesta standardfunktioner, såsom `sqrt`, `exp`, `log`, `sin`, `asin`, `cos` och `tan`, klarar av att ha matriser som argument, varvid funktionen beräknas elementvis. Om vi till exempel vill bilda en kolonnvektor med cosinusvärden för alla vinklar mellan 0° och 90° i steg om $0{,}2^\circ$, kan man göra så här:

```
>> y=cos((0:0.2:90)'*pi/180);
```

Detta är mycket användbart när man skall låta Matlab rita diagram på skärmen. Mera om det nedan.

Övningar

71. Lagra alla värden från -10 till $+10$ i steg om 0,01 i en vektor `x`. Bilda en vektor `y` (av samma storlek) som innehåller motsvarande arctan-värden.

72. Beräkna $\begin{pmatrix} e^{1/6} & e^{-1/2} & e^{1/7} \\ e^{1} & e^{1/4} & e^{-1} \end{pmatrix}$ om matrisen `C2` $= \begin{pmatrix} 6 & 1 \\ -2 & 4 \\ 7 & -1 \end{pmatrix}$ är given som vanligt.

Låt $y = Ax$ vara ett linjärt ekvationssystem med n ekvationer (n obekanta). Då är x en obekant $n \times 1$ -matris, y en bekant $n \times 1$-matris och A en bekant kvadratisk $n \times n$-matris. Om A är inverterbar, har detta system lösningen $x = A^{-1}y$, där A^{-1} är inversa matrisen till A. I Matlab beräknas inversa matrisen till matrisen `A` med kommandot `inv(A)`. Man bör dock inte beräkna lösningen `x` till ekvationssystemet som

```
>> x=inv(A)*y
```

utan med den särskilda operationen

```
>> x=A\y
```

vilken i Matlab är numeriskt stabilare.

Övning

73. Lös ekvationssystemet $\begin{cases} 2x_1 + 3x_2 - x_3 = 1 \\ x_2 + 2x_3 = 0 \\ x_1 - x_2 = -1 \end{cases}$

Inversa matrisen till en kvadratisk matris `A` kan också uttryckas i Matlab som en vanlig (icke elementvis) potens, nämligen `A^-1` (inte någon punkt framför cirkumflexet). På samma sätt kan man skriva `A^0` för enhetsmatrisen, `A^1` för `A`, `A^2` för `A*A`, `A^3` för `A*A*A` o s v. Determinanten av en kvadratisk matris `A` beräknas med anropet `det(A)`.

3.7 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

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

75. Generera 50 rektangelfördelade slumptal mellan 0 och 1. Ange det största och det minsta värdet bland dem.

76. Generera 50 standard-normalfördelade slumptal. Ange 1) det största värdet, 2) det minsta värdet, 3) variationsbredden (avståndet mellan det största och det minsta värdet).

77. Sortera slumptalen i uppgift 76 i stigande ordning och i fallande ordning.

78. Funktionerna `max`, `min` och `sort` kan på begäran leverera två upplysningar. Konstruktionen

```
>> [vmax ix]=max(v)
```

levererar det största värdet bland `v`-elementen i ut-variabeln `vmax`, medan `ix` innehåller positionen i `v` för det största värdet. Analogt gäller för `min`. Konstruktionen

```
>> [vsort ix]=sort(v)
```

levererar de sorterade värdena i variabeln `vsort` och motsvarande `v`-positioner i variabeln `ix`. Det gäller alltså att `vsort` är detsamma som `v(ix)`. Vi skall nu utnyttja detta på en fördefinierad datavektor som heter `vector1`. Om du sitter i sal Windows-salen MH231 skriv in

```
>> load vector1.mat  % Alternativ 1: Windows
```

får du tillgång till den i Matlabs workspace. Sitter du i Linux-salen skapa istället en egen vektor

```
>> vector1 = rand(100,1); % Alternativ 2: Linux
```

Ange det tal i variabeln `vector1` som har det 1) minsta absolutbeloppet, 2) största absolutbeloppet, 3) näst minsta cosinusvärdet. Ange också motsvarande positionen i `vector1`. Hint: *Absolutbelopp* heter på engelska *absolute value*.

79. 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.

4 Diverse

4.1 Logiska operatorer och funktioner

I Matlab kan man hantera logiska uttryck. ”Sant” kodas som `1` och ”falskt” som `0`. Logiska relationsoperatorer i Matlab är `<`, `>`, `<=`, `>=`, `==` (lika med) och `~=` (ej lika med). Begrunda, skriv in och exekvera följande!

```
>> 3<5
ans =
     1
>> 3<3
ans =
     0
>> 3<=3
ans =
     1
>> 3==2
ans =
     0
>> 3==(1+2)
ans =
     1
>> 3~=2
ans =
     1
>> 3~=3
ans =
     0
>> [2 4 -3 8]>0   % Det fungerar elementvis även för matriser
ans =
     1     1     0     1
>> [1+1 1+2 1-4 7]==[2 4 -3 8]
ans =
     1     0     1     0
```

De logiska relationsuttrycken kan förbindas med `~` (”icke”), `&` (”och”) och `|` (”eller”). På matriser fungerar operatorerna elementvis. Eftersom prioritetsordningen för dessa logiska operatorer inte är den vanliga, rekommenderas att man slösar med parenterser. Begrunda, skriv in och exekvera följande!

```
>> ~[0 1 0 0 1; 1 0 1 1 1]
ans =
     1     0     1     1     0
     0     1     0     0     0
>> ([8 2 0; 3 -5 7]<=4)&(4>3)
ans =
     0     1     1
     1     1     0
>> a=4; b=3; c=((a>=1)&(a<=4))|(b==2) % 1<=a<=4 ELLER b==2
c =
     1
>>
```

Notera i det sista försöket att man kan tilldela en variabel ett resultat av en logisk operation (variabeln `c` ovan). Låt `A` vara en matris; till att börja med låter vi den vara en rad- eller kolonnmatris (vektor). Kommandot `find(A)` levererar då de positioner i matrisen som är nollskilda. Exempelvis ger `find([3 0 0 1 0 8])` svaret `[1 4 6]`. Detta används för att skapa funktionsvärden när funktionen har flera grenar. Exempel: Vi har en vektor `x=-10:0.01:10` och vill beräkna vektorn $f(\mathtt{x})$ där f ges av

$$f(x) = \begin{cases} = -x\sin(2x) & x < -\pi \\ = 0 & -\pi \le x < 1 \\ = 1 - \mathrm{e}^{-(x-1)} & x \ge 1 \end{cases}$$

Lösningen blir (svaret lagras i vektorn `y`)

```
>> x=-10:0.01:10;                % "Indata"
>> y=zeros(size(x));    % Skapar en svarsmatris av samma storlek som x
>> ix1=find(x<-pi);              % Första grenen
>> y(ix1)=-x(ix1).*sin(2*x(ix1));
>> ix2=find((x>=-pi)&(x<1));     % Andra grenen
>> y(ix2)=zeros(size(ix2));
>> ix3=find(x>=1);               % Tredje grenen
>> y(ix3)=1-exp(-(x(ix3)-1));
```

Förstod du detta? Skriv in och exekvera! För att åskådliggöra resultatet, skall vi låta Matlab rita en graf på skärmen:

```
>> plot(x,y)
```

Blev det som du trodde? Det som är bra med denna metod är att den faktiskt fungerar för godtyckliga matriser, inte för bara rad- och kolonnmatriser!
Genom att utnyttja möjligheten till så kallad logisk indexering kan man skriva det ovanstående kortare. Det är inte så lätt att förstå vad som händer vid logisk indexering, så vi kommer inte att gå in på logisk indexering i denna introduktionskurs.

Övningar

80. Gör om ovanstående exempel om matrisen `x` i stället är $\begin{pmatrix} 3{,}9 & 2 & -4{,}1 & 2 \\ 0{,}25 & -\pi & 0 & 8{,}3 \end{pmatrix}$ Kontrollera att det blev rätt genom uträkning element för element.

81. Simulera 10 000 rektangelfördelade slumptal mellan 0 och 1. Hur stor andel av dem är i kvadrat mindre än eller lika med $1/2$?

Ibland vill man uttrycka sig generellt om alla element i en rad- eller kolumnmatris. Till exempel vill man kanske veta om alla elementen i vektorn `v` är strängt större än noll. Då skriver man

```
>> all(v>0)
```

Om man i stället vill veta om något element i vektorn `v` är strängt större än noll, skriver man

```
>> any(v>0)
```

Övningar

82. Generera 10 rektangelfördelade slumptal mellan 0 och 1. Låt bli att titta på dem. Är något av dem mindre än 0,1? Är alla mindre än 0,9? Kontrollera dina svar genom att inspektera slumptalen.

83. Hur fungerar `all` och `any` om vi har godtyckliga matriser?

4.2 Att rita grafer, så kallad plottning

Matlab är rikt utrustat med grafiska rutiner. Man kan ställa in det mesta efter egna önskningar. Tyvärr är Matlabs rutiner för grafritning inte så självklara, så vi tar bara upp enkel tvådimensionell grafik i denna introduktion. I datorövning 3 om linjär regression kommer vi vid ett tillfälle att möta tredimensionell grafik.
Vi har redan tidigare ritat en graf. Som synes ritades grafen i ett separat fönster. Om man vill kan man öppna ett nytt fönster för varje graf man låter rita; de olika fönstren numreras då automatiskt 1, 2, 3, ... allteftersom

de öppnas. Endast ett fönster kan vara aktivt åt gången. Man låter fönster 3 (exempelvis) bli det aktiva fönstret genom att skriva `figure(3)` eller genom att klicka i fönster 3. Ett helt nytt fönster öppnas genom att skriva bara `figure`. Man stänger fönster 3 genom att skriva `close(3)`. Man stänger alla fönster genom att skriva `close all`. Läs mera om kommandona `figure` och `close` i `help`-texten. Ett gott råd är att hålla antalet öppna fönster nere.

Övning

84. Stäng alla fönster.

Antag att vi vill rita en graf över funktionen $f(x) = \cos(x)$ för $0 \leq x \leq 10$. Det görs så här:

```
>> x=0:0.1:10;   % Här genereras x-värdena
>> f=cos(x);     % Här generaras motsvarande funktionsvärden
>> plot(x,f)     % Här ritas diagrammet.
```

Kommandot `plot` öppnade ett nytt fönster, eftersom det inte fanns något tidigare öppnat fönster. Hade det funnits öppna fönster, hade grafritningen skett i det fönster som för tillfället vore aktivt.
Om man vill ha en kontinuerlig kurva, får man inte välja `x`-värdena alltför glest. Kurvans utseende kan ändras så här (skriv in och se vad som händer!):

```
>> plot(x,f,'.')  % Apostrofer!
>> plot(x,f,'--')
>> plot(x,f,'r')  % r som i red!
>> plot(x,f,'g*') % g som i green!
```

Läs mera om olika linjetyper, plotsymboler och färger (och hur man kombinerar dem) i hjälptexten till `plot`.
Om man vill rita två grafer i samma diagram, måste man ”låsa fast” den första grafen så att den inte ritas över när man plottar den andra grafen. Ett diagram ”låses” genom att man skriver `hold on`; ”upplåsning” sker följaktligen med `hold off`. Exempel: Antag att vi i samma diagram vill rita grafen $f(x) = \cos(x)$ för $0 \leq x \leq 10$ (heldragen linje, svart) och $g(x) = 1 - \mathrm{e}^{-x/2}$ för $0 \leq x \leq 10$ (streckad linje, rött). Så här gör man:

```
>> x=0:0.1:10;     % Här genereras x-värdena
>> f=cos(x);       % Här generaras motsvarande funktionsvärden (f)
>> g=1-exp(-x/2)   % Här genereras motsvarande funktionsvärden (g)
>> plot(x,f,'k-')  % Här ritas första grafen (svart, heldragen)
>> hold on         % "Lås fast" den första grafen, så att den inte ritas över!
>> plot(x,g,'r--') % Här ritas andra grafen (röd, streckad)
>> hold off        % "Lås upp" diagrammet, diagrammet är ju klart!
```

Det är alltid lättare att avläsa ett diagram om det finns ledlinjer i form av ett rutnät (en ”grid”). Lägg till en grid med kommandot

```
>> grid on
```

Vill man ta bort griden igen, skriver man

```
>> grid off
```

Vill man ändra axlarnas gradering, skriver man (i vårt exempel)

```
>> axis([-1 11 -2 2])
```

vilket ställer om så att x-axeln löper från -1 till $+11$ och y-axeln från -2 till $+2$. Slutligen kan man lägga till axelrubriker och figurrubrik. Pröva exempelvis

```
>> xlabel('x-värdena')
>> ylabel('y-värdena')
>> title('Grafer till funktionerna f och g')
```

Det går att zooma i Matlab-ritade diagram. I menyraden i figurfönstret finns två knappar med ett stiliserat förstoringsglas på. Det ena har ett plustecken i sig (för inzoomning) och det andra ett minustecken (för utzoomning). Klicka på inzoomnings-knappen. Klicka därefter med vänster musknapp någonstans i diagrammet och håll knappen nedtryckt. Dra musen en bit och du ser att en markeringsruta följer med på skärmen. Se till att få med ett intressant område av diagrammet i markeringsrutan. Släpp upp musknappen när du har fångat in det intressanta området. Som du ser, har du nu fått en förstoring av det markerade (intressanta) området. Utzoomning sker genom upprepade tryck på musens högerknapp; du behöver inte trycka på utzoomningsknappen i menyraden. Zoomningsfunktionen är litet opålitlig; det blir sällan som man tänkt sig i första försöket.
Det händer att man vill samla flera samhörande diagram i ett och samma figurfönster. Då använder man kommandot `subplot(m,n,p)` som delar in figurfönstret i `m`×`n` stycken mindre diagram (arrangerade i `m` rader och `n` kolonner) och sätter aktuellt underdiagram till nr `p` (numreringen går från vänster till höger, uppifrån och ned). (Alltså måste $\mathtt{p} \leq \mathtt{m} \cdot \mathtt{n}$.) Om $\mathtt{m} \leq 9$, $\mathtt{n} \leq 9$ och $\mathtt{p} \leq 9$, kan man förkortat skriva exempelvis `subplot(312)`, som staplar tre avlånga grafer ovanpå varandra (3×1) i figurfönstret och sätter det mellersta (nr 2) till att vara aktuellt underdiagram. Kommandot `subplot` använder man (med samma syntax) också för att skifta mellan underdiagrammen. Plottning sker sedan i det aktuella diagrammet enligt ovan.

Övningar

85. Låt `y` vara en rad- eller kolonnmatris. Om man skriver `plot(y)` eller `plot(y,'.')`, plottas elementen i `y` mot ordningsnumren i `y`. Det blir alltså samma sak som `plot(1:length(y),y)` eller `plot(1:length(y),y,'.')` respektive. Prova detta genom att ge kommandot

    ```
    >> plot([2 2 2 0 0 -1 5],'*')
    ```

    i ett nytt fönster.

86. Generera 20 standard-normalfördelade slumptal och plotta dem i växande storleksordning (blåa asterisker förbundna med streckade räta linjer). Lägg till grid. Plotta i samma diagram 20 rektangel-fördelade slumptal mellan 0 och 1 i avtagande storleksordning (röda ringar förbundna med heldragna räta linjer).

87. Öppna ett nytt fönster. Låt översta tredjedelen av fönstret bli ett diagram. Låt nedre tredjedelen bli ett annat diagram. Dela in den mellersta tredjedelen i två diagram bredvid varandra. Övertyga dig om att detta är lösningen:

    ```
    >> subplot(311)    % Översta diagrammet
    >> subplot(313)    % Nedersta diagrammet
    >> subplot(323)    % Mellersta vänstra diagrammet
    >> subplot(324)    % Mellersta högra diagrammet
    ```

    Stäng inte fönstret.

88. Rita en graf över funktionen $f(x) = \sqrt{1-x^2}$ för $-1 \leq x \leq 1$ i det nedre diagrammet i graffönstret från uppgift 87 ovan. Som bekant beskriver funktionen f en halvcirkel. Ser grafen ut som en halvcirkel? Tryck in `help axis` och se vad kommandot

    ```
    >> axis equal
    ```

    gör. Giv kommandot! Hur ser grafen ut nu?

89. Generera 40 standard-normalfördelade slumptal i en vektor och plotta dem mot ordningsnumret. Diagrammet skall utgöra det övre diagrammet i figurfönstret från uppgift 87 ovan, och plot-symbolen skall vara en röd triangel av valfri orientering. Plot-symbolerna får inte vara förbundna med några linjer. Lägg till en grid.

90. Återvänd till det nedre diagrammet från uppgift 88 och lägg till en grid. Sätt också en rubrik `Halvcirkel`. Gå sedan vidare till det mellersta högra diagrammet. Övertyga dig om att du där ritar en liggande parabel om du ger följande kommandon.

    ```
    >> y=-5:0.2:5; x=y.^2; plot(x,y), grid on, title('Liggande parabel')
    ```

    Spara figurfönstret!

4.3 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 till 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 i skrivas ut på skärmen efter hand. Hur många iterationer krävdes?

Övning

91. Gå till det mellersta vänstra diagrammet från uppgift 87 ovan och rita däri tio koncentriska helcirklar med växande radier från 1 till 10 och medelpunkt i origo. Använd en for-sats. Cirklarna skall vara gröna till färgen. Tillse att graderingen blir lika på bägge axlarna. Lägg till en grid.

4.4 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
```

4.5 Teckensträngar

I exemplet på villkorssatser ovan bad vi datorn skriva ut

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

om det inlästa talet `x` var strikt mindre än noll. Vi mötte där teckenföljderna `'Ange ett tal här: '` och `'Haha, talet var negativt!'`, som är exempel på *teckensträngar* (eller bara *strängar*). Tidigare har vi mött teckensträngarna `'x-värdena'`, `'y-värdena'` och `'Grafer till funktionerna f och g'` i avsnittet om grafritning. När teckensträngar anges i klartext i Matlab-kod, måste de omgivas av apostroftecken (`'`). Det finns dock ingenting som hindrar att man låter teckensträngar representeras av variabler, exempelvis (skriv in och exekvera!)

```
>> president1='George Bush'
president1 =
George Bush
>> president2='Bill Clinton'
president2 =
Bill Clinton
>>
```

Teckensträngar kan hanteras på ungefär samma sätt som radvektorer (skriv in och exekvera!):

```
>> president3=[president1(1:6) ' W' president1(7:end)]
president3 =
George W Bush
>> capital='Washington'; firstpresident=['George ' capital]
firstpresident =
George Washington
>> length(president3)
ans =
    13
>>
```

Vill man skriva ut en teckensträng på skärmen, används kommandot `disp`:

```
>> disp([firstpresident ' grundade 1790 huvudstaden ' capital '.'])
George Washington grundade 1790 huvudstaden Washington.
>>
```

Övning

92. Tag reda på vad funktionerna `strcmp`, `int2str`, `num2str`, `ischar` och `isreal` gör. Gör några enkla försök.

5 Filhantering

5.1 Makron, så kallade m-filer

Om man har långa uträkningar, eller om man vill upprepa samma instruktioner vid senare tillfällen, eller om man behöver definiera egna funktioner, skriver man ett makro som sparas i en Matlab-makro-fil. Filerna skrivs i Matlab-kod och är rena textfiler. Alla Matlab-makro-filer har extensionen `.m`, därav benämningen m-fil. m-filer finns av två slag: skriptfiler och funktionsfiler. En skriptfil är helt enkelt en serie Matlab-kommandon som utförs som om man hade skrivit dem direkt på kommandoraden. En funktionsfil är en fil som vid anrop tar inargument och levererar ett svar, ett funktionsvärde. Ett exempel på en funktion är `mean` i funktionsfilen `mean.m`, vilken — som bekant — ger aritmetiska medelvärdet av elementen i en vektor. För att se hur filen `mean.m` är skriven, ger man kommandot `type mean`, varpå m-filen skrivs ut på skärmen.
Under Windows skriver man lämpligen in sin m-fil i Matlabs egen editor. Den startar man genom att i Matlab-fönstrets meny-bar välja File → New → M-file, eller genom att ange kommandot `edit` på kommandoraden. När filen är inskriven får man inte glömma att spara den.
Här är ett exempel på en funktionsfil som är tänkt att finnas i en m-fil `sinunits.m`:

```
function [y,xrad]=sinunits(x,angleunit,varargin)
%SINUNITS Sine of angle given in other units than 1 rad
%
%    SINUNITS(X,UNIT) is the sine of the elemnts in X
%    when measured in the unit UNIT. Currently supported
%    are UNIT='rad' (radians) 'deg' (degrees),
%    'gon' (gon), 'min' (minutes), 'sec' (seconds),
%    and '-' (Swedish military points).
%    Default is UNIT='rad'.
%    [Y,XRAD]=SINUNITS(X,UNIT): Here, firstly, Y will
%    contain the sine values of  the elements in X (when
```

```
%    X is measured in the unit UNIT), and, secondly,
%    XRAD will contain the values of X converted into
%    the unit 1 rad (radians).
%
if nargin>=3, error('Too many input arguments.'), end
if nargin==0, error('Too few input arguments.'), end
if nargin==1, angleunit='rad', end
if ~ischar(angleunit), error('Argument angleunit is not char.'), end
if ~isreal(x), error('Argument x is not real.'), end

if strcmp(angleunit,'rad')
  xrad=x;
elseif strcmp(angleunit,'deg')
  xrad=pi*x/180;
elseif strcmp(angleunit,'gon')
  xrad=pi*x/200;
elseif strcmp(angleunit,'min')
  xrad=pi*x/(180*60);
elseif strcmp(angleunit,'sec')
  xrad=pi*x/(180*3600);
elseif strcmp(angleunit,'-')
  xrad=pi*x/3150;
else
  error(['The unit ''' angleunit ''' is not supported'])
end

y=sin(xrad);
```

Första raden i en funktionsfil inleds alltid med det reserverade ordet `funciton`. Därefter följer en kommaseparerad lista på utvariablerna (`[y,unit]`), ett likhetstecken (`=`), funktionsnamnet (`sinunits`) och slutligen inargumenten inom parentes (`(x,angleunit,varargin)`). De efterföljande raderna med komentarer utgör den hjälptext som skrivs ut då man ger kommandot `help sinunits`. Först därefter kommer själva programtexten.

5.2 Datafiler

Om man vill lagra undan sina variabler till en annan Matlab-session, gör man på något av dessa tre sätt:

```
>> save                              % Sparar undan alla variabler i en fil matlab.mat.
>> save myfile1                      % Sparar undan alla variabler i en fil myfile1.mat.
>> save myfile2 B C1 C2 president3   % Sparar undan variablerna B, C1, C2 och president3
                                     % i en fil myfile2.mat.
```

Matlabs datafiler är på ett särskilt format och får extensionen `.mat`. Vill man ladda in en Matlab-datafil med namnet `myfile.mat` till sin Matlab-workspace, gör man så här:

```
>> load myfile.mat
```

Vill man se vilka variabler som finns i en Matlab-datafil med namnet `myfile.mat` utan att först läsa in filen i Matlab-workspacen, gör man så här:

```
>> whos -file myfile.mat
```

Man kan ibland få datafiler som är rena textfiler. Förutsatt att datafilerna inte innehåller någon annan text än siffertext och att det är lika många dataelement på varje rad, kan man ladda in sådana filer direkt med kommandot `load`. En matris lagrad i textfilen `thisdata.dat` får variabelnamnet `thisdata` i Matlabs workspace.

Övningar

93. Undersök vilka variabler som finns i filen `oxygenation.mat` som används i datorövningarna. Hur stora är matriserna? Ladda in variablerna i workspacen.

94. Spara undan alla de variabler som du har för tillfället i en fil `myvariables.mat`. Töm sedan Matlab-workspacen med kommandot `clear all`. Kontrollera med kommandot `who` att du inte har några variabler kvar. Ladda slutligen in variablerna igen från filen `myvariables.mat`. Verifiera med kommandot `who`.

95. Textfilen `atlantic.dat` innehåller s k signifikanta våghöjder uppmätta vid två lokaler i Atlanten. Den innehåller 582 mätdata. Ladda in filen. Vilket namn fick variabeln efter inladdningen i workspacen? Hur stor är variabeln?

Alla datafiler till de ordinarie datorövningarna kommer att vara mat-filer och dat-filer.
Om ett arbetsblad i MSExcel har sparats som en s k tab-separerad textfil, exempelvis under namnet `Bok1.txt`, kan den filen importeras i Matlab genom att man skriver

```
>> load Bok1.txt -ascii
```

vid Matlab-prompten. De data som finns lagrade i `Bok1.txt` kommer därefter att återfinnas som Matlab-variabeln `Bok1`. Detta fungerar under förutsättning, för det första, att filen `Bok1.txt` bara innehåller numeriska data (alltså ingen text), för det andra, att alla rader har samma antal element, och, för det tredje, att decimaltecknet utgörs av en punkt (inte ett komma). I Matlab 6.5 och Matlab 6.1 kan man läsa in data direkt från en MSExcel-fil (exempelvis `Bok1.xls`) med hjälp av kommandona `xlsread` och `xlsfinfo`; tasta in `help fileformats` och `help xlsread` så får du mera information. Om du vill, kan du starta MSExcel, skriva in några data, spara undan (se till att du får rätt filformat) och försöka läsa in data i Matlab från den undansparade filen.

5.3 Att spara undan och skriva ut figurer

Om man i 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 på datorerna i MH:230 och MH:231 skriver

```
>> print -f4
```

i Matlab, 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 (utskriftshanteraren), i vilket man specificerar skrivaren.

5.4 Kataloger och toolboxar

Under Windows 2000 kommer din hemkatalog att vara `F:\`. När du startar Matlab är det inte säkert att du hamnar i din hemkatalog, varför du omedelbart efter uppstart av Matlab bör ge kommandot `cd F:\` ; kommandot gör att just din hemkatalog blir aktuell katalog i Malab. Skall du läsa in datafiler från MSExcel, måste du se till att du flyttar över datafilerna till din hemkatalog. Alternativt får du förflytta dig med Matlab till defaultkatalogen för MSExcel. 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'
```

Som du nu märkt, finns det stora mängder Matlab-rutiner (m-filer). Man har skapat rutinbibliotek, s k toolboxar, för skilda ändamål, exempelvis statistik (Statistics Toolbox, `stats`), wavelets (Wavelets Toolbox, `wavelet`) och systemidentifiering (System Identification Toolbox, `ident`). För att se vilka toolboxar som finns inlagda, ge kommandona `ver` och `path`. För att se vilka rutiner som finns i en viss toolbox, exempelvis `stats`-toolboxen, ge kommandot `help stats`.

Övning

96. Tag som övning reda på vilka rutiner som finns i System Identification toolbox. Du behöver bara veta hur du hittar dem, inte vad de innebär.

Vid avdelningen för matematisk statistik vid Matematikcentrum i Lund har utvecklats en toolbox med rutiner för våganalys, materialutmattning och oceanografi, WAFO-toolboxen (Wave Analysis, Fatigue, Oceanography), `wafo`. Det är en förhållandevis stor toolbox, uppdelad i ett flertal s k moduler. Vid datorövningarna i FMS065 kommer vi att använda modulen `wstats` och en ensam rutin från modulen `misc`. För att Matlab skall kunna hitta WAFO-rutinerna, måste man ange katalogen där m-filerna finns lagrade. Detta görs med ett för kursen färdigskrivet kommando `initwafo`.
Om man vill veta var i filträdet som en viss m-fil finns, exempelvis `mean.m`, skriver man

```
>> which mean
```

Då skrivs hela sökvägen till filen `mean.m` ut på skärmen.
Det händer ibland när man ger kommandona `type` eller `which`, att man får som svar blott att rutinen i fråga är "built-in". Det innebär att filen tillhör Matlabs mest centrala; sådana rutinfiler är förborgade för användaren.

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)`
15. Basen är $\mathrm{e} = 2{,}718\,28\ldots$.
16. 1,673 351 238 177 75. Skriv `log(5.33)`
17. Funktionen heter `log10`. 0,726 727 209 026 57. Skriv `log10(5.33)`
18. Funktionen heter `atan`.
19. Funktionen `factorial`. Även `gamma` (se uppgift 20).
20. Låt n vara ett naturligt tal. Då erhålls $n!$ som $n! = \Gamma(n+1)$.
21. $2{,}432\,902\,008\,176\,640 \cdot 10^{18}$. Skriv `gamma(20+1)` eller `factorial(20)`
22. Funktionerna `ceil`, `fix`, `floor`, `round`.
26. `ans` har värdet 4 efter kommandosekvensen.
27. `ans` har värdet 100 efter kommandosekvensen.
28. Dessa ord har andra förutbestämda betydelser.
29. `i` och `j` används i Matlab för den imaginära enheten om man vill räkna med komplexa tal.
32. Med kommandot `clear all`
38. Skriv

    ```
    >> B=[-2 4; 3 0; 2 5], C1=[1 0 1 3; 0 2 0 -1]
    >> C2=[6 1, -2 4; 7 -1], D=[4 7 0]
    ```

39. Skriv

    ```
    >> size(B), size(C1), size(C2), size(D)
    ```

40. Skriv

```
>> [zeros(3,1) eye(3) zeros(3,2) diag([-6 8 9]);...
ones(3,5) zeros(3,4); ones(1,9); zeros(1,9)]
```

41. Skriv

```
>> rad1=rand(1,10); matris=[rad1; zeros(1,10); rad1; ones(2,10)]
```

42. Skriv

```
>> B', C1', C2', D', E', mymatrix'.
```

43. Dubbla transponat tar ut varandra.

44. Om 4×4-enhetsmatrisen transponeras, händer just ingenting, eftersom den är symmetrisk.

47. Storleken hos `x6` är 1×0 (ja!). Transponatet `x6'` blir 0×1.

48. -5. Skriv `B(3,2)*C1(2,4)`

49. Skriv

```
>> quintus=C1(2,:)
```

50. 0. Skriv `quintus(3)`

51. Den ger hela matrisen `mymatrix` åter.

54. Den staplar kolonnerna i `E` ovanpå varandra och bildar så en enda lång kolonnmatris. Den vänstraste kolonnen i `E` hamnar överst, och den högraste kolonnen i `E` hamnar nederst.

57. Skriv

```
>> w=C2(2,3), C(2,3)=-99, C(2,3)=w,
```

58. Skriv

```
>> E(3:5,:)=E(end*ones(1,3),:)
```

59. Skriv

```
>> E(:,end-1)=E(end:-1:1,end-1)
```

62. Körningen blir

```
>> [1 9;6 7;2 6;-3 1;0 0]+2.45*[1.9 1.7;4.6 2.7;-3.5 6.5;1.0 4.1;1.1 0]
ans =
   5.65500000000000  13.16500000000000
  17.27000000000000  13.61500000000000
  -6.57500000000000  21.92500000000000
  -0.55000000000000  11.04500000000000
   2.69500000000000                   0
```

63. Körningen blir

```
>> 2.45*[1 9;6 7;2 6;-3 1;0 0]*[1.9 1.7;4.6 2.7;-3.5 6.5;1.0 4.1;1.1 0]'
ans =
   1.0e+002 *
  Columns 1 through 4
   0.42140000000000   0.70805000000000   1.34750000000000   0.92855000000000
   0.57085000000000   1.13925000000000   0.60025000000000   0.85015000000000
   0.34300000000000   0.62230000000000   0.78400000000000   0.65170000000000
  -0.09800000000000  -0.27195000000000   0.41650000000000   0.02695000000000
                   0                  0                  0                  0
  Column 5
   0.02695000000000
   0.16170000000000
   0.05390000000000
  -0.08085000000000
                   0
```

64. Skriv `2*(1:78)`

65. Skriv `2*(1:78).*(1:78)`

66. 322 478. Skriv `2*(1:78)*(1:78)'`

68. `>> 2+1.3*((1-0.5*B).^3+6)./(B-3.5)`

69. `>> 1./C2`

71. `>> x=-10:0.01:10; y=atan(x);`

72. `>> exp(1./(C2'))`

73. Ekvationssystemet kan skrivas $y = Ax$, där $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$, $y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$ och $A = \begin{pmatrix} 2 & 3 & -1 \\ 0 & 1 & 2 \\ 1 & -1 & 0 \end{pmatrix}$.
    I Matlab erhålls

```
>> A=[2 3 -1;0 1 2;1 -1 0]; y=[1;0;-1]; x=A\y
x =
  -0.45454545454545
   0.54545454545455
  -0.27272727272727
```

    Exakt lösning är $x_1 = -5/11$, $x_2 = 6/11$, $x_3 = -3/11$.

74. 2500. Skriv `sum(1:2:99)`

75. `>> slumptal=rand(50,1); max(slumptal), min(slumptal)`

76. `>> slump=randn(50,1); max(slump), min(slump), max(slump)-min(slump)`

77. `>> sort(slump), -sort(-slump)`

78. **1.** $-9{,}036\,631\,354\,414\,626 \cdot 10^{-4}$. Position 9390. Skriv `[v1 ix1]=min(abs(vector1)), vector1(ix1)`
    2. 3,947 612 502 984 53. Position 8355. Skriv `[v2 ix2]=max(abs(vector1)), vector1(ix1)`
    3. $-3{,}156\,245\,608\,328\,08$. Position 4991. Skriv `[v3 ix3]=sort(cos(vector1)); ix3(2), vector1(ix3(2))`

80. Resultatet blir på skärmen

```
y =
   0.94497677994359   0.63212055882856  -3.85699528238707   0.63212055882856
                  0                  0                  0   0.99932446122481
```

81. Andelen hamnar med stor sannolikhet i intervallet 69 %--72 %. Skriv

```
>> sum(rand(10000,1).^2<=0.5)/10000
```

82.
```
>> slumptal=rand(1,10); any(slumptal<0.1), all(slumptal<0.9)
```

84.
```
>> close all
```

86.
```
>> plot(sort(randn(20,1)),'b*--'), grid on, hold on
>> plot(-sort(-rand(20,1)),'ro-'), hold off
```

87. Svaren till uppgifterna 88–91 blir klarare om vi skriver upp kommandona igen:

```
>> subplot(311)  % Översta diagrammet
>> subplot(313)  % Nedersta diagrammet
>> subplot(323)  % Mellersta vänstra diagrammet
>> subplot(324)  % Mellersta högra diagrammet
```

88.
```
>> subplot(313)
>> x=-1:0.01:1;
>> f=sqrt(1-x.^2);
>> plot(x,f)
>> axis equal
```

89.
```
>> subplot(311)
>> plot(rand(40,1),'vr')
>> grid on
```

90.
```
>> subplot(313)
>> grid on
>> title('Halvcirkel')
>> subplot(324)
>> y=-5:0.2:5; x=y.^2; plot(x,y), grid on, title('Liggande parabel')
```

91.
```
>> subplot(323)
>> t=linspace(0,2*pi,1000);
>> for i=1:10, plot(i*cos(t),i*sin(t),'g'), hold on, end, hold off
>> axis equal
>> grid on
```

93. I `oxygenation.mat` finns variablerna `X`, `h` och `v`. De är alla tre 12×1-matriser. Skriv

```
>> whos -file oxygenation.mat, load oxygenation.mat
```

94.
```
>> who, save myvariables, who, clear all, who, load myvariables.mat, who
```

95. Skriv

```
>> load atlantic.dat
```

    Variabeln fick namnet `atlantic`. Den är 582×1.

96. Skriv `help ident`

This is version 2005-01-04.