Matlab instructie

Dr. ir. Gerrit C. Groenenboom
Subfaculteit Scheikunde
KUN

3 september 2003

Voorwoord

MATLAB is een zeer uitgebreid pakket. Er is daarom ook heel veel (online) informatie te vinden over MATLAB. In deze instructie wordt geen poging gedaan een overzicht te geven van de mogelijkheden. In plaats daarvan wordt aan de hand van voorbeelden de kortst mogelijke uitleg gegeven die nodig is om aan de computer opgaven chemische binding te kunnen beginnen. In deze instructie ga ik ervan uit dat je ieder voorbeeld gelijk uitvoert en dat je kijkt wat er gebeurt.

Inhoudsopgave

1 Matlab

Vanuit UNIX geef je het commando

```
matlab
```

en vanuit MS Windows vind je matlab in `G:\MatlabR13\MATLAB6.5`. Na enige tijd krijg je de matlab desktop te zien. Via de "Help" knop kun je alles over matlab te weten komen. Hier zullen we een aantal elementaire commando's bespreken die je in het `command window` achter de prompt (`>`) kunt intypen.

1.1 Invoeren van vectoren en matrices

Een rijvector met drie elementen wordt ingevoerd met, b.v.,

```
a = [ 7 1 -2]
```

Een matrix wordt per rij ingevoerd. matlab onderscheidt hoofd- en kleine letters en het is een goede gewoonte om voor matrices hoofdletters te gebruiken, dus, b.v.

```
A = [ 0.3 1e2 10; -0.1 3 4; 1 5 2]
```

Een kolom vector kun je ook opvatten als een rechthoekige matrix. De vector `b` had dus ook ingevoerd kunnen worden met:

```
b = [ 4 ; 5 ; -0.2]
```

Per definitie is de transpose $\mathbf{A}^T$ van een matrix $\mathbf{A}$ de matrix waarvan de elementen gegeven zijn door $(\mathbf{A}^T)_{i,j} = \mathbf{A}_{j,i}$. In matlab wordt de enkele quote gebruikt voor de transpose:

```
At = A'
```

De vector $\mathbf{b}$ had dus ook ingevoerd kunnen worden met

```
b = [ 4 5 -0.2 ]'
```

Let op: matlab kan ook met complexe getallen werken. De enkele quote geeft eigenlijk de complex geconjugeerde van de transpose van de matrix ("de hermitische toegevoegde"). Voor een matrix met alleen reële getallen maakt dit niets uit.

Het commando

```
a = a(:)
```

maakt van de rij vector `a` een kolom vector, zonder complex conjugeren. Als `a` al een kolom vector was geweest gebeurt er niets.

1.2 Rekenen

Je kunt 1/3 uitrekenen met

```
x = 1/3
```

Als je het resultaat geen naam geeft, b.v.,

```
1/3
```

komt het antwoord terecht in de variabele `ans`. Deze variabele wordt overschreven bij de volgende berekening waarvan het antwoord geen naam heeft gekregen. Je kunt het resultaat dus altijd nog redden door na het vorige commando in te typen:

```
x = ans
```

Een alternatief is om met pijltjes toetsen een vorig commando terug te halen en te editen. Een handigheidje: als je b.v. `A` intypt en daarna het pijltje omhoog krijg je het laatste commando dat met `A` begon terug.

Het is misschien opgevallen dat je de antwoorden maar in 4 cijfers nauwkeurig krijgt. Intern rekent MATLAB echter met zo'n 16 cijfers. Je krijgt deze cijfers te zien met:

```
format long
x
```

Als je meer over het commando `format` wilt weten—b.v. hoe je weer terug gaat naar de default instelling—typ je in:

```
help format
```

Van sommige commando's is de uitleg zo lang dat ie uit beeld scrollt. Dit kun je voorkomen met

```
more on
```

waarna je met `<return>` of `<spatie>` door de tekst gaat.

Bij goniometrische functies rekent MATLAB in radialen. Omdat de constante π is voorgedefinieerd kun je $\cos^2 30°$ uitrekenen met

```
cos(30*pi/180)^2
```

Via de help knop kun je lijsten met MATLAB functies vinden, om er een paar te noemen: `sqrt, exp, acos`.

1.3 Rekenen met matrices en vectoren

Een belangrijk voordeel van MATLAB is dat je heel eenvoudig met matrices en vectoren kunt werken. Een lineaire combinatie

$$\mathbf{c} = 2\mathbf{a} + 3\mathbf{b} \tag{1}$$

wordt berekend met

```
c = 2*a +  3*b
```

We hebben al gezien hoe je een matrix in kunt voeren door alle elementen te geven. De kolommen van een matrix zijn op te vatten als vectoren. Je kunt (kolom) vectoren die even lang zijn samenvoegen tot een matrix door ze naast elkaar te zetten

```
C = [ a b ]
```

Vraagje: wat gebeurt er in dit statement:

```
C2 = [ a' ; b' ]'
```

Nu we weten hoe we vectoren naast elkaar in een matrix kunnen zetten kunnen we gebruik maken van het matrix-vector product om een lineaire combinatie van vectoren te berekenen:

```
x = [2; 3]
c = C * x
```

Ter herinnering: een matrix-vector product

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{2}$$

is gedefinieerd door

$$y_i = \sum_j A_{i,j} x_j \tag{3}$$

Je kunt de elementen van een matrix er weer uit halen met

```
c11 = C(1,1)
c21 = C(2,1)
```

enz. Je kunt ook een hele kolom tegelijk uit een matrix halen:

```
a2 = C(:,1)
b2 = C(:,2)
```

Dit is eigenlijk een verkorte notatie van

```
b2 = C(1:end,2)
```

Omdat in dit geval de matrix drie rijen heeft is dit hetzelfde als

```
b2 = C(1:3,2)
```

waarbij `1:3` een verkorte notatie is van `[1 2 3]`. Je kan deze index-vector ook eerst in een variabele stoppen

```
ind = 1:3
b2  = C(ind,2)
```

Deze notatie is op een logische manier uit te breiden, probeer b.v.

```
D = C([1 3],:)
```

1.4 MATLAB scripts

Voordat we verder gaan is het handig om te weten hoe je je MATLAB commando's kan bewaren zodat je er later mee verder kan werken. Het idee is heel eenvoudig: zet de commando's in een file die de extensie `.m` heeft. *Tip: bij default laat* MS WINDOWS *geen file-extensies zien: pas dit aan door bij* `explorer-tools` *de* `hide file extensions` *weg te halen.* Maak eerst van je UNIX directory de current directory met behulp van het `cd` (change directory) commando

```
cd h:
```

Je kunt vanachter de MATLAB prompt overigens nog meer UNIX achtige commando geven:

```
pwd           (print working directory - laat zien waar je bent)
cd            (zonder argument is dit hetzelfde als pwd)
mkdir folder1 (maak nieuwe directory)
rmdir folder1 (gooi lege directory weg)
ls            (list files in current directory)
```

Je kunt nu op een aantal manieren een script-file met de naam `opg1.m` maken. Vanaf de prompt:

```
edit opg1
```

Links boven op je desk top zit hier ook een knop voor. Je kunt ook je favoriete editor of notepad gebruiken. Je kunt commentaar in een script zetten achter het procent teken, b.v.:

```
% opgave 1, maak een random 4x4 matrix
A = rand(4)
```

Als je je m-file ge-saved hebt kun je hem uitvoeren door de naam van het script (zonder extensie) in te typen:

```
opg1
```

Voor grote matrices wil je misschien niet iedere keer alle getallen over het scherm krijgen. In dat geval zet je een `;` achter een regel. Je kan zo ook meer commando's op één regel zetten:

```
A = rand(50); B=inv(A);
```

Een script dat niet in de current directory staat kun je beschikbaar maken met het `addpath` commando. Als je wilt weten dat MATLAB het juiste script uitvoert zijn de volgende commando's handig:

```
which opg1          (laat het hele pad zien)
more on
type opg1           (laat inhoud zien)
```

Als je met de hand commando's geeft en scripts gebruikt is het een goed idee om je scripts te testen door alle variabelen die je gemaakt hebt weg te gooien en dan het script opnieuw te draaien:

```
clear
clc            % dit maakt je scherm leeg
opg1
```

Je kunt zien welke variabelen je gemaakt hebt:

```
who            % geeft alleen de namen
whos           % laat ook nog zien wat de dimensies zijn
```

2 Lineaire vergelijking

Gegeven zijn de vier vectoren:

$$\mathbf{a}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \mathbf{a}_2 = \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix}, \mathbf{a}_3 = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} -1 \\ 4 \\ 5 \end{pmatrix}. \tag{4}$$

Gevraagd is: welke lineaire combinatie van de vectoren $\mathbf{a}_1, \mathbf{a}_2$ en $\mathbf{a}_3$ is gelijk aan de vector $\mathbf{b}$, oftewel, los op:

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + x_3\mathbf{a}_3 = \mathbf{b}. \tag{5}$$

Dit lineaire stelsel kan herschreven worden als

$$\mathbf{A} = [\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3] \tag{6}$$
$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{7}$$

In MATLAB kun je dit stelsel oplossen met

```
x = A\b
```

Als de inverse van $\mathbf{A}$ bestaat, d.w.z., $\mathbf{A}$ is een vierkante matrix waarvan de determinant ongelijk aan nul is [MATLAB commando: `det(A)`], kun je de oplossing ook schrijven als

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{8}$$

en in MATLAB wordt dit

```
x = inv(A) * b
```

In MATLAB is de methode met de backslash (\) het beste (iets nauwkeuriger en iets sneller). Let op: de forward slash (/) doet net iets anders:

```
C=A/B
```

is de "betere manier" om

```
C=A*inv(B)
```

uit te rekenen. In een enkel geval wil je helemaal geen lineair stelsel oplossen, maar, voor twee matrices `A` en `B` die dezelfde afmetingen hebben ieder element van `A` delen door een element van `B`:

$$D_{i,j} = \frac{A_{i,j}}{B_{i,j}} \tag{9}$$

Dit doe je in MATLAB met

```
D = A./B
```

Voor matrix vermenigvuldiging is de situatie analoog:

```
C = A * B
```

is een matrix vermenigvuldiging:

$$C_{i,j} = \sum_k A_{i,k} B_{k,j} \tag{10}$$

terwijl

```
D = A .* B
```

een zogenaamde puntsgewijze vermenigvuldiging is:

$$D_{i,j} = A_{i,j} B_{i,j} \tag{11}$$

De zogenaamde standaard basis voor $\mathbf{R}^3$ bestaat uit de drie vectoren $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{12}$$

Als je deze vectoren naast elkaar zet krijg je de eenheidsmatrix. In MATLAB heb je daarvoor het commando

```
E = eye(3)
```

Van het vak lineaire algebra weet je misschien nog de regel: “de kolommen van een matrix zijn de beelden van de basisvectoren”. Probeer te voorspellen wat hier uit komt:

```
e2 = E(:,2)
d  = A(:,2) - A*e2
```

Met de volgende commandos kun je een matrix of vector met allemaal nullen respectievelijk allemaal enen maken:

```
zeros(3)      % 3x3 matrix met nullen
ones(3)       % 3x3 matrix met enen
ones(3,1)     % kolom vector met enen
zeros(1,3)    % rijvector met nullen
```

Wat ook handig kan zijn:

```
D = diag([2 3 -1])      % diagonaal matrix met gegeven elementen
d = diag(D)             % stop diagonaal elementen weer in een vector
```

Bedenk een alternatief voor het commando `eye(17)`

2.1 Inproduct en norm

Het inproduct tussen twee vectoren $\mathbf{x}$ en $\mathbf{y}$ is gegeven door

$$\langle \mathbf{x} | \mathbf{y} \rangle = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \equiv x_1 y_1 + x_2 y_2 + x_3 y_3 = \sum_i x_i y_i \tag{13}$$

Door vectoren op te vatten als rechthoekige matrices kun je dit met een matrix vermenigvuldiging uitrekenen:

```
p = x'*y
```

Merk op dat het inproduct tussen complexe vectoren gedefinieerd is als

$$\sum_i x_i^* y_i \tag{14}$$

en dat is precies wat `x'*y` doet. De norm (ook wel lengte) van een vector is gedefinieerd door

$$||\mathbf{x}|| \equiv \sqrt{\langle \mathbf{x} | \mathbf{x} \rangle} \tag{15}$$

In MATLAB kun je de functie `norm` gebruiken:

```
norm(x)-sqrt(x'*x)
```

of je kunt zelf de som van de kwadraten uitrekenen met

```
sum(x.^2)
```

Let op:

```
length(x)
```

geeft het aantal elementen van een vector (3 in dit geval). De afmetingen van een matrix krijg je overigens met

```
size(A)
```

De hoek α tussen twee vectoren $\mathbf{x}$ en $\mathbf{y}$ is impliciet gedefinieerd door

$$\langle x|y\rangle = ||x||\ ||y|| \cos\alpha \tag{16}$$

Gebruik deze definitie om de hoek tussen de vectoren $\mathbf{a}_1$ en $\mathbf{a}_2$ uit vgl. (4) uit te rekenen (antwoord: 70.8934°). Als α gelijk is aan 90° staan de vectoren loodrecht op elkaar (het inproduct is dan nul).

We kunnen nu ook de projectie ($\mathbf{p}$) van de vector $\mathbf{a}_2$ op $\mathbf{a}_1$ uitrekenen. Per definitie is dit een vector die te schrijven is als $\mathbf{p} = \lambda\mathbf{a}_1$ zodanig dat de vector $\mathbf{d} = \mathbf{a}_2 - \mathbf{p}$ loodrecht staat op $\mathbf{a}_1$ (maak voor jezelf een schetsje maakt met al de genoemde vectoren). Leidt een formule af voor λ. Reken nu met MATLAB de vectoren $\mathbf{p}$ en $\mathbf{d}$ uit en controleer of de laatste inderdaad loodrecht staat op $\mathbf{a}_1$.

De formule voor $\mathbf{p}$ wordt iets eenvoudiger als je eerst de vector $\mathbf{a}_1$ normeert:

$$N = ||\mathbf{a}_1|| \tag{17}$$

$$\mathbf{q}_1 = \frac{1}{N}\mathbf{a}_1 \tag{18}$$

Vul nu in

$$\mathbf{a}_1 = N\mathbf{q}_1 \tag{19}$$

in de formule voor λ en laat zien dat N er uit valt.

Het vlak V opgespannen door de vector $\mathbf{a}_1$ en $\mathbf{a}_2$ is de verzameling van alle vectoren die kunnen worden geschreven als een lineaire combinatie van deze twee vectoren. Een orthonormale basis $\mathbf{Q} = [\mathbf{q}_1\mathbf{q}_2]$ voor het vlak V bestaat uit twee genormeerde vectoren in V die onderling loodrecht staan. Maak zo'n basis (hint: de vector $\mathbf{q}_1$ is een mooi begin, lees verder de definitie van projectie nog eens goed).

De vector $\mathbf{a}_2$ kan geschreven worden in de basis $\mathbf{Q}$

$$\mathbf{a}_2 = \mathbf{q}_1 c_1 + \mathbf{q}_2 c_2. \tag{20}$$

Bereken met MATLAB de expansiecoefficienten

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \tag{21}$$

Hoe kun je controleren of je antwoord goed is?

Met welk MATLAB statement kun je in één keer de expansiecoefficiënten van zowel $\mathbf{a}_1$ als $\mathbf{a}_2$ in de basis $\mathbf{Q}$ berekenen?

Laat zien dat de lengte van $\mathbf{a}_2$ gelijk is aan $\sqrt{c_1^2 + c_2^2}$ door vgl. 20 in te vullen in

$$||\mathbf{a}_2|| = \sqrt{\langle \mathbf{a}_2 | \mathbf{a}_2 \rangle} \tag{22}$$

Bereken de componenten b_1 en b_2 van de vector $\mathbf{b}$ (vlg. 4) in de basis $\mathbf{Q}$. Vergelijk $||\mathbf{b}||$ met $\sqrt{b_1^2 + b_2^2}$. Wat is hier aan de hand?

Bereken de projectie $\mathbf{f}$ van de vector $\mathbf{b}$ op het vlak V.

Wat is de afstand van $\mathbf{b}$ tot V?

Het zou nu duidelijk moeten zijn waarom het stelsel

$$[\mathbf{a}_1 \mathbf{a}_2]\mathbf{x} = \mathbf{b} \tag{23}$$

geen oplossing heeft. Echter, als je in MATLAB het commando

```
x = [ a1 a2 ] \ b
```

geeft, krijg je toch een antwoord. Om te begrijpen wat dit antwoord betekent vullen we $\mathbf{x}$ terug in:

```
z = [ a1 a2 ] * x
```

Vergelijk $\mathbf{z}$ met de eerder gevonden projectie $\mathbf{f}$. Omdat de vector $\mathbf{f}$ per definitie een lineaire combinatie is van de vectoren $\mathbf{a}_1$ en $\mathbf{a}_2$ moet het stelsel

$$[\mathbf{a}_1 \mathbf{a}_2]\mathbf{x} = \mathbf{f} \tag{24}$$

wel een oplossing hebben. De exacte oplossing $\mathbf{x}$ van deze vergelijking wordt de *kleinste kwadraten* oplossing van vgl. (23) genoemd.

De vector $\mathbf{g}$ en de matrix $\mathbf{D}$ zijn gegeven door

$$\mathbf{g} = 2\mathbf{a}_1 + 3\mathbf{a}_2 \tag{25}$$

$$\mathbf{D} = [\mathbf{a}_1 \mathbf{a}_2 \mathbf{g}]. \tag{26}$$

Probeer met MATLAB het lineaire stelsel

$$\mathbf{D}\mathbf{x} = \mathbf{b} \tag{27}$$

op te lossen. Wat is hier aan de hand? (Hint: bereken de determinant `det(D)`).

Beschouw het lineaire stelsel

$$\mathbf{Dz} = \mathbf{0} \equiv \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{28}$$

Eén (triviale) oplossing is onmiddelijk duidelijk: $\mathbf{z} = \mathbf{0}$. Als de inverse van $\mathbf{D}$ bestond kon je eenvoudig laten zien dat dit de enige oplossing was:

$$\mathbf{z} = \mathbf{D}^{-1}\mathbf{0} = \mathbf{0}. \tag{29}$$

We hebben—als het goed is—zojuist vastgesteld dat de inverse van $\mathbf{D}$ niet bestaat ($\mathbf{D}$ is *singulier*). Er zijn dus ook niet-triviale oplossingen en die gaan we nu proberen te vinden. Laat eerst zien dat als $\mathbf{z}$ een niet-triviale oplossing is, $\mathbf{z}' = \lambda\mathbf{z}$ ook een oplossing is (voor iedere scalar λ). Minstens één component van $\mathbf{z}$ moet ongelijk aan nul zijn. Laten we gokken dat dit de eerste component z_1 zodat we mogen kiezen $z_1 = 1$ (als we pech hebben moeten we maximaal drie keer opnieuw beginnen). Stel nu een vergelijking op voor z_2 en z_3 door $z_1 = 1$ in te vullen in vgl. (28). Vul hierbij niet de waarden van de elementen van $\mathbf{D}$ in, maar gebruik de notatie van vgl. (26).

Hoe kun je controleren of je het goede antwoord voor $\mathbf{z}$ hebt gevonden?

3 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

$$f(\lambda) = ||\mathbf{a}_2 - \lambda \mathbf{a}_1|| \tag{30}$$

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.

4 Eigenwaarden en eigenvectoren

Het eigenwaardenprobleem

$$A\mathbf{u} = e\mathbf{u} \tag{31}$$

kan herschreven worden als

$$B(e)\mathbf{u} = 0 \tag{32}$$

met $B(e) = A - eI$, waarbij I een éénheidsmatrix is. Vgl. (32) heeft alleen niet-triviale oplossingen als $B(e)$ singulier is (determinant nul heeft). Neem

$$A = \begin{pmatrix} 0 & 2 & 4 \\ 2 & 2 & 3 \\ 4 & 3 & -1 \end{pmatrix}. \tag{33}$$

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`.