Wetenschappelijk Rekenen

Examen - Derde bachelor informatica

Oefeningen – 31 mei 2011

1. De volgende functies zijn gegeven:

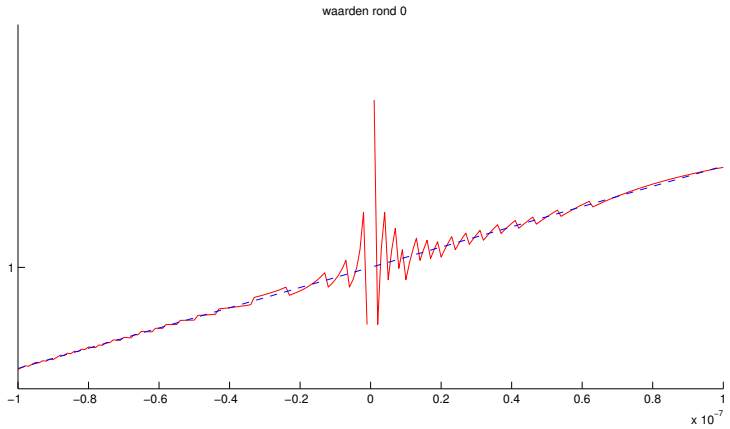
$$f(x) = \frac{e^x - 1}{x} \qquad\qquad g(y) = \frac{y - 1}{\log_e y}$$

Merk op dat de functie $g(y)$ dezelfde functie is als $f(x)$ waarbij we $y = e^x$ nemen. Merk op dat $f(x)$ onbepaald is voor $x = 0$ en dat de functie $g(y)$ onbepaald is voor $y = 1$. Wanneer we inzoomen op de eerste functie rond het punt $x = 0$ krijgen we veel ruis te zien. Zoomen we echter in op de tweede functie rond het punt $y = e^x = 1$, dan zien we veel minder ruis.

De code die we gebruiken om de functies nader te bekijken is

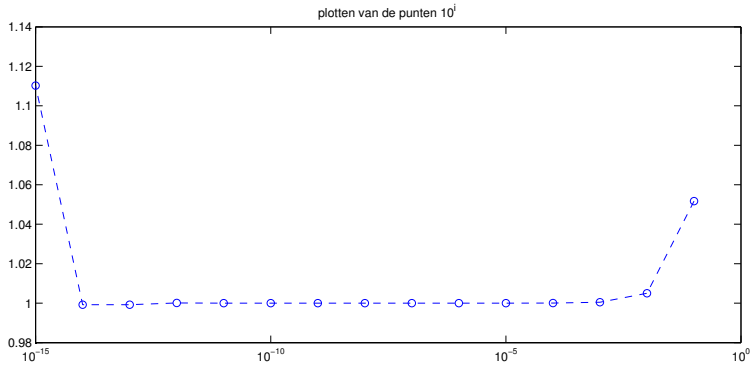
```
x=10.^(-7)*[-1:0.01:1];
dx = (exp(x)-1)./x;

y = exp(x);
dy = (y-1)./log(y);

figure;
hold on;
plot(x,dx,'-r');
plot(x,dy,'--b');
hold off;
```

De grafiek die je zodoende bekomt is

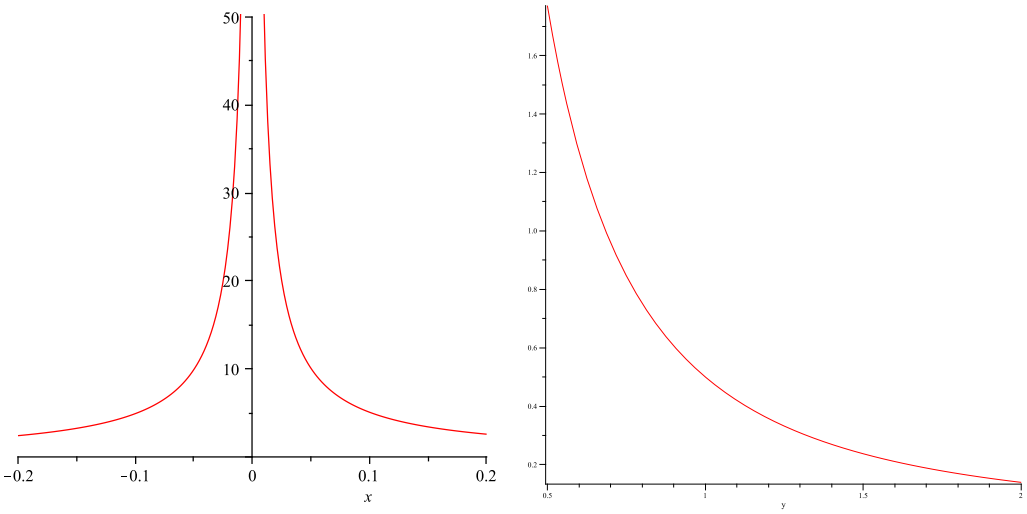


waarbij $f(x)$ met een volle lijn wordt geplot, de waardes $(x, g(y))$ met een gestreepte lijn. Verklaar waarom we bij de functie $f(x)$ zoveel ruis zien.

Oplossing: Plotten we de functiewaardes $f(x)$, dan zien we problemen onstaan naarmate we dichter komen bij 0. We krijgen de grafiek:



Het probleem dat we zien, komt doordat $\lim_{x\to 0}\left(cond(f(x)) \approx \left[\frac{f'(x)}{xf(x)}\right]\right) = \infty$, terwijl $\lim_{y\to 1}\left(cond(g(y)) \approx \left[\frac{g'(y)}{yg(y)}\right]\right) = \frac{1}{2}$. Zie hieronder links een plot van het conditiegetal van $f(x)$ nabij 0, rechts een plot van het conditiegetal van $g(x)$ nabij $e^0 = 1$.



De ruis die we te zien krijgen bij het plotten van de functie $f(x)$ komt omdat de functie $f(x)$ slecht geconditioneerd is voor waardes in de buurt van $x = 0$.

2. In de oefeningen hebben we de bestanden `Voorbeeld1.m` en `methodeEuler.m`, waarbij we de Euler-methode gebruiken om een stelsel van ODEs op te lossen, en de bestanden `Voorbeeld2.m` en `methodeAchterwaartseEuler.m`, waarbij we de Achterwaartse-Euler-methode gebruik, grondig bekeken.

   Maak een nieuw bestand `voorbeeld_vraag2.m` en `methodeTheta.m` aan (uiteraard gebaseerd op de hiervoor vermeldde bestanden!) waarin je ditmaal de algemene theta methode implementeert voor $\theta = \frac{2}{3}$. Deze methode is gedefinieerd als

$$y_{n+1} - y_n \approx h\left[\theta f(t_{n+1}, y_{n+1}) + (1-\theta)f(t_n, y_n)\right] \qquad \theta \in [0,1]$$

Denk goed na over wat je zoal moet aanpassen. Werkt deze methode beter of slechter voor het in `Voorbeeld1.m` en `Voorbeeld2.m` beschreven probleem? Hoe kan je dit zien? Kan je theoretisch verklaren waarom de ene methode beter is dan de andere?

Oplossing: Merk allereerst op dat dit een mogelijk impliciete methode is. Daarom is het dus verstandig om te starten vanaf de code van de achterwaartse Euler, aangezien dit ook een impliciete methode is. Vervolgens volstaat het om – naast het veranderen van een paar functienamen – twee lijntjes code aan te passen in het algoritme. In plaats van

```
f=-(y-y0-h*feval(g,x1,y));
```

schrijven we nu

```
f=-(y-y0-h*(theta * feval(g,x1,y) + (1-theta) * feval(g,x0,y0)));
```

Ook de Jacobiaan verandert lichtelijk. In plaats van

```
A=ident-h*feval(jac,x1,y);
```

schrijven we nu

```
A=ident-theta*h*feval(jac,x1,y);
```

Afhankelijk van de waarde voor θ krijgen we verschillende methodes. Kiezen we $\theta = 0$, dan bekomen we de Euler-methode. Kiezen we $\theta = 1$, dan bekomen we de achterwaartse Euler-methode. De trapeziumregel wordt bekomen als we $\theta = 1/2$ nemen. Voor $\theta = 2/3$ vinden we dat dit nog steeds eerste-orde nauwkeurig is, net zoals de methode van Euler en de achterwaartse Euler en in tegenstelling tot $\theta = 1/2$ die tweede-orde nauwkeurig is, maar dan met een fout die theoretisch drie keer kleiner is dan de fout bekomen met de (achterwaartse-)Euler-methode.

Inderdaad, wanneer we een reeksontwikkeling bekijken om de fout na te gaan, dan vinden we dat

$$\begin{aligned}
&y(x_{n+1}) - y(x_n) - \theta h y'(x_{n+1}) - (1-\theta) h y'(x_n) \\
&= h y'(x_n) + \frac{h^2 y''(x_n)}{2} + \ldots \\
&\quad - \theta h (y'(x_n) + h y''(x_n) + \ldots) \\
&\quad - (1-\theta) h y'(x_n) \\
&= h^2 (\frac{1}{2} - \theta) y''(x_n) + \mathcal{O}(h^3)
\end{aligned}$$

Voor $\theta = \frac{2}{3}$ vinden we dan $\frac{1}{2} - \frac{2}{3} = \frac{3-4}{6} = -\frac{1}{6}$ en voor $\theta = 1$ vinden we $\frac{1}{2} - 1 = -\frac{1}{2}$, bijgevolg is de leidende term van de fout voor $\theta = \frac{2}{3}$ drie keer kleiner dan bij de 2 Euler-regels.

De volledige code in `voorbeeld_vraag2.m` is

```
function voorbeeld_vraag2(h);

% toepassing van de Achterwaartse Euler-methode in [0,1] op het probleem:
%
%     y(1)' = y(2)
%     y(2)' = y(2)*(y(2)-1)/y(1)

xspan=[0,1]; % het integratie-interval
y0=oplossing23(xspan(1)); % beginwaarde
```

```
[x,y]=methodeTheta(@probleem23,@J23,h,xspan,y0);   %   ! andere functie

figure;
plot(x,y(1,:),x,y(2,:)); % een plot van de numerieke oplossing
title(['Numerieke oplossing']);
xlabel('x');
ylabel('y');

% Vermits de analytische oplossing gekend is, kunnen we de fout bestuderen.
% Dit gebeurt hieronder.

z=y-oplossing23(x); % z = absolute fouten in elk punt
for i=1:length(x)
  nrm(i)=norm(z(:,i),2); % nrm = 2-norm in elk punt van de absolute fouten
end
figure;
plot(x,nrm) % plot van de absolute fouten.
title(['Norm van de absolute fout']);
xlabel('x');
ylabel('absolute fout');



% ------------------------------------------------------------------------

function dydx = probleem23(x,y)
% het rechterlid van het voorbeeld
dydx = [ y(2)
    y(2)*(y(2)-1)/y(1)];

% ------------------------------------------------------------------------

function solut = oplossing23(x)
% de analytische oplossing van het voorbeeld
a=1/8;
b=3/8;
solut = [(a+b*exp(-x/a))
            -b/a*exp(-x/a)];

% ------------------------------------------------------------------------

function dfdy = J23(x,y) %   ! ditmaal hebben we een Jacobiaan nodig
% de jacobiaan van het rechterlid van het voorbeeld
dfdy = [ 0                                  1
       - y(2)*(y(2)-1)/y(1)^2   (2*y(2)-1)/y(1) ];
```

De volledige code in `methodeTheta.m` is

```
function [x,y] = methodeTheta(rhs,jac,h,xspan,y0);
% Theta methode
% toegepast op y'=rhs(x,y) met stap h in interval xspan
x=[xspan(1)];
y=y0;
x0=x(1); % x0 geeft aan tot waar y gekend is
y0=y(:,1); % y0 is op elk moment de laatst berekende y-waarde
xend=xspan(2); % xend is het eindpunt van het integratie-interval
while x0 < xend-h/2
  y1=theta(rhs,x0,y0,jac,h); % bereken y1     ! met Jacobiaan
  y=[y,y1]; % voeg deze y1 toe aan de bestaande y'en
    y0=y1; % y0 is opnieuw de laatst berekende y-waarde
  x0=x0+h; % schuif op naar volgende x
  x=[x,x0]; % voeg dit punt toe aan de lijst van x-waarden
end;

% ------------------------------------------------------------------------

function y=theta(g,x0,y0,jac,h);
% neem 1 eulerstap van grootte h en pas dit toe op y'=rhs(x,y)
x1=x0+h;
y=y0;
delta=1;
ident=eye(length(y0)); % eye(n) is een eenheidsmatrix van orde n

% mogelijk impliciete Methode, vereist bijvoorbeeld Newton's methode
% om de oplossing (het fixpunt) te vinden.
while abs(delta) > 10^(-6)
  theta=2/3; % de theta waarde die we gebruiken
  A=ident-theta*h*feval(jac,x1,y); % feval(rhs,x,y) betekent hetzelfde als rhs(x,y)
  f=-(y-y0-h*(theta * feval(g,x1,y) + (1-theta) * feval(g,x0,y0))); % formule van theta methode
  delta=A\f; % los stelsel A*delta=f op
  y=y+delta; % voeg dit toe om een betere benadering te bekomen
end
```

3. Beschouw de warmtevergelijking (heat equation)

$$u_t = u_{xx}, \qquad 0 \leq x \leq 1, \qquad t >= 0,$$

met de beginvoorwaarde

$$u(0,x) = \cos(2\pi x), \qquad 0 \leq x \leq 1,$$

en de Dirichlet randvoorwaarden

$$u(t,0) = u(t,1) = e^{-4\pi^2 t}.$$

Integreer dit van $t = 0$ tot en met $t = 0.1$. Geef de driedimensionale grafiek van de berekende oplossing. Bepaal ook de maximale fout in de berekende oplossing door dit te vergelijken met de exacte oplossing

$$u(t,x) = \exp(-4\pi^2 t)\cos(2\pi x).$$

Gebruik de roosterafstand $\triangle x \in \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\right\}$ en plot de fout als een functie van $\triangle x$ op een log-log schaal.

Oplossing: We introduceren de mesh punten $x_i = i\triangle x$, $i = 0, \ldots, n+1$ waarbij $\triangle x = 1/(n+1)$ en we vervangen de tweede afgeleide u_{xx} door de differentievergelijking

$$u_{xx}(t,x_i) \approx \frac{u(t,x_{i+1}) - 2u(t,x_i) + u(t,x_{i-1})}{(\triangle x)^2}, \qquad i = 1, \ldots, n,$$

waarbij we de tijdsvariable t continu houden. Zo bekomen we een stelsel van ODEs

$$y_i'(t) = \frac{1}{(\triangle x)^2}(y_{i+1}(t) - 2y_i(t) + y_{i-1}(t)), \qquad i = 1, \ldots, n.$$

Merk op dat we voor de randwaarden bekenden hebben die we kunnen uitschrijven. Dit stelsel van ODEs kunnen we in matrix-vorm schrijven als

$$\boldsymbol{y}' = \frac{1}{(\triangle x)^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}\boldsymbol{y} + \frac{1}{(\triangle x)^2}\begin{bmatrix} e^{-4\pi^2 t} \\ 0 \\ \vdots \\ 0 \\ e^{-4\pi^2 t} \end{bmatrix} = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}.$$

Er zijn een aantal veranderingen nodig aan de code. Allereerst moeten we de functies exact_sol_a en ode_fun_a aanpassen. We krijgen

```
function [f]=ode_fun_a(t,u,A,s,n)
f=A*u;
f(1) = f(1) + s*exp(-4*pi^2*t);
f(n) = f(n) + s*exp(-4*pi^2*t);
```

en

```
function [u]=exact_sol_a(x,t)
u=exp(-4*pi^2*t)*cos(2*pi*x);
```

Bovendien hebben we nu randwaarden verschillend van 0. We moeten hier rekening mee houden bij het bepalen van de absolute fout en de daadwerkelijke (en gekende) randwaarden opgeven. We krijgen

```
U(:,2:n+1)=u;
U(:,1) = exp(-4*pi^2*t);
U(:,n+2) = exp(-4*pi^2*t);
```

De volledige code wordt dan

```
function cp_11_02a
for k=1:4
      n=2^k-1;
      dx(k)=1/(n+1);
      dx(k)
      x=dx(k)*(0:n+1)';
      u0=exact_sol_a(x,0);
      s=1/dx(k)^2;
      A=sparse(diag((-2*s)*ones(n,1))+diag((s)*ones(n-1,1),-1)+...
        diag(s*ones(n-1,1),1));

      options = odeset('RelTol',1e-8,'AbsTol',1e-8*ones(1,n));
      [t,u]=ode23s(@(t,u)ode_fun_a(t,u,A,s,n),[0  0.1],u0(2:n+1),options);
      m=length(t);
      U=zeros(m,n+2);
      U(:,2:n+1)=u;
      U(:,1) = exp(-4*pi^2*t);
      U(:,n+2) = exp(-4*pi^2*t);
      uexact=exact_sol_a(x,0.1);
      err(k)=max(abs(uexact'-U(m,:)));
end
figure;
mesh(t,x,U');
view(110,30);
xlabel('t');
ylabel('x');
zlabel('u');
axis([0 t(length(t)) 0 1 0 1]);
title(sprintf('Comp. probleem 11.2(a) --- Numerieke oplossing ,dt=%5.4f',dx(k)));
figure;
loglog(dx,err,'o-',[10^(-5),1] ,[10^(-10), 1],'--');
xlabel('dx');
ylabel('error');
title('Computer probleem 11.2 (a) --- Max. opl. vs. dx');
axis([10^(-3) 1 10^(-6) 1]);


function [f]=ode_fun_a(t,u,A,s,n)
f=A*u;
f(1) = f(1) + s*exp(-4*pi^2*t);
f(n) = f(n) + s*exp(-4*pi^2*t);

function [u]=exact_sol_a(x,t)
u=exp(-4*pi^2*t)*cos(2*pi*x);
```

4. Vul de lijst van Adams-Bashforth methodes aan met de 5-staps en 6-staps methode die volgt op de vier reeds gegeven methodes en geef de Maple commando's die je gebruikt hebt om deze lijst aan te vullen. We hebben:

$$y_{n+1} - y_n = h f_n \text{ (de Euler methode)}$$
$$y_{n+1} - y_n = \frac{h}{2}(3f_n - f_{n-1})$$
$$y_{n+1} - y_n = \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$$
$$y_{n+1} - y_n = \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

Oplossing: De m-staps Adams-Bashforth methodes worden bekomen door het interpoleren van $\boldsymbol{y}' = \boldsymbol{f}$ door m voorgaande punten en het integreren van de resulterende interpolatieveelterm. We hebben

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \int_{x_k}^{x_{k+1}} \boldsymbol{f}(t, \boldsymbol{y}(t))dx \approx \boldsymbol{y}_k + \int_{x_k}^{x_{k+1}} \boldsymbol{p}(t)dx$$

waarbij $\boldsymbol{p}(t)$ de interpolatieveelterm is door de m punten

$$(t_i, \boldsymbol{f}_i), (t_{i-1}, \boldsymbol{f}_{i-1}), \ldots, (t_{i+1-m}, \boldsymbol{f}_{i+1-m})$$

met $\boldsymbol{f}_i$ een benadering van $\boldsymbol{f}(t_i, \boldsymbol{y}(t_i))$.

De interpolatieveelterm die door deze punten gaat kunnen we eenvoudig bepalen met Maple. Zo gebruiken we voor de interpolatieveelterm door twee punten het commando

```
p := PolynomialInterpolation([x[k]-h, x[k]], [f[k-1], f[k]], xx);
```

om deze interpolatieveelterm te bekomen. Het volstaat vervolgens om deze interpolatieveelterm te integreren tussen x_k en x_{k+1} en het resultaat te vereenvoudigen:

```
i := int(p, xx = x[k] ..  x[k]+h);
simplify(i);
```

Om de 5-staps Adams-Bashforth methode op te stellen gebruiken we de code

```
with(CurveFitting):
p := PolynomialInterpolation([x[k]-4*h, x[k]-3*h, x[k]-2*h, x[k]-h, x[k]], [f[k-4], f[k-3], f[k-2], f[k-1], f[k]], xx);
i := int(p, xx = x[k] ..  x[k]+h);
simplify(i);
```

Dit geeft ons de 5-staps methode

$$y_{n+1} - y_n = \frac{h}{720}\left(1901f_n - 2774f_{n-1} + 2616f_{n-2} - 1274f_{n-3} + 251f_{n-4}\right)$$

De 6-staps Adams-Bashforth methode op te stellen gebruiken we de code

```
with(CurveFitting):
p := PolynomialInterpolation([x[k]-5*h, x[k]-4*h, x[k]-3*h, x[k]-2*h, x[k]-h, x[k]], [f[k-5], f[k-4], f[k-3], f[k-2], f[k-1], f[k]], xx);
i := int(p, xx = x[k] ..  x[k]+h);
simplify(i);
```

Dit geeft ons de 6-staps methode

$$y_{n+1} - y_n = \frac{h}{1440}\left(4277f_n - 7923f_{n-1} + 9982f_{n-2} - 7298f_{n-3} + 2877f_{n-4} - 475f_{n-5}\right)$$

Schrijf uw oplossingen neer op papier. Plaats de bestanden die u eventueel gemaakt hebt om de vragen op te lossen op Minerva. Dit doet u door

- te surfen naar http://indianio/

- in te loggen met uw Minerva-username en -password
- uw bestanden te zippen tot 1 enkel bestand en up te loaden.

Zorg ervoor dat ik uit de naamgeving kan afleiden bij welke vraag elk bestand hoort.

Nog enkele tips :

- schrijf proper : het is de beste manier om te slijmen.
- probeer uw antwoorden bondig en correct te formuleren.
- heb je bij Maple en/of Matlab moeite met (de syntax van) bepaalde commando's, aarzel dan niet dit te vragen.