Numerical Analysis (10th Edition)

Chapter 12.2, Problem 13E Bookmark Show all steps: ON

Problem

The temperature $u(x, t)$ of a long, thin rod of constant cross section and homogeneous conducting material is governed by the one-dimensional heat equation. If heat is generated in the material, for example, by resistance to current or nuclear reaction, the heat equation becomes

$$\frac{\partial^2 u}{\partial x^2} + \frac{Kr}{\rho C} = K\frac{\partial u}{\partial t}, \quad 0 < x < l, \quad 0 < t,$$

where l is the length, ρ is the density, C is the specific heat, and K is the thermal diffusivity of the rod. The function $r = r(x, t, u)$ represents the heat generated per unit volume. Suppose that $l = 1.5$ cm, $K = 1.04$ cal/cm · deg · s, $\rho = 10.6$ g/cm^3, $C = 0.056$ cal/g · deg, and $r(x, t, u) = 5.0$ cal/cm^3 · s.

If the ends of the rod are kept at 0 ? C, then $u(0, t) = u(l, t) = 0$, $t > 0$.

Suppose the initial temperature distribution is given by

$$u(x,0) = \sin\frac{\pi x}{l}, \quad 0 \le x \le l.$$

Reference: Excercise 15:

Use the results of Exercise 15 to approximate the temperature distribution with $h = 0.15$ and $k = 0.0225$

Modify Algorithms 12.2 and 12.3 to include the parabolic partial differential equation

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = F(x), \quad 0 < x < l, \; 0 < t;$$
$$u(0,t) = u(l,t) = 0, \quad 0 < t;$$
$$u(x,0) = f(x), \quad 0 \le x \le l.$$

Reference: Algorithms 12.2

ALGORITHM 12.2

Heat Equation Backward-Difference

To approximate the solution to the parabolic partial differential e

$$\frac{\partial u}{\partial t}(x,t) - \alpha^2\frac{\partial^2 u}{\partial x^2}(x,t) = 0, \quad 0 < x < l, \quad 0 <$$

subject to the boundary conditions

$$u(0,t) = u(l,t) = 0, \quad 0 < t < T,$$

and the initial conditions

$$u(x,0) = f(x), \quad 0 \le x \le l:$$

INPUT endpoint l; maximum time T; constant α; integers $m \ge 3$, $N \ge 1$.

OUTPUT approximations $w_{i,j}$ to $u(x_i, t_j)$ for each $i = 1, \dots, m-1$ and $j = 1, \dots, N$.

Step 1 Set $h = l/m$; $k = T/N$; $\lambda = \alpha^2 k/h^2$.

Step 2 For $i = 1, \dots, m-1$ set $w_i = f(ih)$. (*Initial values.*)
(*Steps 3–11 solve a tridiagonal linear system using Algorithm 6.7.*)

Step 3 Set $l_1 = 1 + 2\lambda$; $u_1 = -\lambda/l_1$.

Step 4 For $i = 2, \dots, m-2$ set $l_i = 1 + 2\lambda + \lambda u_{i-1}$; $u_i = -\lambda/l_i$.

Step 5 Set $l_{m-1} = 1 + 2\lambda + \lambda u_{m-2}$.

Step 6 For $j = 1, \dots, N$ do Steps 7–11.

Step 7 Set $t = jk$; (*Current* t_j.) $z_1 = w_1/l_1$.

Step 8 For $i = 2, \dots, m-1$ set $z_i = (w_i + \lambda z_{i-1})/l_i$.

Step 9 Set $w_{m-1} = z_{m-1}$.

Step 10 For $i = m-2, \dots, 1$ set $w_i = z_i - u_i w_{i+1}$.

Step 11 OUTPUT (t); (*Note:* $t = t_j$.)
For $i = 1, \dots, m-1$ set $x = ih$;
OUTPUT (x, w_i). (*Note:* $w_i = w_{i,j}$.)

Step 12 STOP. (*The procedure is complete.*)

Reference: Algorithms 12.3

ALGORITHM 12.3

Crank-Nicolson

To approximate the solution to the parabolic partial differential

$$\frac{\partial u}{\partial t}(x,t) - \alpha^2\frac{\partial^2 u}{\partial x^2}(x,t) = 0, \quad 0 < x < l, \quad 0$$

subject to the boundary conditions

$$u(0,t) = u(l,t) = 0, \quad 0 < t < T,$$

and the initial conditions

$$u(x,0) = f(x), \quad 0 \le x \le l:$$

INPUT endpoint l; maximum time T; constant α; integers m

OUTPUT approximations $w_{i,j}$ to $u(x_i, t_j)$ for each $i = 1, \dots, n$

Step 1 Set $h = l/m$; $k = T/N$; $\lambda = \alpha^2 k/h^2$; $w_m = 0$.

Step 2 For $i = 1, \dots, m-1$ set $w_i = f(ih)$. (*Initial values.*)
(*Steps 3–11 solve a tridiagonal linear system using Algorithm 6.7.*)

Step 3 Set $l_1 = 1 + \lambda$; $u_1 = -\lambda/(2l_1)$.

Step 4 For $i = 2, \dots, m-2$ set $l_i = 1 + \lambda + \lambda u_{i-1}/2$; $u_i = -\lambda/(2l_i)$.

Step 5 Set $l_{m-1} = 1 + \lambda + \lambda u_{m-2}/2$.

Step 6 For $j = 1, \dots, N$ do Steps 7–11.

Step 7 Set $t = jk$; (*Current* t_j.)
$$z_1 = \left[(1-\lambda)w_1 + \frac{\lambda}{2}w_2\right]\Big/ l_1.$$

Step 8 For $i = 2, \dots, m-1$ set
$$z_i = \left[(1-\lambda)w_i + \frac{\lambda}{2}(w_{i+1} + w_{i-1} + z_{i-1})\right]\Big/$$

Step 9 Set $w_{m-1} = z_{m-1}$.

Step 10 For $i = m-2, \dots, 1$ set $w_i = z_i - u_i w_{i+1}$.

Step 11 OUTPUT (t); (*Note:* $t = t_j$.)
For $i = 1, \dots, m-1$ set $x = ih$;
OUTPUT (x, w_i). (*Note:*

Step 12 STOP. (*The procedure is complete.*)

Step-by-step solution

Step 1 of 23

The 1-D heat equation of a long, thin rod of constant cross-section and homogeneous conducting material, and the conditions which govern it, are as follows:

$$\frac{\partial^2 u(x,t)}{\partial x^2} + \frac{Kr}{\rho C} = K\frac{\partial u(x,t)}{\partial t}, \quad \text{for } 0 < x < l \text{ and } t > 0.$$

Here, l is the length of the rod, ρ is the density, C is the specific heat, K is the thermal diffusivity of the rod, and $r = r(x,t,u)$ is the heat generated per unit volume respectively.

Comment

Step 2 of 23

The parameters are defined as shown below:

$$\left\{\begin{array}{l} r = r(x,t,u) \\ l = 1.5 \text{ cm}, \\ K = 1.04 \text{ cal/cm·deg·s}, \\ \rho = 10.6 \text{ g/cm}^3, \\ C = 0.056 \text{ cal/cm·deg, and} \\ r(x,t,u) = 5.0 \text{ cal/cm}^3\text{·s} \end{array}\right\}$$

The required parabolic partial differential equation becomes,

$$\frac{\partial u(x,t)}{\partial t} - \frac{1}{K}\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{r}{\rho C}, \quad \text{for } 0 < x < l \text{ and } t > 0.$$
$$\frac{\partial u(x,t)}{\partial t} - \frac{1}{1.04}\frac{\partial^2 u(x,t)}{\partial x^2} = 8.423180593 \quad \text{for } 0 < x < 1.5 \text{ and } t > 0$$

Comment

Step 3 of 23

Assume that the ends of the rod are kept at $0^\circ C$ and the initial temperature distribution is,

$$u(x,0) = \sin\left(\frac{\pi x}{l}\right) \text{ for } 0 \le x \le l$$

In that case, the boundary conditions are,

$$u(0,t) = u(l,t) = 0 \text{ for } t > 0$$

Use of parameters leads to,

$$u(0,t) = u(1.5,t) = 0 \quad \text{for } t > 0$$
$$u(x,0) = \sin\left(\frac{\pi x}{1.5}\right) \quad \text{for } 0 \le x \le 1.5$$

Comment

Step 4 of 23

Use the **Modified Backward Difference Algorithm** to approximate the temperature distribution with $h = 0.15$ and $k = 0.0225$ as follows:

Set $T = 0.225$ and $\alpha = \sqrt{\frac{1}{1.04}} = 0.9805806757$

The Modified Backward-Difference Algorithm with Maple technology is shown below:

```
> restart:
>
  # HEAT EQUATION MOFIFIED BACKWARD-DIFFERENCE
  ALGORITHM
> # To approximate the solution to the parabolic
  partial-differential
> # equation subject to the boundary conditions
> #     u(0,t) = u(l,t) = 8.423180593, 0 < t < T = max t,
> # and the initial conditions
> #          u(x,0) = F(x), 0 <= x <= l:
> # INPUT:  endpoint l: maximum time T: constant ALPHA: integers
     m, N.
> # OUTPUT:  approximations W(I,J) to u(x(I),t(J)) for each
> #        I = 1, ..., m-1 and J = 1, ..., N.
```

Comment

Step 5 of 23

Continuation of the above is as follows:

```
> print( `This is the Modified Backward-Difference Method for Heat
    Equation.`) :
> print( `Input the function F(X) in terms of x.`) :
> print( `For example:  sin(3.141592654*x)`) :
> F := scanf( `%a`)[1] : print( `F(x) = `) : print(F) :
> F := unapply(F, x) :
> print( `The lefthand endpoint on the X-axis is 0.`) :
> OK := FALSE:
> while OK = FALSE do
> print( `Input the righthand endpoint on the X-axis.`) :
> FX := scanf( `%f`)[1] : print( `Righthand endpoint = `) : print(FX) :
> if FX ≤ 0 then
> print( `Must be positive number.`) :
> else
> OK := TRUE:
> fi:
> od:
```

Comment

Step 6 of 23

Continuation of the above is as follows:

```
> OK := FALSE:
> while OK = FALSE do
> print( `Input the maximum value of the time variable T.`) :
> FT := scanf( `%f`)[1] : print( `Maximum time value = `) : print(FT) :
> if FT ≤ 0 then
> print( `Must be positive number.`) :
> else
> OK := TRUE:
> fi:
> od:
> print( `Input the constant alpha.`) :
> ALPHA := scanf( `%f`)[1] : print( `alpha = `) : print(ALPHA) :
> OK := FALSE:
> while OK = FALSE do
> print( `Input integer m = number of intervals on X-axis`) :
> print( `and N = number of time intervals - separated by a
    blank.`) :
> print( `Note that m must be 3 or larger.`) :
```

Comment

Step 7 of 23

Continuation of the above is as follows:

```
> M := scanf( `%d`)[1] :
> N := scanf( `%d`)[1] :
> print( `Number of intervals on x-axis = `) : print(M) :
> print( `Number of time intervals = `) : print(N) :
> if M ≤ 2 or N ≤ 0 then
> print( `Numbers are not within correct range.`) :
> else
> OK := TRUE:
> fi:
> od:
> if OK = TRUE then
> M1 := M-1 :
> M2 := M-2 :
> N1 := N-1 :
```

Comment

Step 8 of 23

Continuation of the above is as follows:

```
> # Step 1
> H := FX/M:
> K := FT/N:
> VV := ALPHA*ALPHA*K/(H*H) :
> # Step 2
> for I2 from 1 to M1 do
> W[I2-1] := F(I2*H) :
> od:
> # Step 3
> # Steps 3 - 11 solve a tridiagonal linear system using
  Algorithm 6.7
> L[0] := 1 + 2*VV:
> U[0] := -VV/L[0] :
```

Comment

Step 9 of 23

Continuation of the above is as follows:

```
> # Step 4
> for I2 from 2 to M2 do
> L[I2-1] := 1 + 2*VV + VV*U[I2-2] :
> U[I2-1] := -VV/L[I2-1] :
> od:
> # Step 5
> L[M1-1] := 1 + 2*VV + VV*U[M2-1] :
> # Step 6
> for J from 1 to N do
> # Step 7
> # Current t
> T := J*K :
> Z[0] := W[0]/L[0] :
```

Comment

Step 10 of 23

Continuation of the above is as follows:

```
> # Step 8
> for I2 from 2 to M1 do
> Z[I2-1] := (W[I2-1] + VV*Z[I2-2])/L[I2-1] :
> od:
> # Step 9
> W[M1-1] := evalf(Z[M1-1]) :
> # Step 10
> for I1 from 1 to M2 do
> I2 := M2-I1 + 1 :
> W[I2-1] := evalf(Z[I2-1]-U[I2-1]*W[I2]) :
> od:
> od:
```

Comment

Step 11 of 23

Continuation of the above is as follows:

```
> # Step 11
> 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:
```

Comment

Step 12 of 23

Continuation of the above is as follows:

```
> fprintf(OUP, `MODIFIED BACKWARD-DIFFERENCE METHOD\n\n`) :
> fprintf(OUP, ` I        X(I)     W(X(I),%12.6e)\n`, FT) :
> for I2 from 1 to M1 do
> X := I2*H:
> fprintf(OUP, `%3d %11.8f %14.8f\n`, I2, X, W[I2-1]) :
> od:
> fi:
> if OUP ≠ default then
> fclose(OUP) :
> print( `Output file `, NAME, ` created successfully`) :
> fi:
```

Comment

Step 13 of 23

Hit the enter key near restart of the Maple Algorithm and proceed to input the data as per the instructions in the maple pop-up as shown below:

This is the Modified Backward-Difference Method for Heat Equation.

Input the function F(X) in terms of x.

*For example: sin(3.141592654*x)*

F(x) =

$\sin(0.6666666667\,\pi x)$

The lefthand endpoint on the X-axis is 0.

Input the righthand endpoint on the X-axis.

Righthand endpoint =

1.5

Input the maximum value of the time variable T.

Maximum time value =

0.225

Input the constant alpha.

alpha =

0.9805806757

Comment

Step 14 of 23

Continuation of the above is as follows:

Input integer m = number of intervals on X-axis

and N = number of time intervals - separated by a blank.

Note that m must be 3 or larger.

Number of intervals on x-axis =

10

Number of time intervals =

10

Choice of output method:

1. Output to screen

2. Output to text file

Please enter 1 or 2.

Input is

1

Comment

Step 15 of 23

MODIFIED BACKWARD-DIFFERENCE METHOD:

I X(I) W(X(I),2.250000e-01)

1 0.15000000 0.12569689

2 0.30000000 0.23908969

3 0.45000000 0.32907872

4 0.60000000 0.38685524

5 0.75000000 0.40676367

6 0.90000000 0.38685524

7 1.05000000 0.32907872

8 1.20000000 0.23908969

9 1.35000000 0.12569689

Result: Thus, the required approximate solution is the final Maple output.

Use the **Modified Crank-Nicolson Algorithm** to approximate the temperature distribution as follows:

```
> restart:
> # MODIFIED CRANK-NICOLSON ALGORITHM
> # To approximate the solution of the parabolic partial-
  differential
> # equation subject to the boundary conditions
> #          u(0,t) = u(l,t) = 8.423180593, 0 < t < T = max t
> # and the initial conditions
> #          u(x,0) = F(x), 0 <= x <= l:
> # INPUT:   endpoint l: maximum time T: constant ALPHA: integers
       m, N:
> # OUTPUT:  approximations W(I,J) to u(x(I),t(J)) for each
> #        I = 1,..., m-1 and J = 1,..., N.
```

Comment

Step 16 of 23

Continuation of the above is as follows:

```
> print( `This is the Modified Crank-Nicolson Method.`) :
> print( `Input the function F(X) in terms of x.`) :
> print( `For example:   sin(3.141592654*x)`) :
> F := scanf( `%a`)[1] : print( `F(x) = `) : print(F) :
> F := unapply(F, x) :
> print( `The lefthand endpoint on the X-axis is 0.`) :
> OK := FALSE:
> while OK = FALSE do
> print( `Input the righthand endpoint on the X-axis.`) :
> FX := scanf( `%f`)[1] : print( `Righthand endpoint = `) : print(FX) :
> if FX ≤ 0 then
> print( `Must be positive number.`) :
> else
> OK := TRUE:
> fi:
> od:
```

Comment

Step 17 of 23

Continuation of the above is as follows:

```
> OK := FALSE:
> while OK = FALSE do
> print( `Input the maximum value of the time variable T.`) :
> FT := scanf( `%f`)[1] : print( `Maximum time value = `) : print(FT) :
> if FT ≤ 0 then
> print( `Must be positive number.`) :
> else
> OK := TRUE:
> fi:
> od:
> print( `Input the constant alpha.`) :
> ALPHA := scanf( `%f`)[1] : print( `alpha = `) : print(ALPHA) :
> OK := FALSE:
> while OK = FALSE do
> print( `Input integer m = number of intervals on X-axis`) :
> print( `and N = number of time intervals - separated by a
   blank.`) :
> print( `Note that m must be 3 or larger.`) :
```

Comment

Step 18 of 23

Continuation of the above is as follows:

```
> M := scanf( `%d`)[1] :
> N := scanf( `%d`)[1] :
> print( `Number of intervals on x-axis = `) : print(M) :
> print( `Number of time intervals = `) : print(N) :
> if M ≤ 2 or N ≤ 0 then
> print( `Numbers are not within correct range.`) :
> else
> OK := TRUE:
> fi:
> od:
> if OK = TRUE then
> M1 := M-1 :
> M2 := M-2 :
```

Comment

Step 19 of 23

Continuation of the above is as follows:

```
> # Step 1
> H := FX/M:
> K := FT/N:
> # VV is used in place of lambda
> VV := ALPHA^2 * K/(H^2) :
> # Set V(M) to zero
> V[M-1] := 0:
> # Step 3
> for I2 from 1 to M1 do
> V[I2-1] := evalf(F(I2*H)) :
> od:

> # Step 3
> # Steps 3 - 11 solve a tridiagonal linear system using
    Algorithm 6.7
> L[0] := 1 + VV:
> U[0] := -VV/(2*L[0]) :
> # Step 4
> for I2 from 2 to M2 do
> L[I2-1] := 1 + VV + VV* U[I2-2]/2 :
> U[I2-1] := -VV/(2*L[I2-1]) :
> od:
> # Step 5
> L[M1-1] := 1 + VV + 0.5* VV* U[M2-1] :
> # Step 6
> for J from 1 to N do
> # Step 7
> # Current t
> T := J* K:
> Z[0] := ((1-VV) * V[0] + VV* V[1]/2)/L[0] :
```

Comment

Step 20 of 23

Continuation of the above is as follows:

```
> # Step 8
> for I2 from 2 to M1 do
> Z[I2-1] := ((1-VV) * V[I2-1] + 0.5 * VV* (V[I2] + V[I2-2] + Z[I2
   -2]))/L[I2-
     1] :
> od:
> # Step 9
> V[M1-1] := Z[M1-1] :
> # Step 10
> for I1 from 1 to M2 do
> I2 := M2-I1 + 1 :
> V[I2-1] := Z[I2-1]-U[I2-1] * V[I2] :
> od:
> od:
```

Comment

Step 21 of 23

Continuation of the above is as follows:

```
> # Step 11
> 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, `MODIFIED CRANK-NICOLSON METHOD\n\n`) :
> fprintf(OUP, ` I    X(I)        W(X(I),%12.6e)\n`, FT) :
> for I2 from 1 to M1 do
> X := I2* H:
> fprintf(OUP, `%3d %11.8f %13.8f\n`, I2, X, V[I2-1]) :
> od:
> if OUP ≠ default then
> fclose(OUP) :
> print( `Output file `, NAME, ` created successfully`) :
> fi:
> fi:
```

Comment

Step 22 of 23

Hit the enter key near restart of the Maple Algorithm and proceed to input the data as per the instructions in the maple popup as shown below:

This is the Modified Crank-Nicolson Method.

Input the function F(X) in terms of x.

*For example: sin(3.141592654*x)*

F(x) =

sin(0.6666666667 π x)

The lefthand endpoint on the X-axis is 0.

Input the righthand endpoint on the X-axis.

Righthand endpoint =

1.5

Continuation of the above is as follows:

Input the maximum value of the time variable T.

Maximum time value =

0.225

Input the constant alpha.

alpha =

0.9805806757

Input integer m = number of intervals on X-axis

and N = number of time intervals - separated by a blank.

Note that m must be 3 or larger.

Number of intervals on x-axis =

10

Number of time intervals =

10

Choice of output method:

1. Output to screen

2. Output to text file

Please enter 1 or 2.

Input is

1

Comment

Step 23 of 23

MODIFIED CRANK-NICOLSON METHOD:

I X(I) W(X(I),2.250000e-01)

1 0.15000000 0.12047950

2 0.30000000 0.22916564

3 0.45000000 0.31541944

4 0.60000000 0.37079779

5 0.75000000 0.38987987

6 0.90000000 0.37079779

7 1.05000000 0.31541944

8 1.20000000 0.22916564

9 1.35000000 0.12047950

Result: Thus, the required approximate solution is the final Maple output.

Comment

Was this solution helpful? 0 1

Recommended solutions for you in Chapter 12.2

Chegg Study Textbook Solutions Expert Q&A

Become a Tutor
Chegg For Good
College Marketing
Corporate Development
Investor Relations
Jobs
Join Our Affiliate Program
Media Center
Site Map

Advertising Choices
Cookie Notice
General Policies
Intellectual Property Rights
International Privacy Policy
Terms of Use
Chegg Tutors Terms of Service
US Privacy Policy
Your CA Privacy Rights
Honor Code

Cheap Textbooks
Chegg Coupon
Chegg Play
Chegg Study Help
College Textbooks
eTextbooks
Chegg Math Solver
Mobile Apps

Online Tutoring
Sell Textbooks
Solutions Manual
Study 101
Test Prep
Textbook Rental
Used Textbooks
Digital Access Codes

Easybib
Internships.com
Studyblue

Customer Service
Give Us Feedback
Help with Chegg Tutors
Help with eTextbooks
Help to use EasyBib Plus
Manage Chegg Study Subscription
Return Your Books
Textbook Return Policy

PayPal VERIFIED
BBB
OVER 6 MILLION TREES PLANTED

© 2003-2019 Chegg Inc. All rights reserved.