Matlab instructie

Dr. ir. Gerrit C. Groenenboom
Subfaculteit Scheikunde
KUN

12 september 2002

1 plotten

Om een functie te plotten maak je eerst vectoren met de x en y waarden. Om b.v. een sinus te plotten tussen 0 en 2π maak je een vector met x-waarden:

```
x = ( 0 : 0.1 : 2*pi )'
```

de bijbehorende y-waarden kunnen met één statement uitgerekend worden:

```
y = sin(x)
```

het resultaat kun je plotten met

```
plot(x,y)
```

Als je meer functies in één figuur wilt plotten kan dat met, b.v.,

```
plot(x,[ sin(x) cos(x) ])
```

Je kunt ook functies toevoegen aan een plot door het `hold on` commando te geven:

```
clf      % clear figure
plot(x,y)
hold on
plot (x, x.^2)
```

Het lukt niet altijd om functies in één statement uit te rekenen. Stel je wilt de functie

$$f(\lambda) = ||\mathbf{a}_2 - \lambda\mathbf{a}_1|| \tag{1}$$

plotten als functie van λ. Je moet dan eerst een tabel met functiewaarden maken. Dat gaat het handigst op de volgende manier. Maak eerst een lijstje met λ waarden:

```
lambdas = [ -1 : 0.05 : 1 ]'
```

Het aantal λ waarden is nu:

```
n = length(lambdas)
```

Maak nu een lege tabel met functie waarden:

```
y = zeros(n,1)
```

Nu gebruiken we een zogenaamde “for” loop om de tabel te vullen:

```
for i = 1 : n
   lambda = lambdas(i);
   y(i) = norm(a2-lambda*a1);    % de ; voorkomt dat je alles over het scherm krijgt
end
```

en tenslotte maken we de plot

```
plot(lambdas,y)
```

Alles wordt nog overzichtelijker als we $f(\lambda)$ als MATLAB functie definieren. Dit lijkt veel op het maken van een script, behalve dat op de eerste regel komt te staan wat de invoer en uitvoer variabelen zijn. Dus, maak een filetje “flambda.m” met als eerste regel:

```
function [y] = flambda( lambdas, a1, a2)
```

We moeten behalve de lijst `lambdas` ook de vectoren `a1` en `a2` invoeren omdat in een MATLAB functie alle variabelen lokaal zijn. Vervolgens rekenen we de functie uit:

```
   function [y] = flambda( lambdas, a1, a2)
% geef eerst een korte uitleg van de functie, b.v.:
% y=flambda(lambdas, a1, a2)
% y=||a2 - lambda a1||

   y = zeros(size(lambdas));    % met dit trucje werkt het voor zowel rij- als kolom vectoren
   n = length(y);
   for i=1:n
      y(i)=norm(a2-lambdas(i)*a1);
   end
```

Na het saven van de file kunnen we de functie testen:

```
help flambda       % dit zou het commentaar op de eerste regels moeten laten zien
y = flambda(4, a1, a2)   % eerst met 1 lambda
y = flambda([4; 5; 6], a1, a2)   % nu met 3 lambdas
```

Nu kunnen we deze functie plotten op dezelfde manier als bij de sinus:

```
lambdas=[-3:0.1:3]'
plot(lambdas,flambda(lambdas,a1,a2))
```

Nu je de moeite gedaan hebt de functie te implementeren kun je ook eenvoudig het minimum vinden:

```
l0 = 0.3    % maak een schatting van het minimum door naar de plot te kijken
options = []    % later kun je hier extra opties invoeren.
l1 = fminsearch('flambda', l0, options, a1, a2)
```

De functie wordt dus geminimaliseerd voor het eerste argument, `a1` en `a2` worden vast gehouden.

Vraag: leg uit wat deze minimalisatie te maken heeft met projectie van vectoren.

2 Eigenwaarden en eigenvectoren

Het eigenwaardenprobleem

$$A\mathbf{u} = e\mathbf{u} \tag{2}$$

kan herschreven worden als

$$B(e)\mathbf{u} = 0 \tag{3}$$

met $B(e) = A - eI$, waarbij I een éénheidsmatrix is. Vgl. (3) heeft alleen niet-triviale oplossingen als $B(e)$ singulier is (determinant nul heeft). Neem

$$A = \begin{pmatrix} 0 & 2 & 4 \\ 2 & 2 & 3 \\ 4 & 3 & -1 \end{pmatrix}. \tag{4}$$

Maak een plot van $\det(B(e))$ voor $e \in [-5, 8]$ en schat de laagste eigenwaarde e_1 van de matrix A. Gebruik de routine `fzero` (die werkt bijna net als `fminsearch`) om een nauwkeurigere waarde voor e_1 te vinden.

Controleer je antwoord door de eigenwaarden van de matrix $\mathbf{A}$ uit te rekenen met behulp van de MATLAB functie `eig`.