Wetenschappelijk Rekenen

Examen - Bacheloropleiding informatica

Oefeningen – 3 september 2014

1. Beschouw de matrix

$$A = \begin{bmatrix} -8 & -1 & -6 \\ -3 & -5 & -7 \\ -4 & -9 & -2 \end{bmatrix}$$

Deze matrix heeft -15 als dominante eigenwaarde. We proberen deze eigenwaarde te berekenen met een implementatie van machtiteratie in het bestand *vraag1_machtiteratie.m* op Indianio.

(a) Wat loopt er precies mis bij het uitvoeren van

```
dominante_eigenwaarde = vraag1_machtiteratie(A, [1;2;3]);
```

De startvector $[1\ 2\ 3]^T$ is een goedgekozen vector en hoeft niet te worden gewijzigd. Pas het bestand *vraag1_machtiteratie.m* aan zodat dit probleem wordt vermeden. Let op dat je geen nieuwe fouten introduceert voor matrices verschillend van A.

Bewaar je aangepaste bestand onder de naam *vraag1_machtiteratie_oplossing.m*.

(b) Bereken een opzettelijke slechte keuze voor de startvector waardoor *vraag1_machtiteratie.m* de niet-dominante eigenwaarde 4.89... zal teruggeven. Verklaar kort hoe je aan deze keuze komt en waarom deze keuze zal werken. Schrijf hiervoor een script in Matlab.

(c) Zijn de eigenwaarden van A gevoelig voor veranderingen in de elementen van A ? Verklaar kort waarom.

Oplossing:

(a) De variabele `x` zal na verloop van tijd een eigenvector bevatten uit de eigenruimte van de eigenwaarde -15:

- iteratie 996: `x = [1 1 1]'`
- iteratie 997: `x = [-1 -1 -1]'`
- iteratie 998: `x = [1 1 1]'`
- iteratie 999: `x = [-1 -1 -1]'`

De vector `x` zal niet convergeren maar alterneren tussen twee eigenvectoren horende bij -15 die op een factor -1 van elkaar verschillen. Dit is het gevolg van een negatieve dominante eigenwaarde.

- Bij `x = [1 1 1]'` is `A*x` gelijk aan `[-15 -15 -15]`. We normaliseren door te delen door 15, dit is de waarde van `norm(x,Inf)`. We bekomen `[-1 -1 -1]'`.
- Bij `x = [-1 -1 -1]'` is `A*x` gelijk aan `[15 15 15]`. We normaliseren door te delen door 15, dit is de waarde van `norm(x,Inf)`. We bekomen `[1 1 1]'`.

De stopvoorwaarde `norm(x0 - x,Inf) > TOLERANCE` faalt in het geval van negatieve dominante eigenwaarden omwille van de tekenverandering na elke iteratiestap. De berekening `norm(x0 - x,Inf)` zal convergeren naar de waarde 2 en zal nooit kleiner worden dan de tolerantie.

Er zijn verschillende manieren om dit probleem op te lossen. Men kan bijvoorbeeld een nieuwe stopvoorwaarde implementeren die kan omgaan met deze tekenwissels. In deze oplossing zullen

we de stopwaarde behouden en de normering aanpassen zodat de componenten van de vector `x` hun teken behouden bij convergentie naar de eigenruimte.

In het geval waar `x = [1 1 1]'` en `A*x` gelijk aan `[-15 -15 -15]`, delen we beter door -15 in plaats van 15. Hierdoor bekomen we dan terug `[1 1 1]'`. In een normeringsstap zullen we, net zoals bij de berekening van `norm(x,Inf)`, de component met grootste magnitude gebruiken. Deze keer zullen we het teken van het component ook gebruiken in de normalisatiestap. Dit zorgt ervoor dat bij convergentie naar de eigenruimte het teken van de vector `x` niet langer meer zal wisselen na elke stap. Hierdoor leggen we het teken van de componenten van `x` vast.

In *vraag1_machtiteratie.m* vervangen we de normalisatiestap

```
x = x / norm(x,Inf);
```

door

```
[~,index] = max(abs(x));
x = x / x(index);
```

De volledige oplossing kan je vinden in het bestand *vraag1_machtiteratie_opl.m*

(b) De beste keuze voor een startvector is een eigenvector van de eigenwaarde $4.89\ldots$. Omdat de startvector een eigenvector is zal de stopvoorwaarde meteen voldaan zijn en het programma zal onterecht besluiten dat de dominante eigenwaarde gelijk is aan $4.89\ldots$.

We bereken de startvector met de volgende code:

```
>> [X,D] = eig(A)
X =

   0.577350269189626   0.813052529585141  -0.341648008794110
   0.577350269189626  -0.471404520791032  -0.471404520791032
   0.577350269189626  -0.341648008794110   0.813052529585142
D =

 -15.000000000000004                   0                   0
                   0  -4.898979485566359                   0
                   0                   0   4.898979485566358

>> x0 = X(:,3);     #eigenvector van 4.89..
>> vraag1_machtiteratie(A, x0)

ans =
   4.898979485566360
```

In dit geval zal *vraag1_machtiteratie.m* niet de kans krijgen om te convergeren naar de dominante eigenwaarde omwille van de stopvoorwaarde. Indien we de iteraties verder laten doorlopen, dan zal er toch uiteindelijk convergentie zijn naar de dominante eigenwaarde omwille van afrondingsfouten die een component doen onstaan in de eigenruimte van de dominante eigenwaarde.

(c) We berekenen het conditiegetal van de matrix van eigenvectoren van A om de gevoeligheid na te gaan.

```
>> [X,D] = eig(A);
>> cond(X)

ans =
   1.414213562373095
```

Het conditiegetal ligt vrij dicht bij perfecte conditionering (waarde 1), we besluiten hieruit dat de eigenwaarden van A niet gevoelig zijn voor veranderingen in de elementen van A.

2. Pas Newton-Raphson toe om waarden x te vinden waarvoor geldt dat $f(x) = x^2$, waarbij f een $\mathbb{R} \to \mathbb{R}$ functie is. Schrijf hiervoor de volgende MATLAB functie:

   ```
   x = vraag2(f, df, x0)
   ```

   Hierbij is `f` de functie, `df` de afgeleide van de functie en `x0` de startwaarde voor het Newton-Raphson algoritme. Het bestand *vraag2_tests.m* op Indianio bevat enkele tests voor `vraag2` met de verwachte uitkomst.

   Oplossing: We zoeken waarden voor x waarvoor $f(x) = x^2$ of waarvoor $f(x) - x^2 = 0$. We zoeken nulpunten van de functie $g(x) = f(x) - x^2$. Hierop passen we Newton-Raphson toe:

   $$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)} = x_n - \frac{f(x_n) - x_n^2}{f'(x_n) - 2\,x_n}$$

   We implementeren dit iteratief schema in *vraag2.m*:

   ```
   function [x] = vraag2(f, df, x)
   TOL = 1e-14;
   xp = Inf;
   while (abs(xp - x) >= TOL)
       xp = x;
       x = x - (f(x) - x^2) / (df(x) - 2*x);
   end
   end
   ```

   ```
   >> vraag2_tests
   test1: uitkomst = 0.82413231230252, verwacht = 0.824132...
   test2: uitkomst = 4.8135691520462, verwacht = 4.813569...
   test3: uitkomst = 0, verwacht = 0
   test4: uitkomst = 0.83360619440668, verwacht = 0.833606...
   ```

3. Bepaal met Maple de 5-punts Lobatto-kwadratuurformule Q die de integraal

   $$\int_{-1}^{1} f(x)\,\mathrm{d}x$$

   benadert.

   Bepaal ook de constanten p en α in de uitdrukking

   $$\int_{-1}^{1} f(x)\,\mathrm{d}x - Q(f) = \alpha f^{(p)}(\xi)$$

   met $\xi \in [-1, 1]$. Wat is de graad van Q ?

   Oplossing: Bij Lobatto kwadratuur worden twee knopen gelijkgesteld aan de grenzen -1 en 1. De resterende 3 knopen en de 5 gewichten zijn vrij. We maximaliseren de graad door 8 voorwaarden op te leggen.

   De volgende Maple-instructies klaren de klus (zie ook *vraag3.mw*):

```
x[1] := -1; x[5] := 1
Q := f -> w[1]*f(x[1])+w[2]*f(x[2])+w[3]*f(x[3])
         +w[4]*f(x[4])+w[5]*f(x[5])

seq(sum(w[i]*x[i]^k, i = 1 .. 5) = int(x^k, x = -1 .. 1), k = 0 .. 7):
solve({%}):
allvalues(%[1]):
assign(%[1]):
Q(f);
```

$$Q(f) = \frac{1}{10} f(-1) + \frac{49}{90} f\left(\frac{-1}{7}\sqrt{21}\right) + \frac{32}{45} f(0) + \frac{49}{90} f\left(\frac{1}{7}\sqrt{21}\right) + \frac{1}{10} f(1)$$

Maple levert meerdere oplossingen voor het stelsel. Alle oplossingen leiden echter naar dezelfde formule. We bepalen p en α:

```
testDegree := k -> sum(w[i]*x[i]^k, i = 1 .. 5) = int(x^k, x = -1 .. 1);
[testDegree(7), testDegree(8)];
```

$\left[0 = 0,\ \ \frac{58}{245} = \frac{2}{9}\right]$

```
p := 8;
solve({2/9 - 58/245 = alpha*factorial(p)});
```

$\left\{\alpha = -\frac{1}{278300}\right\}$

$$\int_{-1}^{1} f(x)\mathrm{d}x - Q(f) = -\frac{1}{278300} f^{(8)}(\xi)$$

De graad van Q is 7.

4. In de oefeningenles hebben we Adams-formules berekend zoals deze 4-staps formule:

$$y_{n+1} - y_n = \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

Er bestaat ook een formule van de vorm

$$y_{n+1} - y_{n-1} = h\ (A\, f_n + B\, f_{n-1} + C\, f_{n-2} + D\, f_{n-3})$$

Vul de coëfficienten A, B, C en D in met Maple zodanig dat de orde van deze formule maximaal is.

Oplossing: De 4-staps Adams-Bashforth methodes kan je bekomen door het interpoleren van $y' = f$ door 4 voorgaande punten en het integreren van de resulterende interpolatieveelterm. We hebben

$$y_{k+1} = y_k + \int_{t_k}^{t_{k+1}} f(t, y(t))dt \approx y_k + \int_{t_k}^{t_{k+1}} p(t)dt$$

waarbij $p(t)$ de interpolatieveelterm is door de 4 punten

$$(t_i, f_i), (t_{i-1}, f_{i-1}), (t_{i-2}, f_{i-2}), (t_{i-3}, f_{i-3})$$

met f_i een benadering van $f(t_i, y(t_i))$.

We kunnen nu dezelfde techniek toepassen voor de nieuwe formule:

$$y_{k+1} = y_{k-1} + \int_{t_{k-1}}^{t_{k+1}} f(t, y(t))dt \approx y_k + \int_{t_{k-1}}^{t_{k+1}} p(t)dt$$

Voor het berekenen van de coëfficienten kunnen we code uit *AdamsBashforth.mw* (Les 10) hergebruiken. We hoeven enkel de integratiegrenzen aan te passen bij het integreren van de interpolatieveelterm.

We starten met de code voor het opstellen van een 4-staps Adams-Bashforth formule:

```
with(CurveFitting):
p := PolynomialInterpolation([t[k]-3*h, t[k]-2*h, t[k]-h, t[k]], [f[k-3], f[k-2], f[k-1],
f[k]], tt);
i := int(p, tt = t[k]-h ..  t[k]+h);
simplify(i);
```

$$\frac{1}{3}\, h\, (4f_{k-2} - 5f_{k-1} - f_{k-3} + 8f_k)$$

We hebben hier `t[k]..t[k]+h` vervangen door `t[k]-h..t[k]+h`.

Dit geeft ons de methode

$$y_{n+1} - y_{n-1} = \frac{h}{3}\, (8f_n - 5f_{n-1} + 4f_{n-2} - f_{n-3})$$

Besluit: $A = \frac{8}{3}$, $B = -\frac{5}{3}$, $C = \frac{4}{3}$ en $D = -\frac{1}{3}$.