# INTRODUKTIONSÖVNINGAR I MATLAB

## MATLAB som räknedosa

Dessa uppgifter ger Dig en introduktion till MATLAB. Du kan göra dem i din egen takt och efter eget urval men tveka aldrig att rådgöra med din handledare. Tanken är att Du ska ha "Användarhandledning för MATLAB 5" tillgänglig för att kunna lösa uppgifterna. På så sätt lär Du dig också att hitta i boken vilket är mycket bra för kommande behov. Appendix A i boken innehåller även en elementär introduktion till MATLAB.

För uppgifterna 1 - 8 är kapitel 1 och 2 i MATLAB-boken relevanta. Ögna gärna i genom dessa kapitel först, men fördjupa Dig inte i sådant som verkar svårt eller obekant. Det kommer att klarna så småningom.

1. Starta upp MATLAB med kommandot **matlab** i UNIX. När Du senare vill gå ur MATLAB skriver Du **quit** eller **exit**.

   - Ge kommandot **demo** och studera i första hand alternativen

   **Numerics** (Functions of functions) och

   **Visualization** (2-D plots) och

   (i sista hand **Games**).

   - Ge kommandot **helpdesk** och studera speciellt

   **Getting started** och

   **MATLAB Functions** (by Subject och by Index)

   **Anm.** Helpdesk ligger på www och det kan ta lite tid att koppla upp den. Ett råd är att Du kopplar upp den i början av arbetspasset om Du avser att använda den. Alternativ till helpdesk är att använda **help**- eller **helpwin**- kommandona, som ger information om kommandon och funktioner. En första användning av help-kommandot kan vara att skriva **help help** för att få beskrivning av kommandot **help**.

   - Ge kommandot **helpwin**. Hur används detta kommando?

   **Anm.** Ett sätt att ta reda på vilka funktioner som finns i MATLAB är att använda **lookfor**. Kommandot **lookfor text** letar upp alla rutiner som har texten **text** i första raden av sin help-text. Om man i stället skriver **lookfor -all text** så letar MATLAB igenom hela help-texten, inte bara första raden.

   - Ge kommandot **lookfor logarithm** för att se vilka logaritm-funktioner som finns och hur de anropas.

   **Anm.** Du avbryter en pågående exekvering i MATLAB med **Control-c**

2. Beräkna följande uttryck i MATLAB. Elementära funktioner hittar du i boken på sid 26-27 eller med **help elfun** och speciella fördefinierade variabler kan du läsa om på sid 23 i boken.

$sin(\pi)$, $ln(3^2)$, $ln(\sqrt{e})$, $log_{10}(100)$, $e^2$, $|5+3i|$, $e^{i\pi}$, $1/0$, $0/0$, $(1/0)*0$, $(0/0)*(1/0)$

**OBS.** MATLAB skiljer på små och stora bokstäver. Kolla vad som händer om Du skriver t.ex. SIN(2).

3. Tilldela variabeln **x** värdet $\pi$, variabeln **y** värdet 355 och variabeln **z** värdet 226. Beräkna sedan följande uttryck och skriv ut resultatet med 5 resp 15 siffror, se **help format**:

   $a = x/2$, $b = y/z$, $c = a - b$, $d = a - sin(y)$

   Ange utgående från dessa beräkningar en bra rationell approximation av $\pi$. Hur stort är felet?

   **Anm.** Om man inte vill att MATLAB ska svara på alla beräkningar med att skriva ut variabeln eller det beräknade värdet så avslutar man beräknings-satsen med semikolon (;).

4. Eftersom datorn räknar med ändlig precision blir de flesta resultat närmevärden. Vi vill undersöka hur stora felen kan bli i följande (matematiskt korrekta) trigonometriska formler:

   $sin(2x) = 2sin(x)cos(x)$ och $sin^2(x/2) = (1 - cos(x))/2$.

   Välj några olika värden på $x$ och låt MATLAB beräkna och skriva ut skillnaden mellan vänster- och högerled i formlerna dividerat med vänsterledets värde, dvs vi tittar på **relativa** felet. Är det några värden som verkar ge stora fel i någon av formlerna? Jämför med variabeln $eps$, se sid 23 i boken. Hur blir det speciellt med små $x$-värden i den andra formeln?

5. Vi antar att $R >> r$ (innebär mycket större än) och betraktar formeln $R(1 - cos(r/R))$, som då kan approximeras med $r^2/(2R)$, där approximationen blir bättre och bättre ju mindre $r$ är. (Detta följer av Taylorutveckling som vi skall gå igenom senare i kursen.) Låt nu $R = 4 * 10^7$ och undersök genom att utföra beräkningarna i MATLAB hur stort felet i approximationen är om $r = 10^p$, för $p = 0,\ 1,\ 2,\ 3,\ 4,\ 5$. Verkar resultatet rimligt? Har resultatet något med uppgift 4 att göra?

6. Rita kurvorna $y = x$ och $y = 2\sin x$. (Använd gärna MATLAB.) Definiera en talföljd $x_n$ genom $x_0 = 1,5$ och $x_{n+1} = 2\sin x_n$. Undersök $\lim_{n\to\infty} x_n$. Har gränsvärdet något med ekvationen $x = 2\sin x$ att göra? Kan du bevisa det?

7. (a) Visa att ekvationen $x^3 - x = 1$ har en rot mellan 1 och 2.
   (b) Skriv ekvationen $x = x^3 - 1$ och försök lösa ekvationen rekursivt genom att sätta $a_0 = 1$ och $a_{n+1} = a_n^3 - 1$, $n = 0, 1, 2, \ldots$ Vad händer?
   (c) Skriv ekvationen $x = (x+1)^{1/3}$ och försök lösa ekvationen genom att låta $a_0 = 1$ och $a_{n+1} = (a_n + 1)^{1/3}$, $n = 0, 1, 2, \ldots$ Vad händer?
   (d) Försök bevisa din observation i (c) (nämligen att $a_n \to a$ där $a$ löser ekvationen) genom att bevisa att $a_n \leq 2$ och att $a_n$ är en växande följd.

## Programmering i MATLAB

8. Vi ska nu se hur man kan skapa egna kommandon och funktioner i MATLAB. Dessa ska ha namn som slutar med **.m** och skapas som filer i en editor. MATLAB har en egen editor men vi rekommenderar att Du använder EMACS, som Du har tillgång till på Din arbetsstation och kan få hjälp med. Innan Du gör denna uppgift bör Du läsa avsnitt 2.9 i boken ganska noggrant, utom exempel 2.20, som vi inte är riktigt mogna för ännu.

   Skriv nu en kommandofil med namnet **addera.m** som helt enkelt består av matlabberäkningen $a = b + c$. Spara filen och testa att exekvera den från MATLAB. Variablerna **b** och **c** måste ha fått värden innan förstås.

   Lägg nu till kommentaren "Jag adderar b plus c i variabeln a" **först** i filen addera.m. (Kommentarer skrivs med procent-tecken i första position.) Vad händer om Du nu skriver **help addera** i MATLAB. Just det, Du har skapat en egen help-information.

   Gör om **addera.m** till en funktionsfil i stället, med parametrarna **b** och **c**, och anropa den från MATLAB som en funktion.

   Ge nu kommandot **type addera** och konstatera att din funktion betraktas som MATLAB's egna, detta under förutsättning att den ligger i aktuellt MATLAB-bibliotek dvs aktuellt directory där Du startade MATLAB från. Man kan kolla vilka egna m-filer man har i biblioteket genom kommandot **what** i MATLAB. (Man kan även komplettera de bibliotek som man vill att MATLAB ska söka i genom kommandot **matlabpath**)

9. M-filer kan vara rekursiva dvs anropa sig själva. Skriv en rekursiv funktion **fakultet.m**, som beräknar **n! = 1*2*3*...*n** med **n** som parameter, där **n** är ett positivt heltal. Det är lämpligt att använda någon form av **if-sats**, se avsnitt 12.1 och Exempel 12.7 i boken.

10. I uppgift 7 använde vi en villkorssats. Denna är ett verktyg vid programmering i MATLAB. Läs nu om andra sådana i avsnitt 12.1 och 12.2 i boken.

    Harmoniska serien är 1, 1/2, 1/3, 1/4, ... Skriv en funktion **harmonisk.m** med parameter **n**, som beräknar partialsumman av **n** termer (1+1/2+1/3+1/4+...+1/n) genom att på lämpligt sätt använda en for-loop.

    Harmoniska serien är inte konvergent dvs summan blir godtyckligt stor bara man tar med tillräckligt många termer. Vi vill veta hur många termer som behövs för att summan ska överstiga värdet **tak**. Skriv en ny funktion, som har **tak** som parameter och som returnerar antalet termer. Använd lämpligen en while-loop. Hur många termer behövs då tak=12?

## Grafik i MATLAB

11. Det står mycket om grafik i kapitel 13 i boken. I Exempel 13.1 illustreras kommandot **plot**. MATLAB arbetar med **vektorer** och **matriser**, som du är bekant med från linjär algebra. Variablerna **x** och **y** i exempel 13.1 (a) är vektorer medan **Z** i Exempel 13.7 är en (komplex) matris.

Rita kurvor för funktionerna

$f(x) = x(1 - sin\ \pi x)$ och $g(x) = 5e^{-x/2}/(3 - 2\ cos\ 2\pi x)$

i intervallet $0 \leq x \leq 4$. Pröva dig fram med lämpligt steg. Använd sedan kommandot **subplot** för att i nya bildrutor rita summafunktionen $f(x) + g(x)$, produkten $f(x) * g(x)$ och kvoten $f(x)/g(x)$.

**OBS!** Notera de så kallade elementvisa aritmetiska operationerna, som ska användas för vektorer (och matriser), se avsnitt 3.5 i boken.

12. Skriv en funktionsfil **step.m** som definierar funktionen

$$f(x) = \begin{cases} 0, & 0 \leq x \leq 1 \\ 1, & 1 < x \leq 2 \\ 0, & 2 < x \leq 3 \end{cases}$$

Skriv en kommandofil som ritar ut funktionen genom att anropa funktionsfilen. Testa även alternativet att använda MATLAB-funktionen **fplot**.

**Extra:** Försök skriva funktionsfilen **step.m** så att du kan rita genom satserna

$x = 0 : 0.01 : 3;\ \ plot(x, step(x))$

dvs på samma sätt som för exempelvis $x = 0 : 0.01 : 3;\ \ plot(x, sin(x))$

13. Du vet att funktionsfilen **decrease.m** definierar en funktion som avtar strikt från ett positivt värde då $x = -1$ till ett negativt värde då $x = 1$. Skriv en kommandofil som ritar ut funktionen från $x = -1$ så länge den är positiv. Testa ditt program på funktionen $f(x) = -2(sin\ x + 0.3)$, som Du alltså lägger i funktionsfilen **decrease.m**.

**Extra 1:** Försök skriva kommandofilen utan att använda while-loop, utan i stället med hjälp av **min** och **find** funktionerna i MATLAB.

**Extra 2:** Gör om kommandofilen till en funktionsfil **rita.m**, som har den funktion som ska ritas ut som parameter. Hur man gör när man vill ha en funktion som parameter till en annan funktion framgår av avsnitt 12.4 i boken. Nu ska Du alltså kunna rita ut den funktion som är definierad i **decrease.m** (så länge den är positiv) med anropet **rita('decrease')** i MATLAB.

14. Rita funktionen $tan(x)$ på intervallet $(-\pi/2,\ 9\pi/2)$. Här får man tänka sig för lite på grund av de stora funktionsvärdena. En ledning är att Du kan använda kommandot **axis** och en for-loop på lämpligt sätt. Resultatet bör bli något liknande Figur 1.

15. Man bör försöka att undvika for-loopar och while-loopar om man vill få så effektiv MATLAB-kod som möjligt. Ofta kan man utnyttja att MATLAB är konstruerat för att hantera vektorer och matriser på ett bra sätt. Vi ger ett exempel. Följande två alternativa MATLAB-sekvenser ritar funktionen $sin(t)$ för $0 \leq t \leq 20$:

Alternativ 1:

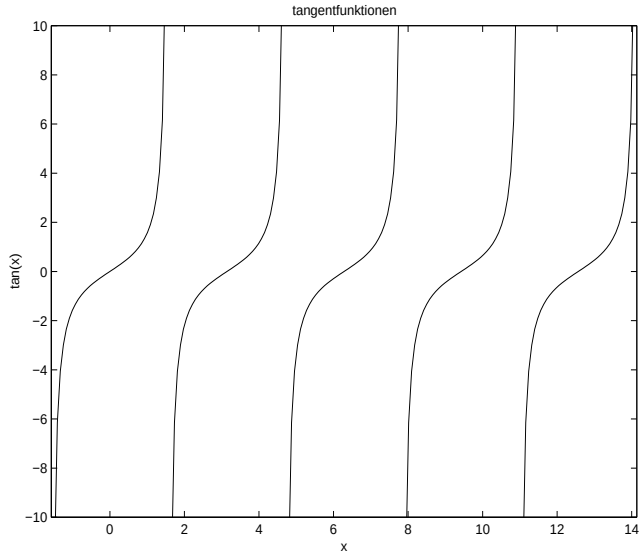


Figure 1: Tangentfunktionen i MATLAB

$t = 0 : 0.001 : 20;$

$plot(t, sin(t))$

Alternativ 2:

$n = 0;$

$for\ t = 0 : 0.001 : 20;$

$n = n + 1;$

$x(n) = t :$

$y(n) = sin(t);$

$plot(x, y)$

end

Undersök hur många flyttalsoperationer, rutinen **flops** i MATLAB, och hur lång tid, rutinerna **tic** och **toc** i MATLAB, som de båda alternativen kräver.

16. Cirklar ritas smidigt i MATLAB med parameterframställning: $x = x_c + r\ cos\phi\ ,\ \ y = y_c + r\ sin\ \phi$, där $(x_c,\ y_c)$ är centrumkoordinaterna, $r$ är radien och där vinkeln $\phi$ går från $0$ till $2\pi$. Rita en cirkel, som har centrum i (0.72, 1.42) och radien 2.5. Använd steget (upplösningen) $2\pi/60$ map vinkeln. Markera ut mittpunkten.

    **Obs.** Använd **axis equal** (efter plot-satsen) för att få skalorna på axlarna lika, annars ser det ut som en oval.

17. Slumpa ut tio cirklar med radie 5 på området $0 \leq x \leq 60\,,\ \ 0 \leq y \leq 45$, jämför uppgift 15. Klicka på tre av cirklarna, som skall rödfärgas. Använd **ginput** och **fill** i MATLAB.

## MATLAB-övningar på vektorer och matriser

18. Skapa följande vektorer med hjälp av "kolon-operatorn":

    $x = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$, $y = [0\ 0.2\ 0.4\ 0.6\ 0.8\ 1]$

    Skapa vektorn $x_3 = [1\ 8\ 27\ 64\ ...\ 1000]$ med hjälp av vektorn x.

    Bilda delvektorerna $x1$ och $x2$ som första resp andra hälften av vektorn $x$.

    Återskapa $x$ ur delvektorerna $x1$ och $x2$

19. Låt $x = [1\ 2\ 3]^T$ och $y = [3\ 2\ 1]^T$. Transponat ges av apostrof ' i MATLAB. Vi transponerar för att få kolonnvektorer, vilket är det normala att arbeta med i linjär algebra. Undersök och försök förstå följande beräkningar:

    $x'$, $y'$, $x.*y$, $x./y$, $x'*y$, $x*y'$, $(x*y')*x$ **Extra:** $x/y$, $x\backslash y$

    **Ingen utmaning:** Beräkna **n!** med kolonoperatorn och **prod**-funktionen i MATLAB, jämför övning 7.

    **Lite större utmaning:** Beräkna summan av n termer i den harmoniska serien med endast ett anrop av **sum**-funktionen i MATLAB, jämför med övning 8.

20. Bilda diagonalmatrisen

$$D = diag(2,3,4) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

    Skriv **help diag** så får Du veta hur man gör. Bilda sedan matrisprodukterna $DA$ och $AD$, där

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

    Vilka transformationer på $A$ åstadkommer multiplikationerna med diagonalmatrisen.

21. Bilda en så kallad permutationsmatris $P$ genom att byta första och tredje raden i enhetsmatrisen av ordning 3. Detta görs med satserna (försök förstå vad som händer):

    $P = eye(3),\ \ P([1,3],:) = P([3,1],:)$

    Bilda nu matrisprodukterna $PA$ och $AP$, där $A$ är matrisen i uppgift 18. Vilka transformationer på $A$ åstadkommer multiplikationerna med permutationsmatrisen.

22. Givet matrisen $A = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}$

bilda delmatriserna $A_{ij}$, av ordning 2 x 2, i partitioneringen $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$. Hur kan man sedan återskapa $A$ från delmatriserna?

23. Kontrollera att associativa och distributiva lagarna gäller för matriserna

$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $C = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix}$

dvs att $A(BC) = (AB)C$ respektive $A(B+C) = AB+AC$ och $(A+B)C = AC+BC$. Visa att matrismultiplikation inte är kommutativ genom att visa att $AC \neq CA$. Vad gäller för $AB$ och $BA$?

24. Beräkna $x^T y$ och $xy^T$ där $x$ och $y$ är vektorerna i uppgift 22. Bilda nu produkten $Axy^T B$, dels som $A(xy^T)B$ och dels som $(Ax)(y^T B)$, där $A$ och $B$ är matriserna i uppgift 26. Undersök med **flops** vilket som kräver minst beräkningsarbete. Försök förstå skillnaden.

## Linjära ekvationssystem

25. Lös det linjära ekvationssystemet $Ax = b$, där $A$ är matrisen i uppgift 25 och $b = [23\ 32\ 33\ 31]^T$. Är lösningen korrekt? Är det enda lösningen?

26. Låt $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 1 \end{bmatrix}$ och $b = [1\ 3]^T$. Lös ekvationssystemet $Ax = b$. Vad ger backslash i MATLAB. Är det enda lösningen? Använd **rref**-kommandot för att ta fram reducerad trappstegsform och därur bestämma lösningsmängden.

27. Lös matrisekvationen $AX = B$ där $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 5 \\ 7 & 8 & 0 \end{bmatrix}$ och $B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ Vad blir $X$ om $B$ är enhetsmatrisen?

28. Lös matrisekvationen (dvs bestäm $X$ i uttrycket) $AXB + 2AX = C$, där $A = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 4 & 2 \\ 0 & -1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$ och $C = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 3 & 1 \end{bmatrix}$.

## Ekvationslösning

29. Ljudnivån $L$ (i decibel) på avståndet $r$ meter från en ljudkälla ges av formeln

    $L = L_0 - 20\ ^{10}log(r) - \beta r$

    där $L_0$ är ljudnivån 1 meter från ljudkällan och $\beta$ är en parameter som anger hur ljudet försvagas då det går genom luften. ($\beta$ beror på luftens viskositet, temperatur och fuktighet.) Vi vill för $L_0 = 80$ dB bestämma det avstånd där ljudnivån är 20 dB då $\beta = 1.15 \cdot 10^{-3}$. Formulera en ekvation som ska lösas, rita en lämplig graf och läs av en approximation av avståndet med **ginput**. Bestäm sedan en noggrannare approximation med **fzero**.

30. En lång stång upphettas momentant vid tiden $t = 0$ i mittpunkten. Med denna punkt i $x = 0$ och $x$-axeln utmed stången bestäms temperaturen $T(x, t)$ för $t > 0$ av uttrycket

    $T(x, t) = Ce^{-x^2/kt}/\sqrt{t}$

    där parametrarna $k = 5.17$ och $C = 28.3$ beror av värmeledningsförmågan respektive den tillförda värmen. Beräkna under hur lång tid som temperaturen vid $x = 1$ överstiger $20^o$. Använd **fplot**, **ginput** och **fzero**.

31. Bestäm alla reella nollställen till

    $f(x) = x^3 - x - 0.3847$

    Utnyttja **fplot**, eventuellt **zoom** och **ginput** för att lokalisera nollställena grovt och **fzero** för att beräkna dem noggrannare.

    Studera känsligheten i funktionen med avseende på $x$-koefficienten, i ostörd variant är alltså denna koefficient = $-1$. Stör nu koefficienten med störningar $+0.1$ resp $-0.1$ och se vad som händer med nollställena.

    Använd även rutinen **roots** för att bestämma nollställena. Vad är det för skillnad på **fzero** och **roots** vad gäller vilka nollställen (rötter) som kan bestämmas?

    Skriv en egen funktionsfil för Newtons metod med funktionen och derivatan som parametrar. Använd funktionsfilen för att beräkna nollställena. Använd **ginput** för att få startapproximationer. Jämför antal flyttalsoperationer för **fzero**, **roots** och din egen Newtonfunktion.

## Integraler

32. a) Beräkna integralen $\int_0^1 \ln(1 + x^2)\ dx$ exakt och med rutinerna **quad** och **quad8** i MATLAB. Jämför noggrannhet och antal operationer.

    b) Beräkna integralen approximativt med trapetsmetoden dvs med **trapz** i MATLAB. Välj olika steglängder $h$ och jämför resultatet med teorin som säger att felet ska vara proportionellt mot $h^2$.

## Ordinära differentialekvationer

33. Beräkna $y(4)$ där $y$ är lösningen till begynnelsevärdesproblemet $y' = e^{-y} + t,\ y(0) = 0$ genom att använda **ode45** i MATLAB. Räkna fram en approximation med Eulers framåtmetod och steg $h = 0.1$. Rita ut lösningen från **ode45** och approximationen från Eulers metod i samma diagram. Hur sort blir felet i $t = 4$?

34. För att kyla en kropp med temperatur $T$ $(^oC)$ används en fläkt. Enligt Newtons avsvalningslag gäller att temperaturminskningen är proportionell mot temperaturdifferensen mellan kroppen och den omgivande luftströmmen. Om luftens temperatur är $30\ ^oC$ får vi därför $T' = -k(T - 30)$, där $k$ är en avsvalningskonstant. Låt $k = 0.04$ och anta att kroppens temperatur är $100\ ^oC$ vid kylningens början. Bestäm kroppens temperatur efter 20 tidsenheter. Använd Eulers framåtmetod och några olika steglängder. Verifiera att felet är proportionellt mot $h$, som teorin säger. Jämför med exakta lösningen som är $T(t) = 30 + 70\, e^{-0.004\ t}$.

35. En viss fastspänd bladfjäder, som vi böjer ut och släpper, beskrivs av differentialekvationen $x'' = -x^3 - 0.05\ x',\ x(0) = 1,\ x'(0) = 0$. Om vi låter $y = (y_1,\ y_2) = (x,\ x')$ får vi standardformen för ett system med två komponeneter $y' = f(t,\ y),\ y(0) = c$ där $f$ och $c$ är vektorerna

    $$f = \begin{bmatrix} y_2 \\ -y_1^3 - 0.05\ y_2 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

    Använd **ode45** för att beräkna lösningen och rita ut den på lämpligt sätt.