home / study / math / advanced mathematics / advanced mathematics solutions manuals / numerical analysis / 10th edition / chapter 11.2 / problem 6e

Numerical Analysis (10th Edition)

Chapter 11.2, Problem 6E Bookmark Show all steps: ON

Problem

a. Change Algorithm 11.2 to incorporate the Secant method instead of Newton's method. Use $t_0 = (\beta - \alpha)/(b - a)$ and $t_1 = t_0 + (\beta - y(b, t_0))/(b - a)$.

b. Repeat Exercise 4(a) and 4(c) using the Secant algorithm derived in part (a), and compare the number of iterations required for the two methods.

Reference: Algorithm 11.2

ALGORITHM 11.2

Nonlinear Shooting with Newton's Method

To approximate the solution of the nonlinear boundary-value pro

$$y'' = f(x, y, y'), \quad \text{for } a \le x \le b, \text{ with } y(a) = \alpha \text{ an}$$

(Note: Equations (11.10) and (11.12) are written as first-order s

INPUT endpoints a, b; boundary conditions α, β; number of s ance TOL; maximum number of iterations M.

OUTPUT approximations $w_{1,i}$ to $y(x_i)$; $w_{2,i}$ to $y'(x_i)$ for each i = that the maximum number of iterations was exceeded.

Step 1 Set $h = (b - a)/N$;
$k = 1$;
$TK = (\beta - \alpha)/(b - a)$. *(Note: TK could also be*

Step 2 While ($k \le M$) do Steps 3–10.

Step 3 Set $w_{1,0} = \alpha$;
$w_{2,0} = TK$;
$u_1 = 0$;
$u_2 = 1$.

Step 4 For $i = 1, \ldots, N$ do Steps 5 and 6.
(The Runge-Kutta method for systems is

Step 5 Set $x = a + (i - 1)h$.

Step 6 Set $k_{1,1} = hw_{2,i-1}$;
$k_{1,2} = hf(x, w_{1,i-1} w_{2,i-1})$;
$k_{2,1} = h\left(w_{2,i-1} + \frac{1}{2}k_{1,2}\right)$;
$k_{2,2} = hf\left(x + h/2, w_{1,i-1} + \frac{1}{2}k\right.$
$k_{3,1} = h\left(w_{2,i-1} + \frac{1}{2}k_{2,2}\right)$;
$k_{3,2} = hf\left(x + h/2, w_{1,i-1} + \frac{1}{2}k_{2,1}, w_{2,i-1} + \frac{1}{2}k_{2,}\right.$
$k_{4,1} = h(w_{2,i-1} + k_{3,2})$;
$k_{4,2} = hf(x + h, w_{1,i-1} + k_{3,1}, w_{2,i-1} + k_{3,2})$;
$w_{1,i} = w_{1,i-1} + (k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})/6$;
$w_{2,i} = w_{2,i-1} + (k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2})/6$;
$k'_{1,1} = hu_2$;
$k'_{1,2} = h[f_y(x, w_{1,i-1}, w_{2,i-1})u_1 + f_{y'}(x, w_{1,i-1}, w_{2,i-1})u_2]$;
$k'_{2,1} = h\left[u_2 + \frac{1}{2}k'_{1,2}\right]$;
$k'_{2,2} = h\left[f_y(x + h/2, w_{1,i-1}, w_{2,i-1})\left(u_1 + \frac{1}{2}k'_{1,1}\right) + f_{y'}(x + h/2, w_{1,i-1}, w_{2,i-1})\left(u_2 + \frac{1}{2}k'_{1,2}\right)\right]$
$k'_{3,1} = h\left(u_2 + \frac{1}{2}k'_{2,2}\right)$;
$k'_{3,2} = h\left[f_y(x + h/2, w_{1,i-1}, w_{2,i-1})\left(u_1 + \frac{1}{2}k'_{2,1}\right) + f_{y'}(x + h/2, w_{1,i-1}, w_{2,i-1})\left(u_2 + \frac{1}{2}k'_{2,2}\right)\right]$
$k'_{4,1} = h(u_2 + k'_{3,2})$;
$k'_{4,2} = h\left[f_y(x + h, w_{1,i-1}, w_{2,i-1})\left(u_1 + k'_{3,1}\right) + f_{y'}(x + h, w_{1,i-1}, w_{2,i-1})\left(u_2 + k'_{3,2}\right)\right]$;
$u_1 = u_1 + \frac{1}{6}[k'_{1,1} + 2k'_{2,1} + 2k'_{3,1} + k'_{4,1}]$;
$u_2 = u_2 + \frac{1}{6}[k'_{1,2} + 2k'_{2,2} + 2k'_{3,2} + k'_{4,2}]$.

Step 7 If $|w_{1,N} - \beta| \le TOL$ then do Steps 8 and 9.

Step 8 For $i = 0, 1, \ldots, N$
set $x = a + ih$;
OUTPUT $(x, w_{1,i}, w_{2,i})$.

Step 9 *(The procedure is complete.)*
STOP.

Step 10 Set $TK = TK - \dfrac{w_{1,N} - \beta}{u_1}$;
(Newton's method is used to compute TK.)
$k = k + 1$.

Step 11 OUTPUT ('Maximum number of iterations exceeded');
(The procedure was unsuccessful.)
STOP.

Reference: Exercise 4(a)

a. $y'' = y^3 - yy'$, $1 \le x \le 2$, $y(1) = \frac{1}{2}$, $y(2) = \frac{1}{3}$; use $h = 0.1$; actual solution $y(x) = (x+1)^{-1}$.

Reference: Exercise 4(c)

c. $y'' = y' + 2(y - \ln x)^3 - x^{-1}$, $2 \le x \le 3$, $y(2) = \frac{1}{2} + \ln 2$, $y(3) = \frac{1}{3} + \ln 3$; use $h = 0.1$; actual solution $y(x) = x^{-1} + \ln x$.

Step-by-step solution

Step 1 of 22

(a)

Write the nonlinear boundary value problem $y'' = f(x, y, y'), a \le x \le b$ with boundary conditions $y(a) = \alpha$ and $y(b) = \beta$.

The modified Nonlinear Shooting algorithm to incorporate the secant method with $t_0 = (\beta - \alpha)/(b - a)$ and $t_1 = t_0 + (\beta - y(b, t_0))/(b - a)$ is as follows:

INPUT endpoints a, b; boundary conditions α, β; number of subintervals $N \ge 2$;

Tolerance TOL; maximum number of iterations M.

OUTPUT approximations $w_{1,i}$ to $y(x_i)$; $w_{2,i}$ to $y'(x_i)$ for each $i = 0, 1, 2, \ldots, N$ or a message that the maximum number of iterations was exceeded.

Step1 Set $h = (b - a)/N$; $k = 2$; $TK1 = (\beta - \alpha)/(b - a)$.

Step2 Set $w_{1,0} = \alpha$; $w_{2,0} = TK1$.

Step3 For $i = 1, 2, \ldots, N$ perform Steps 4 to 7.

Step4 Set $x = a + (i - 1)h$.

Step5 Set
$k_{1,1} = h \cdot w_{2,i-1}$;
$k_{1,2} = h \cdot f(x, w_{1,i-1}, w_{2,i-1})$;
$k_{2,1} = h \cdot \left(w_{2,i-1} + \frac{k_{1,2}}{2}\right)$;
$k_{2,2} = h \cdot f\left(x + \frac{h}{2}, w_{1,i-1} + \frac{k_{1,1}}{2}, w_{2,i-1} + \frac{k_{1,2}}{2}\right)$.

Comment

Step 2 of 22

Continuation of the algorithm:

Step6 Set
$k_{3,1} = h \cdot \left(w_{2,i-1} + \frac{k_{2,2}}{2}\right)$;
$k_{3,2} = h \cdot f\left(x + \frac{h}{2}, w_{1,i-1} + \frac{k_{2,1}}{2}, w_{2,i-1} + \frac{k_{2,2}}{2}\right)$;
$k_{4,1} = h \cdot \left(w_{2,i-1} + \frac{k_{3,2}}{2}\right)$;
$k_{4,2} = h \cdot f\left(x + \frac{h}{2}, w_{1,i-1} + k_{3,1}, w_{2,i-1} + k_{3,2}\right)$.

Step7 Set
$w_{1,i} = w_{1,i-1} + \frac{1}{6}(k_{1,1} + 2 \cdot k_{2,1} + 2 \cdot k_{3,1} + k_{4,1})$;
$w_{2,i} = w_{2,i-1} + \frac{1}{6}(k_{1,2} + 2 \cdot k_{2,2} + 2 \cdot k_{3,2} + k_{4,2})$.

Step8 Set $TK2 = TK1 + \dfrac{(\beta - w_{1,N})}{(b - a)}$.

Step9 while $(k \le M)$ perform Steps 10 to 17

Step10 Set $w_{2,0} = TK2$; $HOLD = w_{1,N}$.

Comment

Step 3 of 22

Continuation of the algorithm:

Step11 For $i = 1, 2, 3, \ldots, N$ perform Steps 12 to 15

Step12 Set $x = a + (i - 1)h$.

Step13 Set
$k_{1,1} = h \cdot w_{2,i-1}$;
$k_{1,2} = h \cdot f(x, w_{1,i-1}, w_{2,i-1})$;
$k_{2,1} = h \cdot \left(w_{2,i-1} + \frac{k_{1,2}}{2}\right)$;
$k_{2,2} = h \cdot f\left(x + \frac{h}{2}, w_{1,i-1} + \frac{k_{1,1}}{2}, w_{2,i-1} + \frac{k_{1,2}}{2}\right)$.

[illegible]

```
> #
> # To approximate the solution of the nonlinear boundary-value problem
> #
> #        Y'' = F(X,Y,Y'), A <=X<=B, Y(A) = ALPHA, Y(B) = BETA:
> #
> # INPUT:   Endpoints A,B: boundary conditions ALPHA, BETA: number of
> #          subintervals N: tolerance TOL: maximum number of iterations M.
> #
> # OUTPUT:  Approximations W(1,I) TO Y(X(I)): W(2,I) TO Y'(X(I))
> #          for each I=0,1,...,N or a message that the maximum
> #          number of iterations was exceeded.
```

Comment

Step 7 of 22

Continuation of the above **maple code:**

```
> print(`This is the Nonlinear Shooting with Newton's Method.`) :
> print(`Input the function F(X,Y,Z) in terms of x, y, z.`) :
> F := scanf(`%a`)[1] : print(`F(x,y,z) = `) : print(F) :
> FY := diff(F, y) :
> FYP := diff(F, z) :
> F := unapply(F, x, y, z) :
> FY := unapply(FY, x, y, z) :
> FYP := unapply(FYP, x, y, z) :
> OK := FALSE :
> while OK=FALSE do
> print(`Input left and right endpoints separated by blank.`) :
```

Comment

Step 8 of 22

Continuation of the above **maple code:**

```
> A := scanf(`%e`)[1] : print(`Left endpoint = `) : print(A) :
> B := scanf(`%e`)[1] : print(`Right endpoint = `) : print(B) :
> if A ≥ B then
> print(`Left endpoint must be less than right endpoint.`) :
> else OK := TRUE :
> fi:
> od:
> print(`Input Y(A).`) :
> ALPHA := scanf(`%e`)[1] : print(`alpha = `) : print(ALPHA) :
> print(`Input Y(B).`) :
> BETA := scanf(`%e`)[1] : print(`beta = `) : print(BETA) :
> TK := (BETA-ALPHA)/(B-A) :
> print(`TK := `) : print(TK) :
> print(`Input new TK? Enter 1 for yes or 2 for no.`) :
> AA := scanf(`%d`)[1] : print(`Input is `) : print(AA) :
> if AA=1 then
> print(`Input new TK`) :
> TK := scanf(`%f`)[1] : print(`New TK = `) : print(TK) :
```

Continuation of the above **maple code:**

```
> fi:
> OK := FALSE :
> while OK=FALSE do
> print(`Input an integer > 1 for the number of subintervals.`) :
> N := scanf(`%d`)[1] : print(`n = `) : print(N) :
> if N < 1 then
> print(`Number must exceed 1.`) :
> else
> OK := TRUE :
> fi:
> od:
> OK := FALSE :
> while OK=FALSE do
```

Comment

Step 9 of 22

Continuation of the above **maple code:**

```
> print(`Input Tolerance.`) :
> TOL := scanf(`%f`)[1] : print(`Tolerance = `) : print(TOL) :
> if TOL ≤ 0 then
> print(`Tolerance must be positive.`) :
> else
> OK := TRUE :
> fi:
> od:
> OK := FALSE :
> while OK=FALSE do
> print(`Input maximum number of iterations.`) :
> NN := scanf(`%d`)[1] : print(`Maximum number of iterations = `) : print(NN) :
> if NN ≤ 0 then
> print(`Must be positive integer.`) :
> else
> OK := TRUE :
> fi:
> od:
```

Comment

Step 10 of 22

Continuation of the above **maple code:**

```
> if OK=TRUE then
> print(`Choice of output method:`) :
> print(`1. Output to screen`) :
> print(`2. Output to text File`) :
> print(`Please enter 1 or 2.`) :
> FLAG := scanf(`%d`)[1] : print(`Input is `) : print(FLAG) :
> if FLAG=2 then
> print(`Input the file name in the form - drive:\\name.ext`) :
> print(`for example  A:\\OUTPUT.DTA`) :
> NAME := scanf(`%s`)[1] : print(`Output file is `) : print(NAME) :
> OUP := fopen(NAME, WRITE, TEXT) :
> else
> OUP := default :
> fi:
> fprintf(OUP, `NONLINEAR SHOOTING WITH SECANT METHOD\n\n`) :
> fprintf(OUP, `  I   X(I)       W1(I)       W2(I)\n`) :
```

Comment

Step 11 of 22

Continuation of the above **maple code:**

```
> # Step 1
> H := (B-A)/N :
> K := 2 :
> # TK is already computed.
> OK := FALSE :
> # Step 2
> while K ≤ NN and OK=FALSE do
> # Step 3
> W1[0] := ALPHA :
> W2[0] := TK :
> U1 := 0 :
> U2 := 1 :

> # Step 4
> # Runge-Kutta method for systems is used in Steps 5 and 6
> for I2 from 1 to N do
> # Step 5
> X := A + (I2-1)*H :
> T := X + 0.5*H :
```

Comment

Step 12 of 22

```
> # Step 6
> K11 := H*W2[I2-1] :
> K12 := H*F(X, W1[I2-1], W2[I2-1]) :
> K21 := H*(W2[I2-1] + 0.5*K12) :
> K22 := H*F(T, W1[I2-1] + 0.5*K11, W2[I2-1] + 0.5*K12) :
> K31 := H*(W2[I2-1] + 0.5*K22) :
> K32 := H*F(T, W1[I2-1] + 0.5*K21, W2[I2-1] + 0.5*K22) :
> K41 := H*(W2[I2-1] + K32) :
> K42 := H*F(X + H, W1[I2-1] + K31, W2[I2-1] + K32) :
> W1[I2] := W1[I2-1] + (K11 + 2*(K21 + K31) + K41)/6 :
> W2[I2] := W2[I2-1] + (K12 + 2*(K22 + K32) + K42)/6 :
```

Continuation of the above **maple code:**

```
> K11 := H*U2 :
> K12 := H*(FY(X, W1[I2-1], W2[I2-1])*U1 + FYP(X, W1[I2-1], W2[I2-1])*U2) :
> K21 := H*(U2 + 0.5*K12) :
> K22 := H*(FY(T, W1[I2-1], W2[I2-1])*(U1 + 0.5*K11) + FYP(T, W1[I2-1], W2[I2
  -
     1])*(U2 + 0.5*K21)) :
> K31 := H*(U2 + 0.5*K22) :
> K32 := H*(FY(T, W1[I2-1], W2[I2-1])*(U1 + 0.5*K21) + FYP(T, W1[I2-1], W2[I2
     1])*(U2 + 0.5*K22)) :
> K4 := H*(U2 + K32) :
> K42 := H*(FY(X + H, W1[I2-1], W2[I2-1])*(U1 + K31) + FYP(X + H, W1[I2-1],
  W2[I2-
     1])*(U2 + K32)) :
> U1 := U1 + (K11 + 2*(K21 + K31) + K41)/6 :
> U2 := U2 + (K12 + 2*(K22 + K32) + K42)/6 :
> od:
```

Comment

Step 13 of 22

Continuation of the above **maple code:**

```
> # Step 7
> # Test for accuracy
> if abs(W1[N]-BETA) < TOL then
> # Step 8
> I2 := 0 :
> fprintf(OUP, `%3d %13.8f %13.8f %13.8f\n`, I2, A, ALPHA, TK) :
> for I2 from 1 to N do
> J := I2 + 1 :
> X := A + I2*H :
> fprintf(OUP, `%3d %13.8f %13.8f %13.8f\n`, I2, X, W1[J-1], W2[J-1]) :
> od:
> fprintf(OUP, `Convergence in %d iterations\n`, K) :
> fprintf(OUP, `t = %14.7e\n`, TK) :
```

```
> # Step 10
> # Secant method is applied to improve TK
> TK := TK-(W1[N]-BETA)/U1 :
> K := K + 1 :
> fi:
> od:
> # Step 11
> # Procedure completed unsuccessfully.
> if OK=FALSE then
> fprintf(OUP, `Method failed after %d iterations\n`, NN) :
> fi:
> fi;
> if OUP ≠ default then
> fclose(OUP) :
> print(`Output file `, NAME, ` created successfully`) :
> fi:
```

Comment

Step 14 of 22

To debug the maple code, keep the cursor near the **restart command** and hit the enter key. The maple response will be in the form of a pop-up and follow the instructions below the pop-up to enter the relevant maple commands as shown below:

This is the Nonlinear Shooting with Secant Method.
Input the function F(X,Y,Z) in terms of x, y, z.
F(x,y,z) =
$y^3 - yz$
Input left and right endpoints separated by blank.
Left endpoint =
1.
Right endpoint =
2.
Input Y(A).
alpha =
0.5
Input Y(B).
beta =
0.3333333333

Comment

Step 15 of 22

Continuation of the above:

TK :=
−0.1666666667
Input new TK? Enter 1 for yes or 2 for no.
Input is
2

Comment

Step 16 of 22

Input an integer > 1 for the number of subintervals.
n =
10
Input Tolerance.
Tolerance =
0.0001

Continuation of the above:

Input maximum number of iterations.
Maximum number of iterations =
100
Choice of output method:
1. Output to screen
2. Output to text File
Please enter 1 or 2.
Input =
1

NONLINEAR SHOOTING WITH SECANT METHOD
I X(I) W1(I) W2(I)
0 1.00000000 0.50000000 -0.24999738
1 1.10000000 0.47619079 -0.22675486
2 1.20000000 0.45454606 -0.20660913
3 1.30000000 0.43478349 -0.18903352
4 1.40000000 0.41666781 -0.17360875
5 1.50000000 0.40000140 -0.15999765
6 1.60000000 0.38461703 -0.14792664
7 1.70000000 0.37037226 -0.13717184
8 1.80000000 0.35714500 -0.12754863
9 1.90000000 0.34482998 -0.11890365
10 2.00000000 0.33333597 -0.11110866
Convergence in 4 iterations
t = -2.4999738e-01

Comment

Step 17 of 22

Compute the actual solution by using the Maple technology as follows:

> **for** x **from** 1 **by** 0.1 **to** 2 **do** $y_x := evalf\left(\frac{1}{1+x}\right)$ **end do;**

$y_1 := 0.5000000000$
$y_{1.1} := 0.4761904762$
$y_{1.2} := 0.4545454545$
$y_{1.3} := 0.4347826087$
$y_{1.4} := 0.4166666667$
$y_{1.5} := 0.4000000000$
$y_{1.6} := 0.3846153846$
$y_{1.7} := 0.3703703704$
$y_{1.8} := 0.3571428571$
$y_{1.9} := 0.3448275862$
$y_{2.0} := 0.3333333333$

Result: The approximate solution requires **4 iterations** for convergence and the actual Solutions at some typical points are shown below:

i	x_i	w_i	$y(x_i)$
2	1.2	0.45454606	0.4545454545
4	1.4	0.41666781	0.4166666667
6	1.6	0.38461703	0.3846153846
8	1.8	0.35714500	0.3571428571

Comment

Step 18 of 22

4(c)

Write the Nonlinear boundary value Problem $y'' = y' + 2(y - \ln x)^3 - x^{-1}, 2 \le x \le 3$ with boundary conditions $y(2) = \left(\frac{1}{2} + \ln 2\right)$ and $y(3) = \left(\frac{1}{3} + \ln 3\right)$.

The actual solution is $y(x) = (\ln x + x^{-1})$.

Choose $h = 0.1$, then $N = 10, TOL = 10^{-4}$, and $M = 100$.

It is required to approximate solution to the following initial value problem involving a parameter t with $t = t_k$:

$y'' = y' + 2(y - \ln x)^3 - x^{-1}, 2 \le x \le 3$ with boundary conditions $y(2) = 0$ and $y'(2) = t_k$.

and $z'' = \left(\frac{\partial f}{\partial y}\right) z + \left(\frac{\partial f}{\partial y'}\right) z' = 2z + z'$ for $2 \le x \le 3$ with $z(2) = 0$ and $z'(2) = 1$.

Comment

Step 19 of 22

To debug the maple code, keep the cursor near the **restart command** and hit the enter key. The maple response will be in the form of a pop-up and follow the instructions below the pop-up to enter the relevant maple commands as shown below:

This is the Nonlinear Shooting with Secant Method.
Input the function F(X,Y,Z) in terms of x, y, z.
F(x,y,z) =

Comment

Step 20 of 22

$z + 2(y - \ln(x))^3 - \frac{1}{x}$
Input left and right endpoints separated by blank.
Left endpoint =
2.
Right endpoint =
3.

Continuation of the above:

Input Y(A).
alpha =
1.193147181
Input Y(B).
beta =
1.431945622
TK :=
0.2307994410
Input new TK? Enter 1 for yes or 2 for no.
Input is
2
Input an integer > 1 for the number of subintervals.
n =
10

Comment

Step 21 of 22

Continuation of the above:

Input Tolerance.
Tolerance =
0.0001
Input maximum number of iterations.
Maximum number of iterations =
100
Choice of output method:
1. Output to screen
2. Output to text File
Please enter 1 or 2.
Input =
1

NONLINEAR SHOOTING WITH SECANT METHOD
I X(I) W1(I) W2(I)
0 2.00000000 1.19314718 0.24999821
1 2.10000000 1.21812768 0.24943122
2 2.20000000 1.24300252 0.24793184
3 2.30000000 1.26769124 0.24574444
4 2.40000000 1.29213470 0.24305303
5 2.50000000 1.31628977 0.23999716
6 2.60000000 1.34012558 0.23668317
7 2.70000000 1.36362056 0.23319249
8 2.80000000 1.38676031 0.22958764
9 2.90000000 1.40953591 0.22591673
10 3.00000000 1.43194270 0.22221674
Convergence in 4 iterations
t = 2.4999821e-01

Comment

Step 22 of 22

Compute the actual solution by using the Maple technology as follows:

> **for** x **from** 2 **by** 0.1 **to** 3 **do** $y_x := evalf\left(\ln(x) + \frac{1}{x}\right)$ **end do;**

$y_2 := 1.193147181$
$y_{2.1} := 1.218127821$
$y_{2.2} := 1.243002815$
$y_{2.3} := 1.267691732$
$y_{2.4} := 1.292135404$
$y_{2.5} := 1.316290732$
$y_{2.6} := 1.340126830$
$y_{2.7} := 1.363622143$
$y_{2.8} := 1.386762274$
$y_{2.9} := 1.409538323$
$y_{3.0} := 1.431945622$

Result: The approximate solution requires **4 iterations** for convergence and the actual Solutions at some typical points are shown below:

i	x_i	w_i	$y(x_i)$
2	2.2	1.24300252	1.243002815
4	2.4	1.29213470	1.292135404
6	2.6	1.34012558	1.340126830
8	2.8	1.38676031	1.386762274

Comment

Recommended solutions for you in Chapter 11.2