Linnéuniversitetet

Linjär algebra för ingenjörer, 1MA133, 7,5 hp, HT23

Sofia Eriksson

Inlämingsuppgift Matlab (1,5 hp)

Uppgifterna görs i normalfallet i grupp om 2-3 personer
(I undantagsfall görs uppgiften individuellt eller max fyra personer. Fråga i så fall)

Inlämning: En pdf-fil som lämnas in i MyMoodle.
Filen produceras förslagsvis i Word eller liknande och exporteras sedan till pdf (alternativt används Latex om man är van vid det).

Redovisning: En kortfattat rapport. Format: pdf-fil.

Börja med rubrik och ALLA gruppmedlemmars namn.

Fortsätt med underrubriker för uppgifterna.
Från respektive uppgift behöver följande redovisas:

Del I (uppg 1,2,3) - KAN godkännas på plats under labbtillfälle. Om du inte godkänts vid labbtillfälle ska även Del I inkluderas i rapporten.

1: Figuren som 1(vi) producerar (bild med 3 plan, en linje samt en punkt).

2: Svar på 2(i) och 2(ix). Kod från 2(viii) samt figuren som 2(viii) producerar.

3: Kod från 3(v). Figuren som 3(v) producerar.

Del II (uppg 4,5) - Rapport behövs

4: Kod från 4(v). Tre figurer (samma figur sedd från 3 olika vinklar) enligt 4(vi).

5: Kod som utför 5(i-vi). Fyra figurer från 5(iii,iv,v,vi).

Spara Matlab-figurerna i lämpligt format, så att de kan infogas i ditt Word-dokument. När du klistrar in en bit kod i Word som du kopierat från en .m-fil, kan det hända att typsnittet automatiskt formateras som i Matlab. Om det inte händer automatiskt, använd gärna ett annat typsnitt/font (t.ex. `Courier`) eller en annan färg, så det blir tydligt vad som är vanlig rapporttext och vad som är inklistrad kod.
https://blogs.mathworks.com/community/2009/04/13/bring-along-your-syntax-highlighted-code/

Tips: Använd Matlabs hjälp. Skriv `help plot`*, o.s.v. i Matlabs kommandofönster om du inte vet hur ett kommando används. Testa kommandona i kommandofönstren tills du vet hur de fungerar (klistra inte in dem i m.-filer förrän du är nöjd).*

Diskutera gärna med lärare och studiekamrater om något är oklart eller om ni kör fast men KOPIERA INTE KODER från andra grupper och ge inte era koder till andra grupper.

Uppgifter, Del I

1. **Geometriska tolkningar**

   Betrakta ekvationssystemet

$$\begin{cases} x_1 + x_2 + 6x_3 = 10 & \text{(ekv. 1)} \\ x_1 + 4x_2 + 3x_3 = 7 & \text{(ekv. 2)} \\ 5x_1 - x_2 + 3x_3 = -10 & \text{(ekv. 3)} \end{cases}$$

i. Varje ekvation kan var för sig tolkas geometriskt som ett plan i 3D. Vi kan rita upp planet som den första ekvationen representerar. Detta göra på följande vis:

Välj ut fyra ”hörnpunkter” i xy-planet, exempelvis $(-10, -10)$, $(-10, 10)$, $(10, 10)$ och $(10, -10)$. Spara dess x-koordinater i en matris och dess y-koordinater i en annan matris. Lös sedan ut motsvarande z-koordinater med hjälp av den första ekvationen i ekvationssystemet ovan. Med hjälp av Matlabkommandot `fill3` kan vi rita upp en yta mellan dessa hörnpunkter. Skriv i Matlab:

```
>> X=[-10 -10 10 10];
>> Y=[-10 10 10 -10];
>> Z1=(10-X-Y)/6;
>> fill3(X,Y,Z1,'b','facealpha',0.4)
>> xlabel('x'); ylabel('y'); zlabel('z')
```

Att skriva `'b'` bestämmer att planet ska bli blått, och `'facealpha',0.4` bestämmer hur genomskinligt planet ska bli (0.4 innebär 40% ogenomskinligt).

ii. Gör detsamma för ekvation 2. Rita ut det andra planet i samma bild som det första planet, välj en annan färg så man kan skilja planen åt. För att det första planet inte ska försvinna behövs kommandot `hold on` innan uppritningen av andra planet.

iii. Löser du ekvationssystemet som bara består av ekvation 1 och ekvation 2 får du en parameterlösning (gör gärna detta för hand som övning). Lösningen kan du rita upp tillsammans med de två planen från uppgift 1(i-ii) med hjälp av

```
>> t=[1/7 3];    % Detta t har valts av estetiska skäl
>> plot3(11-7*t,-1+t,t,'--k',LineWidth=2)
```

För att titta på figuren från olika håll, välj i figurfönstrets toolbar: Tools → Rotade 3D. Hur tolkar du lösningen?

iv. Rita även ut det plan som svarar mot ekvation 3 (fortfarande i samma figur). Välj en ny färg så man kan skilja de 3 planen åt.

v. Skriv om ekvationssystemet på matrisform $A\mathbf{x} = \mathbf{b}$ för hand. Skriv in A och $\mathbf{b}$ i Matlab. Lös ekvationssystemet (se t.ex. s. 8 i KomIgångMedMatlab). Testa de tre sätten att lösa som presenterats (backslash, invers, `rref(M)`). Blir det samma svar?

vi. Spara svaret från 1(v.) i en matris `x`. Du kan tolka denna lösning till ekvationssystemet som en punkt i 3D. Rita upp denna punkt i samma figur som du tidigare ritade upp planen.

```
>> plot3(x(1),x(2),x(3),'k*',MarkerSize=10,LineWidth=2)
```

- -

2. **Anpassning av en cirkel till data**

Nedan ges koordinaterna för 4 punkter som ligger ungefär på en cirkel. Din uppgift är att hitta vilken cirkel som passar punkterna bäst (i minstakvadrat-mening).

x	-4	0	1	5
y	3	2	12	6

Kom ihåg att ekvationen för en cirkel är $(x-p)^2 + (y-q)^2 = r^2$, där r är cirkelns radie och där (p, q) anger cirkelns centrum i xy-planet. För att kunna använda minstakvadratmetoden skrivs ekvationen om till

$$c_1 x + c_2 y + c_3 = x^2 + y^2,$$

där $c_1 = 2p$, $c_2 = 2q$ och $c_3 = r^2 - p^2 - q^2$.

- -

i. Ställ upp det överbestämda ekvationssystem $A\mathbf{c} = \mathbf{b}$ som ska lösas för att hitta $\mathbf{c} = (c_1, c_2, c_3)^T$. *Obs! Du behöver inte lösa systemet, skriv bara upp det för hand.*

ii. Konstruera matriserna A och $\mathbf{b}$ (som du bestämt ovan) i Matlab. Matriser görs med hjälp av hak-paranteser.

iii. Lös det överbestämda ekvationssystemet $A\mathbf{c} = \mathbf{b}$ med Matlabs kommando \ för att få fram $\mathbf{c}$.

iv. Från $\mathbf{c} = (c_1, c_2, c_3)^T$ kan du hitta radien r och cirkelns centrum (p, q).

Obs, använd Matlabs notation för att få ut enskilda element ur matriser. D.v.s skriv *inte* av värdena i decimalform.

v. Skapa en vektor $\mathbf{v}$ med 101 element som startar i 0 och stegar upp till 2π (d.v.s. med steg $2\pi/100$). Du ska nu rita upp cirkeln $(x, y) = (p, q) + (r\cos(v), r\sin(v))$. Användbart kommando är `plot`.

vi. Fortsätt i samma figur och rita även in de 4 punkterna från tabellen ovan. (Användbart kommando: `hold on`). Välj själv vilka tecken (t.ex. stjärnor eller cirklar) som punkterna markeras med. Om du gjort allt rätt så ska punkterna ligga nästan på cirkeln.

vii. Gör figuren tolkningsbar med hjälp av kommandona `xlabel('x')` och `ylabel('y')`. För att det ska se ut som en cirkel och inte som en ellips måste du också se till att x- och y-axeln har samma skalning. Detta görs genom använda `axis equal`. Gör gärna figuren lite snygg också, genom att specificera linjetjocklek, markerstorlek och fontstorlek.

viii. När du är nöjd med punkt 2(ii-vii): Spara din kod i en .m-fil som du döper till något lämpligt. Din kod ska sedan köras genom att skriva filens namn i kommandofönstret.

När koden körs ska en figur med cirkeln och punkterna skapas, spara den i lämpligt filformat (förslagsvis eps, jpg eller pdf beroende på vad du lättast/snyggast kan infoga i din Wordfil/texfil).

Tips: Klistra in lite i taget från Kommandofönstret till .m-filen, testkör innan du klistrar in mer. (En bra tumregel är att överst i din kod starta med kommandot `clear`*; det rensar Workspace från gamla variabler så att du kan vara säker på att koden i din .m-fil fungerar självständigt.)*

ix. I punkten (2.iii) ovan löstes systemet $A\mathbf{c} = \mathbf{b}$ med kommandot \ (backslash). Beskriv hur du skulle ha gjort för att lösa systemet i minsta kvadratmening för hand. *Du behöver inte lösa systemet, bara kortfattat beskriva hur man gör.*

- -

Övn. *Övning inför Uppgift 3:*

Använd **b** från uppgift 2.
Låt MATLAB beräkna produkten `b*b`. Vad händer? Varför?
Testa sedan att beräkna `b.*b` istället. Vad är skillnaden? Vad gör `.*`?
(Kolla `help *` respektive `help .*` för att se om du gissat rätt. Testa även `b'*b`)

- -

3. **Anpassning av en cirkel till en större datamängd**

I MyMoodle hittar du en fil som heter `cirkel300.mat`. Ladda ner den och spara den, förslagsvis på samma ställe som du sparat .m-filen från uppgift 2.

i. .mat-filer öppnas inte på samma sätt som .m-filer. I Matlab (antingen i Command Window eller i Editor) kan du istället importera data från filen genom kommandot `load('cirkel300.mat')`, om du befinner dig i samma folder/mapp som filen. När du skrivit detta kommando ska två nya variabler, X och Y, dyka upp i Workspace.

ii. Rita upp de punkter som x- och y-koordinaterna i X och Y beskriver. De beskriver 300 punkter som befinner sig ungefär på en cirkel. Välj själv vilka tecken och färg du vill använda för att markera punkterna.

(PS. Om linjerna mellan punkterna ritas ut, kommer det se ut mer som ett virrvarr av trådar istället för som punkter på en cirkel.)

iii. Precis som i uppgift 2 (fast med data från 300 punkter istället för bara 4), konstruera matriserna A och $\mathbf{b}$ i Matlab som ingår i det motsvarande överbestämda systemet $A\mathbf{c} = \mathbf{b}$. Tips: Användbara kommandon: `.*` och `ones`.

iv. När A och $\mathbf{b}$ konstruerats, lös normalekvationerna med hjälp av backslash. Precis som tidigare får du ett svar $\mathbf{c}$, och precis som tidigare ska du använda $\mathbf{c}$ för att rita upp en cirkel i samma figur som punkterna. Om allt gjorts rätt ska punkterna och cirkeln matcha hyffsat. Använd `axis equal`, och gör figuren snygg precis som i uppgift 2.

v. När allt fungerar, gör en ny .m-fil som utför uppgift 3(i-iv).

- -

Uppgifter, Del II

4. **Rita upp ett plan i 3D.**

   Låt $\vec{u} = (3, 3, -1)$ och $\vec{v} = (2, 4, -1)$ vara två vektorer i rummet.

i. Bestäm en vektor $\vec{w}$ som är ortogonal mot både $\vec{u}$ och $\vec{v}$. Ett smidigt sätt är att beräkna vektorprodukten (också kallad kryssprodukten) mellan $\vec{u}$ och $\vec{v}$. Gör detta antingen för hand och/eller m.h.a. Matlabs kommando `cross`.

ii. För hand: Bestäm sedan ekvationen för det plan Π som spänns upp av $\vec{u}$ och $\vec{v}$ och som innehåller origo. Minns att detta plan har $\vec{w}$ som normal.

   Därefter: Låt $P_1 = (6, 6, z_1)$, $P_2 = (6, -6, z_2)$, $P_3 = (-6, -6, z_3)$ och $P_4 = (-6, 6, z_4)$ vara fyra punkter i rummet. Bestäm de fyra z-koordinaterna $z_{1,2,3,4}$ så att alla fyra punkterna ligger i planet Π.

iii. Betrakta de fyra punkterna $P_{1,2,3,4}$ du undersökte i uppgift 4(ii). Spara punkternas x-koordinater i en vektor X, y-koordinaterna i en vektor Y och z-koordinaterna i en vektor $Z = [z_1, z_2, z_3, z_4]$ i MATLAB. Dessa X-, Y- och Z-koordinater beskriver tillsammans hörnen av en yta (en delmängd av planet Π) i 3D. Rita upp denna yta med hjälp av kommandot `fill3`, med blå färg.

   Använd `help fill3` för att ta reda på hur kommandot används. Med den extra egenskapen `'facealpha',0.4` görs ytan dessutom halv-genomskinlig.

iv. Låt en representant av $\vec{u}$ från uppgift 4(i) starta i origo. Rita upp $\vec{u}$ med hjälp av funktionen `RitaVektor3D` som du hittar i MyMoodle, utan att planet från 4(iii) försvinner. Skriv `help RitaVektor3D` för att se hur den används.

   Fortsätt i samma figur som tidigare, rita upp de två andra vektorerna, $\vec{v}$ och $\vec{w}$, också med start i origo. Dessa två vektorer ska ha andra färger än $\vec{u}$.

v. När allt fungerar, gör en .m-fil med alla kommandon från 4(iii-iv). Koden ska alltså ska producera en figur med en blå yta (delmängd av planet Π), två vektorer som spänner upp detta plan Π samt en normal till planet.

   Gör figuren tolkningsbar och fin genom att använda kommandona `xlabel('x')`, `ylabel('y')` och `zlabel('z')` följt av kommandona `grid on` och `box on` och `set(gca,'fontsize',16)` och (viktigt i denna uppgift!) `axis equal`.

vi. Du kan vrida runt på din figur med hjälp av symbolen ”Rotate 3D”  som finns i menyn i figurfönstret (eller under Tools → Rotade 3D).

   Spara en figur där du vridit vyn så du själv tycker du får en bra ”3D-känsla”.

   Ett annat sätt att ändra infallsvinkel på, är att använda kommandot `view`. Testa t.ex. att skriva `view([1,0,0])` eller `view([0,1,0])` eller `view([0,0,1])` och se vad som händer med din figur.

   Använd nu `view` för att se figuren dels från $\vec{u}$:s håll och dels från $\vec{w}$:s håll. Spara även dessa två figurer.

   Totalt ska du alltså spara 3 figurer.

- -

5. **Linjär avbildning av ett träd**

i. Ladda ner filen `lnu.mat`. Läs in dess innehåll med hjälp av kommandot `load`. Du ska nu få in en 2×37233-matris kallad `xy` i WorkSpace.

ii. Den översta raden i `xy` innehåller 37233 x-koordinater, den andra raden innehåller motsvarande y-koordinater. Rita upp de 37233 punkterna m.h.a. kommandot `plot`. Här är det bäst att välja markören ”point” `'.'` annars blir det inte så snyggt.

```
>> plot(xy(1,:),xy(2,:),'.')
```

Använd även kommandot `axis equal`.

iii. `xy` kan ses som en lista med 37233 punkter eller ortsvektorer. I kursen lär vi oss att för att göra en *linjär avbildning* av en vektor multiplicerar man vektorn från vänster med matrisrepresentationen A för önskad avbildning.

Multipliceras hela matrisen `xy` med matrisen A från vänster avbildas alla de 37233 ortsvektorerna på en gång.

T.ex. ges matrisen för spegling i x-axeln av matrisen

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Beräkna avbildningen av `xy` speglad i x-axeln, genom att multiplicera `xy` med S. Spara resultatet (en 2×37233-matris). Rita sedan upp denna resulterande spegelbild i samma figur (Figur 1) som originalbilden.

iv. Ta reda på vad matrisen för rotation är. Gör sedan en ny figur, Figur 2, (kommandot `figure` öppnar ett nytt figurfönster) som innehåller originalbilden tillsammans med en bild som roterats $2\pi/3$ moturs (prova gärna andra vinklar också).

v. Sammansatt linjär avbildning: Gör en ny figur (Figur 3), rita upp den sammansatta avbildningen ”spegling i x-axeln följt av $2\pi/3$ moturs rotation” med lila färg (magenta), samt den sammansatta avbildningen ”$2\pi/3$ moturs rotation följt av spegling i x-axeln” med turkos färg (cyan).

vi. Testa ytterligare någon avbildning, exempelvis skalning, projektion, skevning eller skjuvning (exempel finns i boken, kap. 7). Är det svårt att bestämma sig, testa

$$M = \frac{1}{2}\begin{bmatrix} 1+k & k-1 \\ k-1 & 1+k \end{bmatrix}$$

med eget val av värdet k. M ger en stretchad bild med skalfaktorn k i $45°$ vinkel (längs linjen y=x). Gör ett nytt figurfönster (Figur 4), rita in originalbilden tillsammans med avbildningen. Glöm inte `axis equal`.