Wetenschappelijk Rekenen

Examen - Bacheloropleiding informatica

Oefeningen – 22 augustus 2013

1. Hoe zou je de vector x in de uitdrukking

$$Q\,x = A^{-n}\,y$$

op een computationeel slimme manier berekenen ?

Hierbij is gegeven:

- een vector y van lengte m.
- een strikt positief geheel getal n.
- een orthogonale $m \times m$ matrix Q.
- de $m \times m$ onderdriehoeksmatrix L_A, de $m \times m$ bovendriehoeksmatrix U_A en de $m \times m$ permutatiematrix P_A waarbij $L_A\,U_A = P_A\,A$.

Oplossing: Laten we uitdrukking $Q\,x = A^{-n}\,y$ herschrijven. Noem $\hat{x} = Q\,x$, nu zoeken we $\hat{x}$ in de uitdrukking $A^n\,\hat{x} = A(A(A(\ldots A\,\hat{x})\ldots)) = y$. Dit komt neer op het n maal oplossen van het stelsel $A\,\hat{x}^{(i+1)} = \hat{x}^{(i)}$ met $\hat{x}^{(0)} = y$. De matrix A kennen we niet rechtstreeks maar we kennen wel de LU-decompositie van A met (expliciete) pivotering: $L_A\,U_A = P_A\,A$. Dit is de decompositie die we bekomen indien we de `lu` functie van MATLAB oproepen met drie uitvoervariabelen: "`[LA, UA, PA] = lu(A);`". Zie ook *GaussElim.m* op Minerva. Om op een computationeel slimme manier één maal het stelsel $A\,\hat{x}^{(i+1)} = \hat{x}^{(i)}$ op te lossen zullen we eerst het stelsel $L_A\,z = P_A\,\hat{x}^{(i)}$ oplossen met voorwaartse substitutie en vervolgens zullen we het stelsel $U_A\,\hat{x}^{(i+1)} = z$ oplossen met achterwaartse substitutie. Tenslotte bekomen we x door $x = Q^T\hat{x}$ te berekenen. Indien we dit zouden implementeren in MATLAB bekomen we de volgende oplossing:

```
function x = Vraag1(n , Q , LA , UA , PA , y)
    x = y;
    for i = 1:n
       x = UA\(LA\(PA*x));
    end
    x = Q'*x;
end
```

2. Het bestand *zoekVastPunt.m* bevat de implementatie van de functie F en zijn jacobiaan J_F, gegeven door

$$F : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 3x(1-x) - \frac{2xy}{1+x} \\ \frac{7xy}{2(1+x)} \end{pmatrix}, \quad J_F : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 3(1-2x) - \frac{2y}{(1+x)^2} & -\frac{2x}{1+x} \\ \frac{7y}{2(1+x)^2} & \frac{7x}{2(1+x)} \end{pmatrix}$$

We zijn op zoek naar de drie vaste punten van F, dit zijn de punten $\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$ waarvoor $\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = F\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$.

(a) Bepaal de drie vaste punten van F in exacte arithmetiek.

(b) Pas de methode van Newton toe om de vaste punten van F te bepalen door deze toe te passen op de uitdrukking $F(X) = X$. Je kan hierbij gebruik maken van de code uit *zoekVastPunt.m* maar let op welke aanpassingen er nodig zullen zijn bij de toepassing van de methode van Newton. Merk op dat we voor dit probleem niet vast-punt-iteratie kunnen toepassen omdat niet elk vast punt de convergentie-eigenschap bezit (zie dia 39, Hoofdstuk 5). Met de startwaarden $\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, $\begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$ en $\begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$ kan men de drie vaste punten van F vinden door de methode van Newton toe te passen.

(c) Eenmaal een vast punt gevonden met de methode van Newton, ga na of het gevonden vast punt al dan niet de convergentie-eigenschap bezit beschreven in dia 39. Laat je programma antwoorden met een 'convergent' of 'niet convergent'.

Oplossing:

(a) Uit $x = 0$ volgt dat $y = 0$. Dus $(0, 0)^T$ is een vast punt. Voor $x \neq 0$ en $y = 0$ volgt dat $x = 3\,x\,(1 - x)$, $1 = 3\,(1 - x)$ en dus $x = \frac{2}{3}$. Een tweede vaste punt is $(\frac{2}{3}, 0)^T$. Voor $x \neq 0$ en $y \neq 0$ kunnen we uit $\frac{7xy}{2(1+x)} = y$ afleiden dat $x = \frac{2}{5}$. Dan is y gelijk aan $\frac{14}{25}$. Het derde vaste punt is $(\frac{2}{5}, \frac{14}{25})^T$. We kunnen dit ook berekenen met Maple:

```
solve({x = 3*x*(1-x)-2*x*y/(1+x), y = 7*x*y/(2*(1+x))});
```

$$\{x = 0, y = 0\}\,,\ \left\{x = \frac{2}{3}, y = 0\right\},\ \left\{x = \frac{2}{5}, y = \frac{14}{25}\right\}$$

(b) We vormen de vergelijking $F(X) = X$ om naar een nulpuntprobleem: $F(X) - X = 0$. De jacobiaan voor deze vergelijking wordt dan $J_F(X) - I$, de jacobiaan die we zullen gebruiken in de Newton iteraties. I is de 2×2 eenheidsmatrix.

(c) Uit dia 39 (H5) leren we dat een vast punt de convergentie-eigenschap bezit indien $\rho(J_F(X)) < 1$. We moeten dus nagaan of de absolute waarden van de eigenwaarden van de jacobiaan in het vast punt X strikt kleiner zijn dan 1. Een manier om dit na te gaan is via de MATLAB commando: `abs(eig(F_Jac(X))) < 1`.

We bekomen de volgende oplossing in MATLAB:

```
function X = zoekVastPunt(startX , tolerantie)
X = startX;
r = F(X) - X;
while (norm(r) > tolerantie)
    delta = (F_Jac(X) - eye(2)) \ -r;
    X = X + delta;
    r = F(X) - X;
end
```

```
%opmerking: er zijn verschillende aanvaardbare implementaties.

if (abs(eig(F_Jac(X))) < 1 );
    fprintf('convergent \n');
else
    fprintf('niet convergent \n');
end
```

We passen de oplossing toe op de drie gegeven startwaarden:

```
>> zoekVastPunt([0.5;0.5] , 10^-12)
convergent

ans =

   0.400000000000000
   0.560000000000000

>> zoekVastPunt([0.5;0] , 10^-12)
niet convergent

ans =

   0.666666666666667
   0

>> zoekVastPunt([0;0.5] , 10^-12)
niet convergent

ans =

     0
     0
```

3. (a) Het bestand *methodeMeerstapsMethode.m* bevat de implementatie van de volgende meerstapsmethode voor het oplossen van differentiaalvergelijkingen met een vaste stap h:

$$y_{k+1} = y_k + \frac{h}{24}\left(9\,f(t_{k+1}, y_{k+1}) + 19\,f(t_k, y_k) - 5\,f(t_{k-1}, y_{k-1}) + f(t_{k-2}, y_{k-2})\right)$$

Aangezien deze methode niet zelfstartend is doen we, als deel van de initialisatie, beroep op de trapeziumregel tot er voldoende y-waarden beschikbaar zijn. In het bestand *voorbeeld_MeerstapsMethode.m* wordt deze methode toegepast op het stelsel beschreven in PC-les 10 (hoofdstuk 9), vraag 1. Pas het bestand *methodeMeerstapsMethode.m* aan zodanig dat we de volgende meerstapsmethode toepassen:

$$y_{k+1} = y_k + \frac{h}{24}\left(55\,f(t_k, y_k) - 59\,f(t_{k-1}, y_{k-1}) + 37\,f(t_{k-2}, y_{k-2}) - 9\,f(t_{k-3}, y_{k-3})\right) \tag{1}$$

(b) Verifieer dat de orde van de meerstapsmethode uit formule (1) gelijk is aan 4.

Oplossing:

(a) Merk op dat de nieuwe methode een vier-staps methode is in plaats van een drie-staps methode. Voor dat men de nieuwe meerstapsmethode kan toepassen moeten er dus 4 stappen beschikbaar zijn. We passen hiervoor lijn 14 aan in het bestand *methodeMeerstapsMethode.m*. We vervangen

```
while (x0 < xend-h/2) && (initSteps < 3)
```

door

```
while (x0 < xend-h/2) && (initSteps < 4)
```

Merk ook op dat de nieuwe methode een expliciete methode is in plaats van een impliciete methode. Dit is vergelijkbaar met het verschil tussen de voorwaartse Euler en achterwaartse Euler. We vervangen lijn 39 tot en met lijn 51 door:

```
function y_next = meerstapsregel(g,x,y,jac,h)
y_next= y(:,end) +(h/24)*(55*g(x(end),y(:,end)) ...
   - 59 * g(x(end-1),y(:,end-1)) + 37 * g(x(end-2),y(:,end-2)) ...
     - 9 * g(x(end-3),y(:,end-3)));
end
```

Het aangepaste bestand is beschikbaar als *methodeMeerstapsMethode_oplossing.m* samen met *voorbeeld_MeerstapsMethode_oplossing.m*.

(b) We bepalen de orde met Maple als volgt:

```
verschil := y(t[k]+h) - y(t[k]) -(1/24)*h*( 55*(D(y))(t[k]) -59*(D(y))(t[k]-h)
+37*(D(y))(t[k]-2*h) -9*(D(y))(t[k]-3*h)):
series(verschil, h);
```

$$\frac{251}{720} D^{(5)}(y(t_k))h^5 + O(h^6)$$

Uit h^5 lezen we af dat de orde van deze methode vier is.

4. Het bestand *trein.mat* op Indianio bevat een geluidsfragment van een treinfluit dat exact 1.5 seconden duurt. We laden het fragment in MATLAB als volgt:

```
>> load('trein.mat'); %zie 'pwd' en 'ls' indien 'no such file or directory'
>> plot(trein); %visualiseer geluidsfragment
```

Bepaal de dominante geluidsfrequentie in Hz. [1]

Oplossing: In MATLAB voeren we uit:

```
>> Y = fft(trein);
>> [maximum , index] = max(abs( Y(1:length(Y)/2) ));
```

De variabele `index` bevat nu de index van de dominante frequentie, maar aangezien de 'DC' component nog aanwezig is in `Y` op positie 1 moeten we dit nog corrigeren:

[1] Hertz: De eenheid van Frequentie in cycli per seconde.

```
>> index = index - 1; %corrigeer voor aanwezigheid 'DC'
```

De waarde in `index` komt nu overeen met de frequentie (in Hz) indien het fragment 1 seconde zou duren, we moeten dit nog corrigeren:

```
>> dominante_freq = index / 1.5;

dominante_freq =

     928.67
```