Projektuppgift 1: Linjära ekvationssystem

Uppgiften ska lösas i Matlab. I Canvas kan ni ladda ner filen
`IX1303_Projekt1_LinearEquations.m`
Filen innehåller en ofullständig kod som ni ska fylla när ni gör uppgiften. I denna fil finns även de frågor ni ska besvara (samma frågor som nedan). Svaren på frågorna ska ni skriva in i filen. När ni är klara med uppgiften ska den uppdaterade filen laddas upp i Canvas för rättning.

PROBLEM 1: Numeriska beräkningar av determinanter.

Numeriska beräkningar i Matlab sker i allmänhet med approximationer av de reella talen. Som exempel kan 1/3 approximeras med 0.3333333333333333. Här ska vi studera de två matriserna M_1 och M_1 som båda är singulära eftersom kolumnvektorerna är parallella. Därmed *borde* determinanten av de två matriserna vara noll. Använd funktionen *det* för att beräkna determinanterna för matriserna M_1 och M_1.

$$M_1 = \begin{bmatrix} 1 & 4 \\ 2 & 8 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 4/3 \\ 1.1 & 4.4/3 \end{bmatrix}$$

(a) Varför blir inte båda determinanterna exakt noll?
(b) Använd beräkningen för determinanterna ovan får att uppskatta hur stora fel kan vi förvänta oss vid beräkning av determinanter?

PROBLEM 2: Matrisinverser

Skapa en *första* kolumnvektor, m_1, som baserat på ditt födelsedatum så att m_1 = ["år (bara 2 siffror)"; "månad"; "dag"]. T.ex. om du är född 15 juni 2004 blir vektorn: m_1=[4; 6; 15].
Skapa en *andra* kolumnvektor, m_2, som är linjärt oberoende av m_1. Här får du själv välja komponenternas värden.
Skapa en *tredje* vektor m_3 som kryssprodukten av m_1 och m_2. Här kan du använda funktionen *cross*.
Slutligen skapa en matris, M, vars kolumner består av m_1, m_2 och m_3.

(a) Vad är vinkeln mellan m_1 och m_3?
(b) Vad är vinkeln mellan m_2 och m_3?
(c) Är de tre vektorerna linjärt oberoende?
(d) Vad är determinanten av matrisen M? Här kan du använda funktionen *det*.
(e) Vilket påstående i "*invertable matrix theorem*", från Lay's bok, ska man använda tillsammans med svaret i c) för att avgöra om matrisen M är inverterbar?
(f) Använd Matlab's funktion *inv* för att beräkna inversen av matrisen M. Är svaret från *inv* rimligt?

PROBLEM 3: Matrisers rank och nollrum

Skapa en matris M_4 vars kolumnvektorer är m_1, m_2, m_3 och m_4. Här är m_1 och m_2 givna (se nedan), men m_3 och m_4 får ni skapa genom att själva välja två olika som linjärkombinationer av m_1 och m_2.

$$m_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, m_2 = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

(a) Vad har matrisen M_4 för rang? Använd funktionen *rank*.
(b) Hur många pivotkolumner och hur många fria variabler har M_4? Notera att detta är relaterat till matrisens rang.
(c) Vad är nollrummet till M_4? För detta kan du använda funktionen *null*. Notera att svaret måste skrivas på ett matematiskt korrekt sätt, så att det beskriver ett vektorrum. Svaret kan skrivas som en mening, eller med matematisk notation. Talen i vektorerna kan avrundas till att ha fyra decimaler.

PROBLEM 4: Ekvationssystem med många obekanta

När man löser linjära ekvationssystem på formen $A\mathbf{x} = \mathbf{b}$ med många obekanta är det viktigt att man använder en effektiv metod. Här kommer vi att lösa ekvationssystem med upp till ett par tusen obekanta, vilket inte är jättestort, men tillräckligt stort för att man ska kunna se vilka metoder som är bra och dåliga.

Er uppgift är att lösa minst fyra ekvationssystem med mellan 10 och 10 000 obekanta. Ni ska mäta tiden det tar att lösa varje system och generera grafer som visar denna tid som funktion av antalet obekanta. För varje ekvationssystem ska ni:

I. Skapa en matris, A, full med slumptal. För att matrisen ska bli inverterbar använder vi oss av teorem 11 från sid 167 i Lay upplaga 6 (från kapitlet om Leontief modellen). Därmed ska vår matris ska vara på formen $A = I - C$, där I är identitetsmatrisen (se funktionen *eye*) och C ska innehålla slumptal. En funktion som genererar slumptal är funktionen *rand*, som skapar en matris av slumptal mellan 0 och 1. För att teoremet ska vara uppfyllt ska a summan av värdena i varje kolumn i C vara mindre än 1, samt att C inte får innehålla några negativa tal. För en $n \times n$ matris, kan vi skriva $C = kR$, där $R =$ *rand(n)* och $k = 0.9/n$.
II. Använd *rand* för att generera en vektor $\mathbf{b}$ full med slumptal.
III. Lös ekvationssystemet $A\mathbf{x} = \mathbf{b}$ med hjälp av funktionen *mldivide*, som kan köras med kortkommandot ”\”. Denna metod använder både matrisen och högerledet för att hitta lösningar, på ett sätt som påminner om radreducering (Gauss-eliminering). För att mäta tiden det tar för datorn att lösa ekvationen kan man använda Matlab’s inbyggda tidtagningssystem. Tidtagning startas med funktionen *tic*, därefter löser du ekvationen, och till sist stänger ni av tiden med funktionen *toc*. Notera att om ni skriver `T(i)=toc` så sparas tiden i element `i` av vektorn `T` och därmed kan ni sedan använda värdet för att rita en graf med de tider ni mätt upp.

IV. Lös ekvationssystemet $A\mathbf{x} = \mathbf{b}$ med hjälp av inversen till A. Här kan ni använda funktionen *inv*. Även här ska ni använda *tic* och *toc* för att mäta tiden det tar att lösa ekvationssystemet.

V. Ni ska kontrollera att resultaten från *mldivide* och *inv* är desamma. Notera att när en dator räknar med reella tal så räknar den inte exakt. Matlab har en upplösning på ungefär 16 decimaler, så 17:e decimalen kan bli fel! Därmed kommer resultaten inte vara exakt lika. Som ett mått på skillnaden mellan de två lösningarna kan man första skapa differensen mellan vektorerna. Om du nu har 3000 värden är det svårt att gå igenom alla, så istället kan vi ta fram längden på differens-vektor. Längden på en vektor kan man beräkna med funktionen *norm*.

När ni löst de fyra olika ekvationssystem ska ni skapa två grafer som visar era resultat. I den första grafen ska ni rita antalet obekanta på x-axeln och tiden det tog att lösa ekvationssystemen på y-axeln. Ni ska rita resultaten från *mldivide* och *inv* i samma graf. I den andra grafen ska ni rita samma sak, men här ska ni skala om både x-axeln och y-axeln till log-skala genom att lägga till raderna:

```
set(gca,'xscale','log')
set(gca,'yscale','log')
```

Nu återstår bara att analysera resultaten och svara på följande frågor.

(a) Antag att du ska lösa ett problem med tio eller färre obekanta en eller ett par gånger. Hur väljer du metod? Är det viktigt att välja rätt metod?

(b) Antag att du ska lösa ett problem med tio obekanta 1000 000 gånger. Hur väljer du metod? Är det viktigt att välja rätt metod?

(c) Antag att du ska lösa ett problem med 3000 obekanta en eller ett par gånger. Hur väljer du metod? Är det viktigt att välja rätt metod?

(d) Kör om alla räkningar tre gånger. Varför får du olika resultat varje gång du kör programmet?

(e) Uppskatta den relativa skillnaden i beräkningstid mellan de två metoderna för 3000 obekanta?