TMA226 Matematisk fördjupning, Kf 2019
Tobias Gebäck
Matematiska vetenskaper, Chalmers & GU

TMA226 datorlaboration: introduktion

Syfte

Syftet med denna introduktion är att träna grundläggande kommandon i MATLAB, vilket kommer att underlätta arbetet med datorlaborationen. Dessa uppgifter behöver ej redovisas, men kontakta gärna lärare för att kontrollera att du gjort rätt. Se till att du förstår

1 Matriser och vektorer

Vi går här igenom några enkla sätt att skapa matriser och vektorer i MATLAB. Ett sätt att skapa en vektor är med kommandot `colon` (:). Du kan få hjälp om kommandot i MATLABgenom att skriva `doc colon` eller `help colon`. Testa båda!

Skapa nu tre 1×4-vektorer:

```
>> a=1:4; b=1:0.5:2.5; c=2*b;
```

Vad händer om du använder komma (,) istället för semikolon (;) mellan kommandona? Alltså, om du kör:

```
>> a=1:4, b=1:0.5:2.5, c=2*b
```

För att se hur du kan använda dina vektorer, kör följande kommandon och övertyga dig om att du förstår vad som händer för varje kommando. Vilka av dem är inte korrekta MATLAB-kommandon? Varför?

```
>> a * c
>> a .* c
>> a^2
>> a.^2
>> a'.^2
>> a' * c
>> a * c'
>> sum(a)
```

Notera att `.^` och `.*` är element-visa kommandon.

Vad är skillnaden mellan `length` och `size`? Prova

```
>> length(a), size(a)
```

För att skapa matriser, prova följande kommandon och undersök hur de fungerar:

```
>> [a c], [a,c]
>> [a; c]
>> diag(a), diag(a,1)
>> ones(3), ones(3,2)
```

```
>> zeros(3), zeros(3,2)
>> eye(3), eye(4,3)
>> A = [1 2 3; 4 5 6; 7 8 9]
>> B =[ diag(a) zeros(4,1); ones(1,5)]
```

För att komma åt enskilda rader eller kolonner i en matris kan du pröva

```
>> A(1,:), A(2,:), A(:,2), A(2:3,:)
>> B(end,:), B(:,end), B(:,1:3)
```

Några ytterligare användbara kommandon är:

```
>> C=repmat(a,3,1), D=repmat(a,3,2), E=repmat(a,1,2)
>> F=reshape(E,2,4), G=reshape(E,4,2)
>> sort(B), sortrows(B), sum(E)
>> sum(G), sum(G,1), sum(G,2)
```

Uppgift 1.
a) Skapa följande matris på en kommandorad:

$$A = \begin{bmatrix} 1 & 8 & 0 & 0 & 0 & 0 \\ -1 & 2 & 8 & 0 & 0 & 0 \\ 0 & -1 & 3 & 8 & 0 & 0 \\ 0 & 0 & -1 & 4 & 8 & 0 \\ 0 & 0 & 0 & -1 & 5 & 8 \\ 0 & 0 & 0 & 0 & -1 & 6 \end{bmatrix}$$

b) Ändra kommandot så att det givet en variabel `N` genererar en $N \times N$-matris med samma mönster.

Uppgift 2. Skapa matrisen A ovan med hjälp av en `for`-loop, dvs

```
for i=1:N
  A(i,i) = ...
  ...
end
```

2 Plotta data och funktioner

Vi repeterar här hur man plottar en funktion av en variabel, $y = f(x),\ x \in [a, b]$. För att göra det behöver vi först en indelning av intervallet $[a, b]$ i delintervall, eller punkter där funktionsvärdet skall plottas. Det kan göras antingen genom att välja en steglängd h och använda `colon`, dvs

```
>> x=a:h:b;
```

eller, genom att använda kommandot `linspace` och ange antalet punkter N:

```
>> x = linspace(a,b,N);
```

Om vi väljer $h = \frac{b-a}{N-1}$ kommer de två metoderna att ge samma vektor `x`.

En enkel funktion kan definieras med ett funktionshandtag (se `help function_handle`) som en så kallad "anonym funktion", t.ex.

```
>> f = @(x) exp(-8*(x-1).^2)
```

Notera att vi använd `.^` för att argumentet `x` skall kunna vara en vektor.

För att plotta funktionen på intervallet $[0, 2]$ kan vi nu skriva

```
>> x = linspace(0,2,100);
>> plot(x,f(x))
```

För att plotta mer data i samma figur och lägga till titlar kan man t.ex. skriva

```
>> hold on
>> A=[0.5 1 1.5]; B=[0.5 0 1];
>> plot(A,B,'ro')
>> title('Funktion med cirklar')
>> xlabel('x'), ylabel('y')
>> legend(func2str(f), 'cirklar')
```

Kommandot `plot` har många argument och möjligheter. Se `doc plot` för att se dem och andra relevanta kommandon.

Uppgift 3. Plotta funktionen $f(x) = \sin(3x) * \cos(x)$ på intervallet $[-2\pi, 2\pi]$, med en steglängd på $h = \pi/5$. Punkterna som plottas skall markeras med en röd +-symbol och förbindas med röda streckade linjer. Använd kommandot `axis` för att sätta axlarnas gränser till $[-2\pi, 2\pi]$ i x-led och $[-1, 1]$ i y-led. Figuren skall se ut så här: