Міністерство науки і освіти України
Національний університет "Львівська політехніка"
Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Звіт

про виконання лабораторної роботи №3
з курсу "Чисельні методи, частина 2"
на тему
"Лінійні багатокрокові методи чисельного розв'язування задачі Коші для системи звичайних диференціальних рівнянь"
Варіант 9

Виконав:
студент групи ПМ-42
Сватюк Д.Р.
Перевірила:
Візновмч В.О.

Львів 2017

Мета: навчитися чисельно знаходити розв'язок задачі Коші для системи ЗДР формулами диференціювання назад, використовуючи програму STIFF, написану на мові FORTRAN.

Постановка задачі

На відрізку $t \in (0,20]$ потрібно чисельно розв'язати задачу Коші

$$\begin{cases} u_1' = -u_1 + u_2^2 + u_3^2 + u_4^2, \\ u_2' = -10u_2 + 10(u_3^2 + u_4^2), \\ u_3' = -40u_3 + 40u_4^2, \\ u_4' = -100u_4 + 2, \quad t \in (0,20], \end{cases}$$

$$u_1(0) = u_2(0) = u_3(0) = u_4(0) = 1.$$

з точністю $\varepsilon = 10^{-2}, 10^{-4}, 10^{-6}$ та початковим кроком інтегрування $h_0 = 10^{-2}$.

Матриця Якобі правих частин цієї системи ЗДР має вигляд:

$$\begin{pmatrix} -1 & 2u_2 & 2u_3 & 2u_4 \\ 0 & -10 & 20u_3 & 20u_4 \\ 0 & 0 & -40 & 80u_4 \\ 0 & 0 & 0 & -100 \end{pmatrix}$$

Написати та відлагодити програму STIFF для розв'язування заданої задачі Коші.

Формули диференціювання назад

Розглянемо багатокрокові методи, які грунтуються на чисельному диференціюванні.

Припустимо, що відомі значення $y_{n-k+1}, y_{n-k+2}, ..., y_n$ розв'язку диференціального рівняння. Щоб вивести формулу для y_{n+1}, використаємо інтерполяційний многочлен $Q(t)$, який проходить через точки $\{(x_j, y_j) \mid j = \overline{n-k+1, n+1}\}$. Його можна виразити через різниці назад , а саме

$$Q(t) = Q(t_n + s\tau) = \nabla^0 y_{n+1} + \frac{s-1}{1!}\nabla y_{n+1} + \frac{(s-1)s}{2!}\nabla^2 y_{n+1} + \\ + ... + \frac{(s-1)s...(s+k-2)}{k!}\nabla^k y_{n+1}. \qquad (7)$$

Визначимо тепер невідоме значення y_{n+1} так, щоб многочлен $Q(t)$ задовольняв диференціальне рівняння хоча б в одному вузлі сітки, тобто

$$Q'(t_{n+1-r}) = f(t_{n+1-r}, y_{n+1-r}).$$

Враховуючи , що $s = (t - t_n)/\tau$, продиференціюємо (7) по змінній t

$$Q'(t) = \frac{1}{\tau}\sum_{j=1}^{k}\frac{d}{ds}\left(\frac{(s-1)s...(s+j-2)}{j!}\right)\nabla^j y_{n+1}.$$

Для $r = 1$ одержимо явні формули

$$\sum_{j=1}^{k}\delta_j \nabla^j y_{n+1} = \tau f_n ,$$

де

$$\delta_1 = 1, \quad \delta_j = \frac{d}{ds}\left[\frac{(s-1)s...(s+j-2)}{j!}\right]_{s=0} = -\frac{1}{j(j-1)}, \quad j \geq 2 .$$

Кращі властивості мають формули, які одержуються з (7) при $r = 0$. Це неявні формули

$$\sum_{j=1}^{k}\delta_j^* \nabla^j y_{n+1} = \tau f_{n+1} \qquad (8)$$

з коефіцієнтами

$$\delta_j^* = \frac{d}{ds}\left[\frac{(s-1)s...(s+j-2)}{j!}\right]_{s=1} ,$$

Тому (8) зводиться до формули

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = \tau f_{n+1} .$$

Такі багатокрокові методи називають формулами диференціювання назад. Вони вперше були виведені Кертісом і Хіршфельдером.

Формули диференціювання назад мають вигляд:

$$\sum_{j=0}^{k} \alpha_j y_{n-j+1} = \tau f_{n+1} .$$

Програмний код

```
 PROGRAM LAB4
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(10,13),YMAX(10),ERROR(10),PW(100),
     1          FSAVE(20),IWORK(10)
      COMMON/STCOM1/T,H,HMIN,HMAX,EPS,N,MF,KFLAG,JSTART,MAXORD
      COMMON/STCOM2/HUSED,NQUSED
      COMMON/STCOM3/ML,MU
      COMMON/STCOM4/NSTEP,NFUN,NJAC
      NYDIM=10
      EPS=1.D-2
      KB=0
401   CONTINUE
      N=3
      T=0.0D0
      TEND=1.D0
      Y(1,1)=1.D0
      Y(2,1)=0.D0
      Y(3,1)=0.D0
      H=3.3D-8      HMAX=TEND
      HMIN=1.D-15
      JSTART=0
      MF=21
      MAXORD=5
      WRITE(0,20) MF,EPS
20    FORMAT(//3X,'MF=',I2/,' EPS='D11.3)
      NSTEP=0
      NFUN=0
      NJAC=0
      DO 30 I=1,N
30    YMAX(I)=1.D0
40    CONTINUE
      CALL STIFF(Y,YMAX,ERROR,PW,FSAVE,IWORK,NYDIM)
      IF(KFLAG.EQ.0)GO TO 60
      WRITE(0,50) KFLAG
50    FORMAT(/'  KFLAG=',I2/)
      STOP
60    CONTINUE
      IF(DABS(TEND-T).LE.1.D-15) GO TO 90
      IF(TEND-T-H) 80,40,40
80    E=TEND-T
      S=E/H
      DO 85 I=1,N
      DO 85 J=1,JSTART
85    Y(I,1)=Y(I,1)+Y(I,J+1)*S**J
      T=T+E
      GO TO 60
90    CONTINUE
      WRITE(0,556) H,T,(Y(I,1),I=1,N)
556   FORMAT(1X,5D16.8)
```

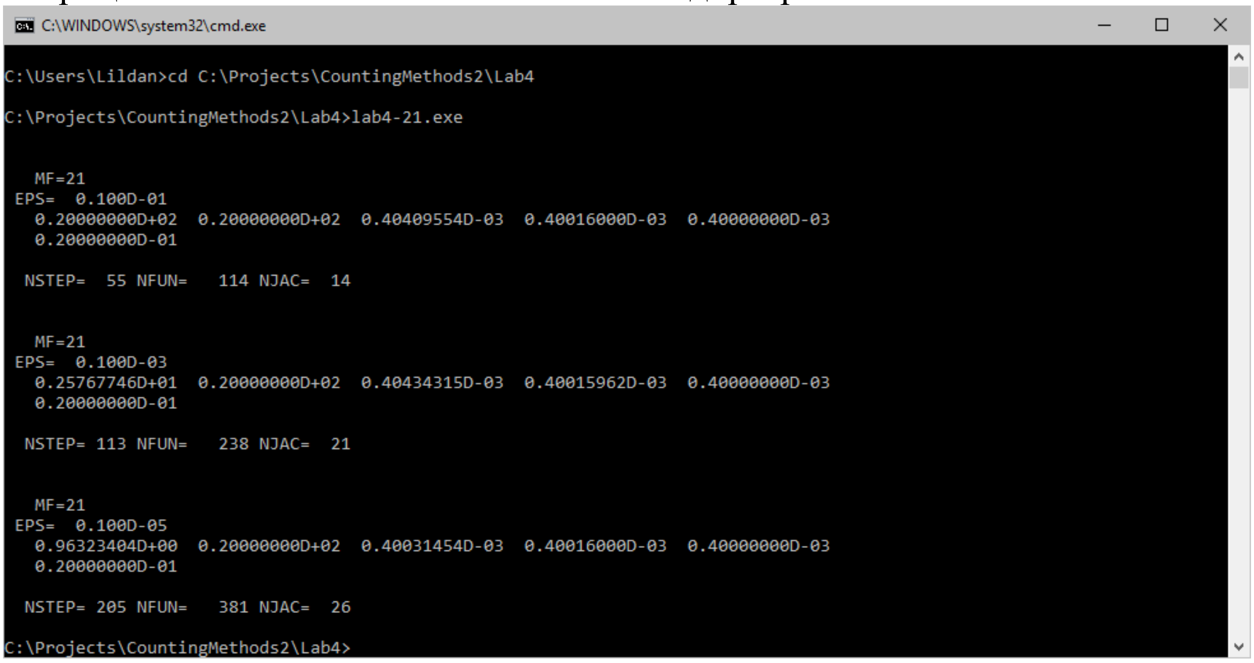
```
      WRITE(0,95) NSTEP,NFUN,NJAC
95    FORMAT(/'  NSTEP=',I4,' NFUN= ',I5,' NJAC=',I4)
      KB=KB+1
      IF(KB.GE.3) GO TO 402
      EPS=EPS*1.D-2
      GO TO 401
402   CONTINUE
      STOP
      END

SUBROUTINE DIFFUN (N,T,Y,YDOT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1),YDOT(1)
      YDOT(1)=-Y(1)+100000000.D0*Y(3)*(1.D0-Y(1))
      YDOT(2)=-10.D0*Y(2)+30000000.D0*Y(3)*(1.D0-Y(2))
      YDOT(3)=-YDOT(1)-YDOT(2)      RETURN
      END

SUBROUTINE PEDERV(N,T,Y,PW,NYDIM)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1),PW(1)
      PW(1)=-1.0D0-10.D0*Y(3)
      PW(2)=0.D0
      PW(3)=1.D0+100000000.D0*Y(3)
      PW(NYDIM+1)=0.D0
      PW(NYDIM+2)=-10.D0-30000000.D0*Y(3)
      PW(NYDIM+3)=10.D0+30000000.D0*Y(3)
      N2=NYDIM*2
      PW(N2+1)=100000000.D0*(1.D0-Y(1))
      PW(N2+2)=30000000.D0*(1.D0-Y(2))
      PW(N2+3)=-100000000.D0*(1.D0-Y(1))-30000000.D0*(1.D0-Y(2))      RETURN
      END
```

Результати виконання

1) Використовується модифікований ітераційний метод Ньютона, матриця Якобі обчислюється аналітично підпрограмою PEDERV.


```
C:\WINDOWS\system32\cmd.exe

C:\Users\Lildan>cd C:\Projects\CountingMethods2\Lab4

C:\Projects\CountingMethods2\Lab4>lab4-21.exe


   MF=21
 EPS=  0.100D-01
   0.20000000D+02  0.20000000D+02  0.40409554D-03  0.40016000D-03  0.40000000D-03
   0.20000000D-01

  NSTEP=  55 NFUN=   114 NJAC=  14


   MF=21
 EPS=  0.100D-03
   0.25767746D+01  0.20000000D+02  0.40434315D-03  0.40015962D-03  0.40000000D-03
   0.20000000D-01

  NSTEP= 113 NFUN=   238 NJAC=  21


   MF=21
 EPS=  0.100D-05
   0.96323404D+00  0.20000000D+02  0.40031454D-03  0.40016000D-03  0.40000000D-03
   0.20000000D-01

  NSTEP= 205 NFUN=   381 NJAC=  26

C:\Projects\CountingMethods2\Lab4>
```

Рис: 1

2) Використовується модифікований ітераційний метод Ньютона, матриця Якобі обчислюється чисельним диференціюванням.

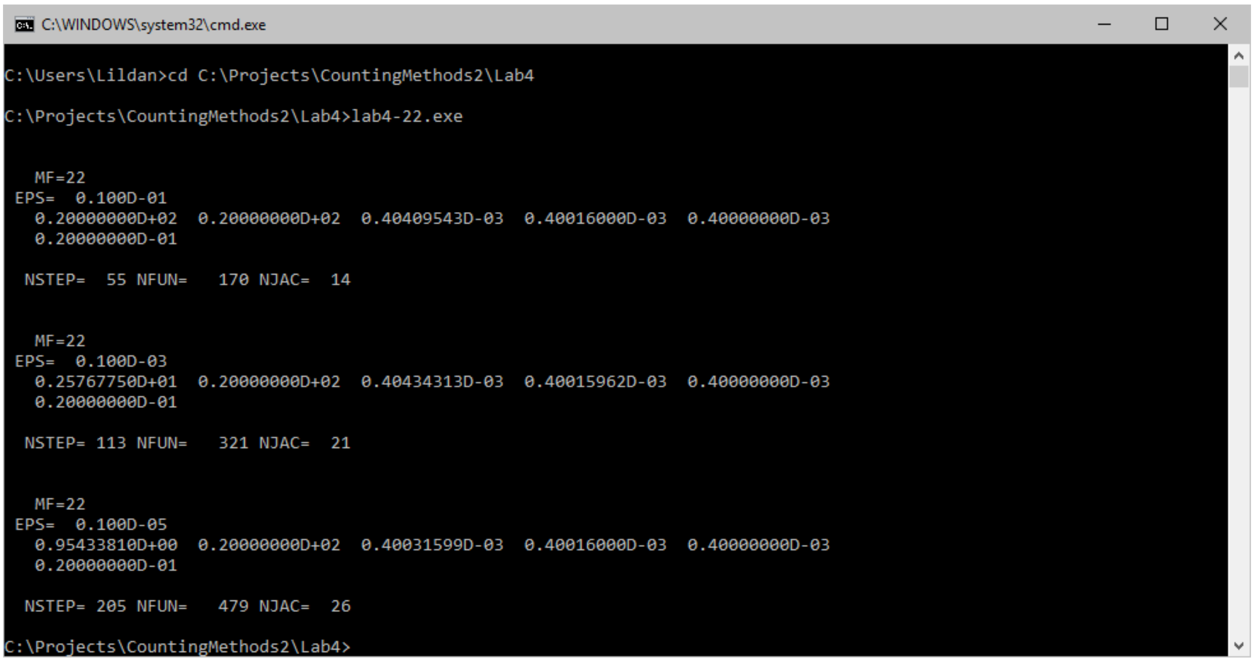


Рис: 2

Висновок

Під час виконання лабораторної роботи, я навчився розв’язувати задачу Коші для системи ЗДР формулами диференціювання назад використовуючи программу STIFF.

Також знайшов аналітично матрицю Якобі для моєї системи ЗДР.