TMV166 Linjär algebra M

VECKOPM 2 — LINJÄRA EKVATIONSSYSTEM

Kapitel 1. Linjära ekvationer i linjär algebra

Innehåll: I kursen *Matematisk analys i en variabel* såg vi att man kan skriva ett system av första ordningens differentialekvationer som ett linjärt ekvationssystem. Detsamma gäller för otaliga andra frågeställningar. I avsnitt **1.1** beskrivs därför linjära ekvationssystem från grunden, och en allmän procedur för att lösa sådana system introduceras. Denna metod, ofta kallad *Gausselimination*, kan också visa att systemet inte har några lösningar alls, eller att det har oändligt många lösningar.

Tidigare har du sett att en linjär ekvation med tre obekanta kan uppfattas som ekvationen för ett plan. Lösningen till ett ekvationssystem med tre obekanta kan därmed ses som skärningen mellan plan. Skriver vi ekvationssystemet med utvidgad koefficientmatris (totalmatris) är det alltså raderna i denna matris vi har i fokus då vi tänker på detta sätt.

I avsnitt **1.3** har vi istället totalmatrisens kolonner i fokus och ser lösningen som ett samband mellan dessa kolonnvektorer. Detta synsätt är centralt i en del tillämpningar, t.ex. inom mekanik och hållfasthetslära. Viktiga begrepp är *linjärkombination* och *linjärt hölje*.

I avsnitt **1.4** införs matrisbeteckningen $A\boldsymbol{x} = \boldsymbol{b}$ för ekvationssystem. Här har vi fortfarande kolonn-vektorerna i fokus men koefficientmatrisen A ger oss möjlighet att enkelt tala om alla kolonnerna samtidigt. Satserna 3 och 4 är centrala.

Avsnitt **1.5** är i viss mån repetition av inslag i Adams kapitel 10, men samtidigt ger det en ny syn på det du redan kan. I avsnitt **1.6** tillämpas ekvationssystem på tre områden vi inte tidigare behandlat. Det viktiga här är att se hur man kan systematisera resonemang genom att arbeta med vektorer och matriser istället för enskilda termer. Se t.ex hur en kemisk molekyl kan beskrivas med en vektor och att kemisk jämvikt därför kan tecknas med en vektorekvation.

I avsnitt **1.7** behandlas två begrepp *linjärt beroende* och *linjärt oberoende* och deras samband med lösningar till ekvationssystem. Här är det också koefficientmatrisens kolonnvektorer som är i fokus. De två begreppen är mycket viktiga för fortsättningen av kursen. Tre vektorer i rummet är linjärt beroende om de ligger i ett plan, annars är de linjärt oberoende. Fler än tre vektorer i rummet är alltid linjärt beroende, det finns ett linjärt samband mellan dem. Dessa tankar utvecklas i satserna 7, 8 och 9.

I **1.8** och **1.9** behandlas *linjära avbildningar*. Detta är en utvidgning av en idé du mött ganska tidigt i skolan. Om du har en ekvation som $x^2+3x = 5$ kan du dels se den bara som en ekvation, du vill veta vad x är, det är allt. Men du kan också se uttrycket x^2+3x som ett funktionsuttryck. Du har en funktion $f(x) = x^2 + 3x$ och vill veta för vilket värde på x som $f(x) = 5$. Samma ekvation men ett annat sätt att tänka om den, Du kan ställa helt nya frågor om ekvationen. På samma sätt är $T(\boldsymbol{x}) = A\boldsymbol{x}$ en funktion och vi kan undersöka denna funktions egenskaper. Vad har den för definitionsmängd och värdemängd? Vilka $\boldsymbol{x}$ avbildas på ett visst $\boldsymbol{y}$? Hur hänger funktionens egenskaper samman med matrisen? Dessa avsnitt lägger grunden för resonemang och tankar som återkommer genom hela kursen.

Date: 2018–01–18, Stig Larsson, Matematiska vetenskaper, Chalmers tekniska högskola.

Rekommenderade uppgifter. (PP är förkortning av Practice problems. Här menas att du bör inleda med att göra alla dessa. Du hittar dem direkt före övningarna till respektive avsnitt, lösningar finns i slutet av avsnittet. Uppgifter i fet stil demonstreras av lärare.)

Avsnitt	Godkändnivå		Överbetygsnivå
	Instuderingsuppgifter	Träningsuppgifter	
1.1	PP, 3, 7, 13, 15, ***18***	14, 24, 27, 31	
1.2	PP, 1, 3, 7	***11***, 14, 16, 21, 25, 26, 33	
1.3	PP, 9, 11, ***13***, 17	23, 24, 29, 30	
1.4	PP, 1, 3, 11	16, 23, 24, 26	
1.5	PP, 3, 6, ***11***	21, 23, 24, 33	
1.6		3, 7, 13	
1.7	PP, 5, 7	***9***, 13, 21, 22	30, 33–38
1.8	PP, 3, 11	18, 21, 22, 33	31
1.9	PP, 1, 3, 6, 17, 25, 27	7, ***11***, 23, 24	31, 32, 34, 35

Datorövningar på följande sidor.

TMV166 Linjär algebra för M

Datorövningar vecka 2: Matriser och linjära ekvationssystem

Allmänt

Meningen med dessa datorövningar är att ge grundläggande färdigheter i användning av MATLAB för beräkningar inom linjär algebra. Det finns många andra programvaror man kan använda för detta, men M-teknologer förväntas kunna MATLAB.

Det rekommenderas att ni sparar er kod i en fil för varje uppgift, så att ni kan repetera inför tentamen samt eventuellt återanvända koden senare.

Hjälpsystemet i MATLAB är bra. Använd `doc` eller `help`. (`doc help` eller `doc doc` för hjälp om hjälp!). Ett annat tips är att Googla felmeddelandet. Sannolikt har andra haft samma problem och lösningen hittas lätt.

Mål

- Repetera grundläggande MATLAB-hantering
- Lösa linjära ekvationssystem

Skapa matriser

Matriser och vektorer skapas i MATLAB med hjälp av hak-parenteser. En vektor i $\mathbb{R}^n$ tolkas alltid som en $n \times 1$-matris. Elementen på en rad avskiljs av antingen kommatecken eller mellanslag, och rader avskiljs av semikolon.

Exempelvis kan matrisen $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ och vektorn $x = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$ skapas via

```
>> A = [1 2 3; 4, 5, 6]
A =
     1     2     3
     4     5     6

>> x = [7; 8; 9]
x =
     7
     8
     9
```

Uppgifter av C.-H. Fant och L. Falk, uppdaterade av T. Stillfjord

Med kommandot `size` kan man se dimensionerna av en matris:

```
>> size(A)
ans =
     2     3
>> size(x)
ans =
     3     1
```

Notera hur MATLAB tolkar x som en 3×1-matris snarare än en vektor i $\mathbb{R}^3$. Man kan även använda `size(A,1)` för att få antalet rader, och `size(A, 2)` för att få antalet kolonner. I vektor-fallet så ger `length(x)` antalet komponenter (dvs. rader).

Matris-vektor-multiplikation görs via *:

```
>> A*x
ans =
    50
   122
```

Kontrollera att det stämmer!

Det går även att addera, subtrahera och multiplicera matriser med varandra om de har rätt dimensioner. Vi återkommer till det nästa vecka.

För att skapa större matriser kan man sätta ihop dem från mindre matriser. Då anger ett komma eller ett mellanslag horisontell separation och ett semikolon vertikal separation. Exempel:

```
>> A = [1 2; 3 4];
>> B = [5; 6];
>> C = [7 8];
>> D = [9];
>>
>> M = [A, B; C, D]
M =
   1     2     5
   3     4     6
   7     8     9
```

Detta kallas block-notation och används också utanför MATLAB; M är en 3×3 block-matris, uppbyggd av blocken A (2×2), B (2×1), C (1×2) och D (1×1). Blockens dimensioner måste passa ihop för att detta ska vara väldefinierat: Alla blocken på en rad måste ha samma antal rader och alla blocken i en kolonn måste ha samma antal kolonner. Testa t.ex. `[A, C]` och studera felmeddelandet från MATLAB.

Ändra matriser

För att komma åt elementet på rad r och kolonn k i matrisen A använder man `A(r, k)`. Hela rad r får man via `A(r,:)`, och hela kolonn k via `A(:,k)`, dvs. kolon betyder "alla index". Vill man välja ut de element som finns på raderna $r_1, r_2, \ldots, r_n$ och kolonnerna $k_1, k_2, \ldots, k_m$ så kan man skicka in två vektorer `r` och `k` med dessa index. Vill man t.ex. ha alla rader från 1 till n kan man använda kolon-notationen igen för att generera en indexvektor: `1:n` ger en radvektor med talen $1, \ldots, n$. Det går även att hoppa över (t.ex.) vart 3:e element med `1:3:n`. Ett större exempel:

```
>> A
A =
     1     4     7    10
     2     5     8    11
     3     6     9    12

>> A(2,3)
ans =
     8

>> A(2,:)
ans =
     2     5     8    11

>> A(:,3)
ans =
     7
     8
     9

>> A([1,2],[2,4])
ans =
     4    10
     5    11

>> A(:,2:2:4)
ans =
     4    10
     5    11
     6    12
```

Är v en $n \times 1$-matris (vektor i $\mathbb{R}^n$) så behövs bara ett index, och `v(i)` ger v_i. MEN, om A är en $n \times m$-matris så ger $A(i)$ det element som finns på plats i när man staplar alla kolonnerna i A ovanpå varandra, dvs. `A(p) = A(r,k)` om $p = (k-1)n + r$. Det är därför en bra vana att alltid indexera matriser med två index, för att undvika missförstånd.

För att ändra värdet på ett element i en matris kan man tilldela direkt till detta:

```
>> A(2,2) = -1
A =
     1     4     7    10
     2    -1     8    11
     3     6     9    12
```

Det fungerar även att ändra många värden samtidigt genom att först välja ut dem enligt ovan. Då måste det vi tilldelar ha rätt dimensioner:

```
>> A(1:2,3:4) = [0,0;0,0]
A =
     1     4     0     0
     2    -1     0     0
     3     6     9    12
```

Försöker vi tilldela ett värde till ett element som inte finns i matrisen A så utökar MATLAB helt enkelt A så att det går, och inför nollor där det behövs.

```
>> A(1,7) = 17
A =
     1     4     0     0     0     0    17
     2    -1     0     0     0     0     0
     3     6     9    12     0     0     0
```

(Detta är både användbart, och en källa till diverse fel.)

Speciella matriser

Det finns många funktioner för att skapa speciella matriser, t.ex. noll-matrisen som användes ovan. De mest använda är

`zeros(n,m)`	$n \times m$-matris med bara nollor
`ones(n,m)`	$n \times m$-matris med bara ettor
`eye(n,m)`	$n \times m$-matris med ettor på "diagonalen" och nollor annars
`rand(n,m)`	$n \times m$-matris med slumpmässiga värden mellan 0 och 1

Dessa kan alla också användas med bara ett index: `eye(n)` är samma sak som `eye(n,n)`. Fler funktioner för att skapa kända matriser hittar man genom hjälpsystemet i MATLAB, testa t.ex. `doc gallery`.

Andra användbara funktioner är

`tril(A)`	ger den undre triangulärmatrisen av A, dvs. sätter $A_{i,j} = 0$ om $i < j$
`triu(A)`	ger den övre triangulärmatrisen av A, dvs. sätter $A_{i,j} = 0$ om $i > j$
`diag(V)`	ger, om `V` är en rad- eller kolonnmatris, en matris med `V` på diagonalen och 0 i övrigt
`diag(A)`	ger, om `A` är en matris, en kolonnmatris med diagonalelementen i `A` som element
`flipud(A)`	ger en matris med raderna i omvänd ordning
`fliplr(A)`	ger en matris med kolonnerna i omvänd ordning

Uppgift 1. Skapa 6×8-matrisen

$$A = \begin{bmatrix} 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ -1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 7 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

genom att använda flera av de metoderna som beskrivits hittills. (Dvs. inte genom bara `A = [2, 4, 6, ...]`, det är både tråkigt och många siffror att skriva.) □

Ekvationssystem

För att lösa ett ekvationssystem $Ax = b$ har vi på föreläsningarna skrivit upp totalmatrisen $\left[\begin{array}{c|c} A & b \end{array}\right]$ och sedan transformerat den till radkanonisk form (reducerad trappstegsform) via radreducering. (Gausselimination.) De motsvarande två operationerna i MATLAB ges av `T = [A, b]` och `rref(T)`. Förkortningen `rref` står för *reduced row echelon form*.

I kursens övningsuppgifter behöver man vanligtvis inte göra radbyten när man radreducerar. MATLAB ser dock alltid till att göra radbyten så att pivotelementen blir maximala, för att mini-

mera numeriska fel. Eftersom den radkanoniska formen är entydig får vi likväl samma resultat som utan radbyte. (Tänk igenom varför!)

Exempel:

```
>> A = [1 2 3; 4 5 6; 7 8 0];
>> b = [3; 9; 6];
>> rref([A b])
ans =
     1     0     0     2
     0     1     0    -1
     0     0     1     1
```

Alltså är ekvationssystemet $Ax = b$ konsistent, och lösningen ges av $x = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$. Byter vi ut nollan i A mot värdet 9 får vi istället

```
>> A(3,3) = 9;
>> rref([A b])
ans =
     1     0    -1     0
     0     1     2     0
     0     0     0     1
```

dvs. $0 = 1$, och ekvationssystemet har ingen lösning. Det är dock möjligt att det finns en lösning för ett annat högerled, t.ex. har ju $Ax = 0$ alltid den triviala lösningen $x = 0$. Vi testar:

```
>> b = [6; 15; 24];
>> rref([A b])
ans =
     1     0    -1     0
     0     1     2     3
     0     0     0     0
```

Nu är istället $0 = 0$, dvs. det finns oändligt många lösningar som beror av en parameter.

Uppgift 2. Konstruera

$$A = \begin{bmatrix} 3 & 1 & 1 & 2 \\ 0 & 5 & 2 & 1 \\ 0 & 0 & 5 & 3 \\ 4 & 6 & 0 & 4 \end{bmatrix} \quad \text{och} \quad B = \begin{bmatrix} 3 & 7 & -4 & 1/2 \\ 0 & 5 & 2 & 1 \\ 0 & 0 & 5 & 3 \\ 4 & 6 & 0 & 4 \end{bmatrix}$$

samt

$$b = \begin{bmatrix} 6 \\ 9 \\ -2 \\ 20 \end{bmatrix} \quad \text{och} \quad c = \begin{bmatrix} 1/2 \\ 1 \\ 3 \\ 4 \end{bmatrix}.$$

Ekvationssystemen $Ax = b$ och $Bx = c$ har båda minst en lösning. Radreducera dem i tur och ordning via `rref`. Om det finns precis en lösning, plocka ut denna från resultatet och testa att det verkligen är en lösning. Om det finns oändligt många, använd resultatet från `rref` till att bestämma en av dem.

Tips: ett enkelt sätt att göra det senare är att byta sista raden i den radkanoniska formen till $\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}$ och sedan radreducera igen. Varför? □

Kommentarer

Vanligtvis använder man syntaxen `x = A\b` i MATLAB för att direkt bestämma lösningen till systemet $Ax = b$. Detta ger rätt svar så länge systemet har precis en (1) lösning. Om det inte finns en lösning får man en varning om att A är singulär (kapitel 2). Finns det många lösningar så kan olika saker hända. Ibland klarar MATLAB inte att lösa systemet, ibland får man en varning, och ibland får man en specifik lösning med eller utan varning. I det senare fallet är det en så kallad minsta-kvadrat-lösning, vilket vi skall prata mer om först i kapitel 6. Poängen är dock att det kan vara bra att köra `rref` om man är osäker.