MAT 1110 V-06: Løsningsforslag til Oblig 1

Oppgave 1: a) Antall sykler i stativet X *rett før* påfyllingen i måned $n+1$ er lik 40% av antall sykler i X måneden før, pluss 10% av antall sykler i Y måneden før, pluss 10% av antall sykler i Z måneden før, pluss 30% av antall sykler i U måneden før, dvs. $0.4x_n + 0.1y_n + 0.1z_n + 0.3u_n$. Ganger vi dette med 1.15 for å fange opp påfyllet, får vi

$$x_{n+1} = 0.46x_n + 0.115y_n + 0.115z_n + 0.345u_n$$

De andre ligningene fremkommer på tilsvarende måte.

b) Det er mange måter å organisere en slik m-fil på. Følgende forslag ligger tett opp til oppgaveteksten:

```
function C=bysykler(a,b,c,d) %ber programmet returnere C når det
                             %får input a,b,c,d
x=[a];                     %med disse linjene forteller vi
y=[b];                     %MATLAB at det i utgangspunktet er
z=[c];                     %a sykler i X, b i Y osv.
u=[d];
for n=1:49                 %starter for-løkke
    x(n+1)=.46*x(n)+.115*y(n)+.115*z(n)+.345*u(n);
    y(n+1)=.23*x(n)+.46*y(n)+.23*z(n)+.23*u(n);
    z(n+1)=.23*x(n)+.23*y(n)+.2875*z(n)+.23*u(n);
    u(n+1)=.115*x(n)+.23*y(n)+.2875*z(n)+.23*u(n);
end                         %avslutter for-løkke
C=[x;y;z;u];                %lager en matrise med x som første rad,
                            %y som andre osv. Rutinen returnerer
                            %denne matrisen (se første linje)
```

Man kan også lage en variant av programmet som returnerer x, y, z og u som vektorer:

```
function [x,y,z,u]=bysykler(a,b,c,d)
x=[a];
y=[b];
z=[c];
u=[d];
for n=1:49
    x(n+1)=.46*x(n)+.115*y(n)+.115*z(n)+.345*u(n);
    y(n+1)=.23*x(n)+.46*y(n)+.23*z(n)+.23*u(n);
    z(n+1)=.23*x(n)+.23*y(n)+.2875*z(n)+.23*u(n);
    u(n+1)=.115*x(n)+.23*y(n)+.2875*z(n)+.23*u(n);
end
```

c) Antar at vi har laget den første av m-filene ovenfor. For å tegne x, y, z, u i samme koordinatssystem kan vi da gi kommandoene:

```
>>C=bysykler(100,100,100,100); %kjører m-filen med angitt input
>> plot(C(1,:))        %plotter x (første rad i C)
```

```
>> hold on              %legger plottene i samme figur
>> plot(C(2,:),'r')     %plotter y i rødt
>> plot(C(3,:),'g')     %plotter z i grønt
>> plot(C(4,:),'y')     %plotter u i gult
```

d) Planen er å erstatte C med en ny matrise D der hvert element i C er dividert med 1.0071^{n-1}, og så tegne radene i D. Vi kan skrive:

```
>> a=1.0071.^[0:49]  %lager en vektor med komponenter 1.0071^(n-1).
>> E=[a;a;a;a];      %lager hjelpematrise E med alle linjer lik a
>> D=C./E            %deler komponentene i C på komponentene i E
>> figure(2)         %åpner et nytt plottevindu
>> plot(D(1,:))      %plotter følgen x_{n}/1.0071^{n-1}
>> hold on           %legger plottene i samme figur
>> plot(D(2,:),'r')  %plotter følgen y_{n}/1.0071^{n-1} i rødt
>> plot(D(3,:),'g')  %plotter følgen z_{n}/1.0071^{n-1} i grønt
>> plot(D(4,:),'y')  %plotter følgen u_{n}/1.0071^{n-1} i gult
```

(De som er smarte og har kikket litt fremover i teksten, vil kanskje lagre denne rutinen som en m-fil for å kunne bruke den på nytt i neste punkt!) Plottene indikerer at følgene konvergerer mot konstante verdier.

e) Kjør rutinen ovenfor på nytt. Det eneste du behøver å endre er input-verdiene for m-filen `bysykler`. Du må nå skrive `>>C= bysykler(200,0,0,200)`. Figurene viser samme oppførsel som i stad. Ser du nærmere etter, vil du se at den prosentvise fordelingen av sykler mellom X, Y, Z og U konvergerer mot det samme som i punkt d). Det samme gjelder for andre (naturlige) startverdier.

Kommentarer til oppgave 1: Oraklene melder at oppgave 1 stort sett var bra løst, men det er et par punkter å kommentere på. I punkt 1b) er det unødvendig å skrive ut følgene $x_1, x_2, \ldots, x_n$ osv. — at MATLAB skal "returnere" disse følgene, betyr bare at programmet skal regne dem ut og lagre dem slik at de kan brukes i det videre arbeidet (det er selvfølgelig ikke galt å skrive dem ut, men det blir upraktisk og uoversiktelig når datamengdene blir store). Et viktigere punkt er at man bør "navnsette" outputen fra en m-fil på en fornuftig måte. Det er dumt bare å skrive

```
>> bysykler(100,100,100,100)
```

man bør skrive

```
>> C=bysykler(100,100,100,100)
```

eller

```
>> [x,y,z,u]=bysykler(100,100,100,100)
```

(avhengig av strukturen på output). Husk at selv om dere skriver

```
function C=bysykler(a,b,c,d)
```

inni m-filen, så gjelder ikke denne tilordningen i det vanlige MATLAB-vinduet (variabeltilordningene i en m-fil er *lokale* og gjelder ikke utenfor filen selv).

Oppgave 2 a) Vi ganger ut:

$$\begin{bmatrix} 4 & 2 & -3 \\ -2 & 6 & -8 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ -3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}$$

Dette viser at $\begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}$ er en egenvektor med egenverdi 3.

b) Per definisjon av egenverdi har vi $A\mathbf{v} = \lambda\mathbf{v}$, dvs. $A^1\mathbf{v} = \lambda^1\mathbf{v}$. Ganger vi begge sider av denne likheten med A, får vi $A^2\mathbf{v} = A(\lambda\mathbf{v}) = \lambda A\mathbf{v} = \lambda^2\mathbf{v}$. Ganger vi begge sider av denne likheten med A, får vi $A^3\mathbf{v} = A(\lambda^2\mathbf{v}) = \lambda^2 A\mathbf{v} = \lambda^3\mathbf{v}$. Ved å fortsette på denne måten, får vi $A^n = \lambda^n\mathbf{v}$ for alle naturlige tall n. (Jeg synes det er unødvendig å føre et formelt induksjonsbevis for noe som er så enkelt å gjennomskue.)

c) Vi har

$$\begin{aligned} A^n\mathbf{w} &= A^n(c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k) \\ &= c_1A^n\mathbf{v}_1 + c_2A^n\mathbf{v}_2 + \cdots + c_kA^n\mathbf{v}_k \\ &= c_1\lambda_1^n\mathbf{v}_1 + c_2\lambda_2^n\mathbf{v}_2 + \cdots + c_k\lambda_k^n\mathbf{v}_k \end{aligned}$$

der vi har brukt resultatet i punkt b).

d) Ifølge c) er

$$\frac{A^n\mathbf{w}}{\lambda_1^n} = c_1\mathbf{v}_1 + c_2\frac{\lambda_2^n}{\lambda_1^n}\mathbf{v}_2 + \cdots + c_k\frac{\lambda_k^n}{\lambda_1^n}\mathbf{v}_k$$

Siden λ_1 har større tallverdi enn de andre egenverdiene, vil $\frac{\lambda_2^n}{\lambda_1^n}, \ldots, \frac{\lambda_k^n}{\lambda_1^n}$ gå mot null når $n \to \infty$. Dermed er

$$\lim_{n\to\infty} \frac{A^n\mathbf{w}}{\lambda_1^n} = c_1\mathbf{v}_1$$

Kommentar til oppgave 2: Ifølge oraklene var det litt mer "blanda drops" på denne oppgaven. Det er mange som ikke helt forstår forskjellen på matriser, vektorer og tall. Noen har for eksempel løst punkt b) ved å forkorte bort $\mathbf{v}$ i formelen $A\mathbf{v} = \lambda\mathbf{v}$ og dermed fått $A = \lambda$. Dette er åpenbart meningsløst (hvordan kan matrisen A være lik tallet λ?). Generelt kan man ikke forkorte bort vektorer — når man forkorter, deler man egentlig med samme størrelse på begge sider av likhetstegnet, og det er ikke mulig å dele med en vektor.

En annen ting som skiller matriser fra tall, er at matrisemultiplikasjon ikke er kommutativ. Skal man gange begge sider av en ligning med en matrise A (det er lov!), må man derfor passe på å gange på samme måte på begge sider; man kan ikke høyremultiplisere på den ene siden og venstremultiplisere på den andre! Hvis $B = C$, kan man altså slutte at $AB = AC$ og at $BA = CA$ (forutsatt at matrisedimensjonene stemmer), men ikke at AB=CA. Det er også viktig å passe på dimensjonene når man multipliserer en matrise og en vektor. Er $\mathbf{v}$ en søylevektor med n-komponenter og A en $m \times n$-matrise, så gir $A\mathbf{v}$ mening, mens $\mathbf{v}A$ ikke gjør det!

Generelt var det mye dårlig føring på oppgave 2. Husk at å lære seg å formidle matematikk er en viktig del av en enhver matematikkutdannelse — det er få jobber som består i å løse matematikkproblemer på en måte man bare forstår selv!

Oppgave 3 a) At $w_{n+1} = Aw_n$, er bare en omskrivning av ligningssystemet i 1a). Bruker vi denne sammenhengen for $n = 1, 2, 3$ osv, får vi $w_2 = Aw_1$, $w_3 = Aw_2 = A(Aw_1) = A^2w_1$, $w_4 = Aw_3 = A(A^2w_1) = A^3w_1$. Ved å fortsette på samme måten, får vi $w_n = A^{n-1}w_1$ for alle naturlige tall n (før et skikkelig induksjonsbevis hvis du har lyst!)

b) Legg A inn i MATLAB og gi kommandoen `>> [V,D]=eig(A)`. MATLAB svarer med

```
>> [V,D]=eig(A)

V =
   -0.4925   -0.8165   -0.6528   -0.7071
   -0.5878    0.4082    0.0259   -0.0000
   -0.4811    0.0000   -0.1345    0.0000
   -0.4248    0.4082    0.7451    0.7071

D =
    1.0071         0         0         0
         0    0.2300         0         0
         0         0    0.0854         0
         0         0         0    0.1150
```

Søylene i V gir egenvektorer, mens tallene på diagonalen i D er de tilsvarende egenverdiene. Vi ser at den første egenverdien (1.0071) har størst tallverdi.

c) Vi bruker at søylene i V danner en basis hvis og bare hvis den reduserte trappeformen er identitetsmatrisen I. Kommandoen `>> rref(V)` gir oss

```
>> rref(V)

ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

så egenvektorene danner en basis.

d) Å skrive $\mathbf{w}_1$ som en lineærkombinasjon av basisvektorene er det samme som å løse ligningssystemet $V\mathbf{x} = \mathbf{w}_1$. Dette kan gjøres på flere måter, men den letteste er kanskje å bruke disse kommandoene:

```
>> w1=[100;100;100;100];

>> x=V\w1
```

```
x =
 -201.1917
  -43.2263
  -23.9200
   70.7107
```

Vi har dermed at lineærkombinasjonen

$$\mathbf{w}_1 = x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + x_3\mathbf{v}_3 + x_4\mathbf{v}_4$$

er gitt ved:

$$\begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} = -201.1917 \begin{bmatrix} -0.4925 \\ -0.5878 \\ -0.4811 \\ -0.4248 \end{bmatrix} - 43.2263 \begin{bmatrix} -0.8165 \\ 0.4082 \\ 0 \\ 0.4082 \end{bmatrix}$$
$$- 23.9200 \begin{bmatrix} -0.6528 \\ 0.0259 \\ -0.1345 \\ 0.7451 \end{bmatrix} + 70.7107 \begin{bmatrix} -0.7071 \\ 0 \\ 0 \\ 7071 \end{bmatrix}$$

Resultatet fra 2d) sier nå at

$$\lim_{n\to\infty} \frac{A^{n-1}\mathbf{w}}{\lambda_1^{n-1}} = x_1\mathbf{v}_1 = -201.1917 \begin{bmatrix} -0.4925 \\ -0.5878 \\ -0.4811 \\ -0.4248 \end{bmatrix} = \begin{bmatrix} 99.0869 \\ 118.2605 \\ 96.7933 \\ 85.4662 \end{bmatrix}$$

(At vi har $n-1$ som eksponent — og ikke n — spiller selvfølgelig ingen rolle siden $n-1$ går mot uendelig når n gjør det.)

Legg merke til poenget (som lett drukner i alle tallene): I det lange løp er veksten i antall sykler bestemt av den største egenverdien λ_1, og den prosentvise fordelingen av sykler mellom de forskjellige stativene er bestemt av den tilhørende egenvektoren $\mathbf{v}_1$. (Hvis du vil ha dette illustrert grafisk, kan du plotte følgene $\{99.0869 \cdot 1.0071^{n-1}\}$, $\{118.2605 \cdot 1.0071^{n-1}\}$, $\{96.7933 \cdot 1.0071^{n-1}\}$ og $\{85.4662 \cdot 1.0071^{n-1}\}$ i samme koordinatsystem som du brukte i 1c.) Starter vi med en annen verdi for $\mathbf{w}_1$, vil de samme beregningene gjenta seg. Den eneste forskjellen er at vi ville få en annen x_1 (dvs. at det totale antall sykler vil være annerledes, men vekstraten og fordelingen mellom stativene den samme). I prinsippet kan man gjøre de samme beregningene uansett hvor mange sykler og stativer man har, og da blir egenverdiberegninger meget tidsbesparende (ifølge

http://www.oslosurf.com/innhold/00000185.shtml

tar Oslo sikte på å ha 2500 sykler i 100 stativer, men den matrisen får noen andre taste inn!)

Kommentar til oppgave 3: MATLAB-delen av denne oppgaven gikk rimelig greit, men det er få som hadde en god begrunnelse på punkt d). Det er litt synd, for det var dette punktet man egentlig skulle lære noe nytt av!