**AULA PRÁTICA DE PROBLEMA DE VALOR DE CONTORNO (PVC)**

Exemplo 7.10) O problema de camada limite fluida e térmica em uma placa plana de comprimento L, em escoamento forçado laminar e temperatura uniforme na placa, , pode ser resolvido a partir de uma transformação de variáveis que se baseia na introdução de uma variável independente conhecida como similar , onde  é a espessura da camada limite fluida (desde a velocidade  na superfície da placa até atingir suficientemente longe da placa) em uma posição x qualquer sobre a placa (direção do escoamento forçado) de um referencial cartesiano com origem situada no bordo de ataque da placa, i.e., no local em que o escoamento livre encontra a placa, e y uma posição qualquer no interior da camada limite na direção perpendicular ao escoamento forçado, conforme mostra a Fig. 7.13. O domínio de integração é definido por , i.e., de um ponto na superfície da placa, , até um ponto suficientemente distante da placa, , em que o escoamento livre já não sofre mais a influência dos efeitos de atrito provocados pela presença da placa plana. Em suma, um problema originalmente bidimensional formulado por 3 equações diferenciais parciais (princípios de conservação da quantidade de movimento e da energia) com duas variáveis independentes, x e y, passa a ser unidimensional sendo formulado por um sistema de 2 equações diferenciais ordinárias em relação a uma variável independente, . Para maiores detalhes conceituais sobre o problema, sugere-se consultar a literatura técnica no assunto Mecânica dos Fluidos e Transferência de Calor no regime de convecção forçada (e.g., Bejan, 1995). O problema é formulado matematicamente com as equações transformadas pelos seguintes PVCs:

1. Camada limite fluida

 (7.94)

1. Camada limite térmica

 (7.95)



0

L

placa plana

u

T









x

y

Figura 7.13 – Diagrama esquemático de formação de camada limite em placa plana sob escoamento forçado em regime laminar.

Tomando  para o fluido, que é a relação entre as propriedades termofísicas viscosidade cinemática, , e difusividade térmica de um fluido, , observa-se que  que é a espessura da camada limite térmica em uma posição x qualquer sobre a placa (desde a temperatura  na superfície da placa até atingir  suficientemente longe da placa). Pede-se: i) apresentar a solução  (velocidades adimensionais, em que ) e  (temperaturas adimensionais, em que ) para  numericamente em uma tabela e na forma gráfica, tomando , usando o método da secante para obter as condições iniciais desconhecidas  e , e ii) os valores encontrados para  e , que determinam o coeficiente de atrito local, , em que , e o número de Nusselt local, , para placa plana, em que , h o coeficiente de transferência de calor por convecção e k a condutividade térmica do fluido, que permitem calcular a força de arrasto na placa e a taxa de transferência de calor entre a placa e a corrente livre (Bejan, 1995).

**Solução**

1. O primeiro aspecto a ser notado é que o problema da Eq. (7.94) é desacoplado do problema da Eq. (7.95), podendo ser resolvido sem o conhecimento da solução térmica, i.e., . Assim, resolve-se primeiramente o PVC da Eq. (7.94) e, a seguir, resolve-se o PVC da Eq. (7.95), uma vez que depende do conhecimento da solução do primeiro problema, i.e., . O PVC equivalente à Eq. (7.94) em termos de EDOs de 1ª ordem é dado por:

 (7.96)

1. De acordo com a metodologia apresentada na seção, resolver o PVC da Eq. (7.96) consiste de resolver uma equação algébrica não linear de 1 incógnita, i.e., o valor inicial desconhecido, , conforme é indicado pela Eq. (7.92). Para tanto, utiliza-se um dos métodos apresentados no capítulo 2 deste livro. Neste exemplo, opta-se pelo método da secante para resolver a seguinte equação:

 (7.97)

em que  a cada iteração “i” do método da secante é obtido pela solução do seguinte PVI para :

  (7.98)

onde  é o número máximo de iterações estabelecido para o método da secante apresentar uma solução convergida de acordo com uma tolerância pré-estabelecida.

1. Neste ponto, resolve-se o PVC da Eq. (7.95), uma vez que agora é conhecida a solução do primeiro problema, i.e., . O PVC equivalente à Eq. (7.95) em termos de EDOs de 1ª ordem é dado por:

 (7.99)

e prosseguindo de forma análoga ao item anterior, usa-se o método da secante para resolver a seguinte equação para o valor inicial desconhecido :

 (7.100)

em que  a cada iteração “i” do método da secante é obtido pela solução do PVI representado pela Eq. (7.99) usando  para .

1. A solução para as variáveis  é obtida a partir de um programa computacional implementado em linguagem Fortran (DVD em anexo ao livro). Aqui se usa o método de Runge-Kutta de 4ª ordem de passo fixo, que foi implementado em um programa computacional (DVD em anexo ao livro). O resultado para  é mostrado numericamente na Tabela 7.7 e na forma gráfica na Fig. 7.14, tomando . No método da secante foram utilizados os valores iniciais 0,1 e 1,2 para .

Tabela 7.7 – Solução do PVC dado pela Eq. (7.96).

---------------------------------------------------------------------------------------------------

 Passo Nr  f1 f2  f3

 --------------------------------------------------------------------------------------------------

 0 0.000000E+00 0.000000E+00 0.000000E+00 3.320580E-01

 100 9.999993E-01 1.655720E-01 3.297805E-01 3.230076E-01

 200 1.999998 6.500254E-01 6.297665E-01 2.667518E-01

 300 2.999998 1.396810 8.460455E-01 1.613604E-01

 400 3.999997 2.305750 9.555193E-01 6.423409E-02

 500 5.000020 3.283278 9.915429E-01 1.590675E-02

 600 6.000042 4.279625 9.989738E-01 2.402028E-03

 700 7.000065 5.279244 9.999226E-01 2.201674E-04

 800 8.000088 6.279212 9.999973E-01 1.224082E-05

 900 9.000111 7.279225 1.000000 4.127843E-07

 1000 10.000130 8.279247 1.000000 8.442745E-09

---------------------------------------------------------------------------------------------------



Figura 7.14 – Solução gráfica para a variável f2 (velocidade adimensional na direção x) para o PVC dado pela Eq. (7.96).

1. A solução para as variáveis  é obtida com o mesmo programa computacional implementado em linguagem Fortran (DVD em anexo ao livro). Novamente, é usado o método de Runge-Kutta de 4ª ordem de passo fixo, que foi implementado em um programa computacional (DVD em anexo ao livro). O resultado para  é mostrado numericamente na Tabela 7.8 e na forma gráfica na Fig. 7.15, tomando . No método da secante foram utilizados os valores iniciais 0,1 e 1,2 para .

Tabela 7.8 – Solução do PVC dado pela Eq. (7.99).

-----------------------------------------------------------------------------

 Passo Nr   

 ----------------------------------------------------------------------------

 0 0.000000E+00 0.000000E+00 3.313405E-01

 100 9.999993E-01 3.291237E-01 3.224813E-01

 200 1.999998 6.288295E-01 2.667465E-01

 300 2.999998 8.453377E-01 1.617588E-01

 400 3.999997 9.552202E-01 6.458800E-02

 500 5.000020 9.914726E-01 1.602420E-02

 600 6.000042 9.989626E-01 2.423784E-03

 700 7.000065 9.999206E-01 2.225320E-04

 800 8.000088 9.999961E-01 1.239292E-05

 900 9.000111 9.999993E-01 4.186104E-07

 1000 10.000130 9.999993E-01 8.576186E-09

------------------------------------------------------------------------------



Figura 7.15 – Solução gráfica para a variável  (temperatura adimensional) para o PVC dado pela Eq. (7.99).

**PROGRAMA COMPUTACIONAL:**

**main.f90**

! program blasius

use msflib ! systemqq

logical calling

 parameter (nmax=20000)

! dimension tp(nelmax)

! external fcn,rkqc

 common /param/ eta0,deta,etaend,f10,f20

 common /noeq/ n

 common /noeqt/ nt

 common /div/ ndiv

 common /prandtl/ pr

 common /etavec/ eta(nmax)

 common /f1vec/ f1(nmax)

 common /param1/ t10

 external ff,tf

 ! common /const/ ht2

 open(1,file='inpclass.txt')

 open(2,file='f1.txt')

 open(3,file='f2.txt')

 open(4,file='f3.txt')

 open(5,file='t1.txt')

 open(6,file='t2.txt')

 open(7,file='f.txt')

 open(8,file='t.txt')

! Solution of a Boundary Value Problem - BVP

!

! data input

!

 read(1,\*)n

 write(\*,\*)'n=',n

 read(1,\*)eta0

 write(\*,\*)'eta0=',eta0

 read(1,\*)etaend

 write(\*,\*)'etaend=',etaend

 read(1,\*)deta

 write(\*,\*)'deta=',deta

 read(1,\*)f10

 write(\*,\*)'f10=',f10

 read(1,\*)f20

 write(\*,\*)'f20=',f20

 read(1,\*)f30a

 write(\*,\*)'f30a=',f30a

 read(1,\*)f30b

 write(\*,\*)'f30b=',f30b

 read(1,\*)maxit

 write(\*,\*)'maxit=',maxit

 read(1,\*)tol

 write(\*,\*)'tol=',tol

 read(1,\*)nt

 write(\*,\*)'nt=',nt

 read(1,\*)pr

 write(\*,\*)'pr=',pr

 read(1,\*)t10

 write(\*,\*)'t10=',t10

 read(1,\*)t20a

 write(\*,\*)'t20a=',t20a

 read(1,\*)t20b

 write(\*,\*)'t20b=',t20b

 ! Blasius solution

 call secante(f30a,f30b,maxit,tol,c,ff)

 ! Energy equation

 ndiv=etaend/deta

 write(\*,\*)'ndiv=',ndiv

 call secante(t20a,t20b,maxit,tol,d,tf)

 write(\*,\*) 'f30=',c

 write(\*,\*) 't20=',d

 close(2)

 close(3)

 close(4)

 close(5)

 close(6)

 close(7)

 close(8)

calling = systemqq('notepad f1.txt') ! list of data

calling = systemqq('notepad f2.txt') ! list of data

calling = systemqq('notepad f3.txt') ! list of data

calling = systemqq('notepad t1.txt') ! list of data

calling = systemqq('notepad t2.txt') ! list of data

calling = systemqq('notepad f.txt') ! list of f-table

calling = systemqq('notepad t.txt') ! list of t-table

calling = systemqq('wgnuplot data.gnu') ! graph - velocities

calling = systemqq('wgnuplot datat.gnu') ! graph - temperatures

 stop

 end

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 subroutine fcn(n,t,fi,f,nelmax)

 dimension fi(nelmax),f(nelmax)

!

 f(1)=fi(2)

 f(2)=fi(3)

 f(3)=-fi(1)\*fi(3)/2

!

 return

 end

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 subroutine tfcn(n,t,fi,f,nelmax)

 dimension fi(nelmax),f(nelmax)

 common /etavec/ eta1(1)

 common /f1vec/ f1(1)

 common /div/ ndiv

 common /param/ eta0,deta,etaend,f10,f20

 common /prandtl/ pr

!

! identify appropriate position of eta in f-table

!

! write(\*,\*)'ndiv=',ndiv

 ipos=t/deta+1

! write (\*,\*) ipos,t,eta1(ipos),f1(ipos)

!

 f(1)=fi(2)

 f(2)=-pr\*f1(ipos)\*fi(2)/2

!

 return

 end

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!234567890123456789012345678901234567890123456789012345678901234567890

 subroutine fore(n,fcn,time,fi,tend,nelmax)

!

! implicit real \*8 (a-h,o-z)

 parameter (nd1=100)

 dimension fi(nelmax),f(nd1)

 common /const/ ht2

 external fcn

 k=0

 50 k=k+1

 time=min(time+ht2,tend)

 call fcn(n,time,fi,f,nelmax)

 do 100 i=1,n

 fi(i)=fi(i)+ht2\*f(i)

 100 continue

 if (time.lt.tend) goto 50

 return

 end

!------------------------------------------------------------------

 function ff(x)

 parameter (nmax=100)

 dimension tp(nmax)

 external fcn,rkqc

 common /noeq/ n

 common /flag/ iflag

 common /etavec/ eta1(1)

 common /f1vec/ f1(1)

 common /param/ eta0,deta,etaend,f10,f20

!

! initial values

!

 eta=eta0

 tp(1)=f10

 tp(2)=f20

 tp(3)=x

!

 k=0

 if (iflag.eq.1) then

 eta1(k+1)=eta

 f1(k+1)=tp(1)

 write(2,\*)eta,tp(1)

 write(3,\*)eta,tp(2)

 write(4,\*)eta,tp(3)

 write(7,\*)'--------------------------------------------'

 write(7,\*)'Nr do passo eta f1 f2 f3'

 write(7,\*)'--------------------------------------------'

 write(7,\*)k,eta,(tp(l),l=1,n)

 endif

 ht2=deta/1

 nd=nmax

!

! beginning of time loop

!

 50 k=k+1

 tendi=eta+deta

! write(\*,\*)'-------------eta=',tendi

!

! call odeint(tp,n,time,tendi,1.e-6,ht2,1.e-20,id1,id2,nelmax,fcn,rkqc)

 call rk4ord(tp,n,eta,deta,fcn,nd)

! call fore(n,fcn,time,tp,tendi,nmax)

 if (iflag.eq.1) then

 if (k.eq.100.or.k.eq.200.or.k.eq.300.or.k.eq.400.or.k.eq.500.or.k.eq.600.or.k.eq.700.or.k.eq.800.or.k.eq.900.or.k.eq.1000) write(7,\*)k,tendi,(tp(l),l=1,n)

 eta1(k+1)=tendi

 f1(k+1)=tp(1)

 write(2,\*)tendi,tp(1)

 write(3,\*)tendi,tp(2)

 write(4,\*)tendi,tp(3)

 endif

!

 if (tendi.lt.etaend) then

 eta=tendi

 goto 50

 endif

 aux=tp(2)-1.

 ff=aux

 return

 end

!------------------------------------------------------

**ode.f90**

!234567890123456789012345678901234567890123456789012345678901234567890

 subroutine odeint(ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad,nd,derivs,rkqc)

 parameter (maxstp=10000,nmax=100,two=2.0,zero=0.0,tiny=1.d-30)

 parameter (nd1=100)

 common /path/ kmax,kount,dxsav

 dimension ystart(nd),yscal(nd1),y(nd1),dydx(nd1)

 external derivs,rkqc

 x=x1

 h=sign(h1,x2-x1)

 nok=0

 nbad=0

 kount=0

 do 11 i=1,nvar

 y(i)=ystart(i)

 11 continue

 if (kmax.gt.0) xsav=x-dxsav\*two

 do 16 nstp=1,maxstp

 call derivs(nvar,x,y,dydx,nd)

 do 12 i=1,nvar

 yscal(i)=abs(y(i))+abs(h\*dydx(i))+tiny

 12 continue

 if ((x+h-x2)\*(x+h-x1).gt.zero) h=x2-x

 call rkqc(y,dydx,nvar,x,h,eps,yscal,hdid,hnext,derivs,nd)

 if (hdid.eq.h) then

 nok=nok+1

 else

 nbad=nbad+1

 endif

 if ((x-x2)\*(x2-x1).ge.zero) then

 do 14 i=1,nvar

 ystart(i)=y(i)

 14 continue

 return

 endif

 if (abs(hnext).lt.hmin) then

 write(\*,\*) 'stepsize small',hmin

 stop

 endif

 h=hnext

 16 continue

 write(\*,\*) 'too many steps',nstp

 stop

 end

!234567890123456789012345678901234567890123456789012345678901234567890

**rk.f90**

 subroutine rkqc(y,dydx,n,x,htry,eps,yscal,hdid,hnext,derivs,nd)

!

! fifth-order RK

!

! implicit real \*8 (a-h,o-z)

 parameter (nmax=100, pgrow=-.20,pshrnk=-.25,fcor=1.d0/15.,one=1., safety=.9, errcon=6.e-4,nd2=100)

 external derivs

 dimension y(nd),dydx(nd),yscal(nd),ytemp(nd2),ysav(nd2),dysav(nd2)

 xsav=x

 do 11 i=1,n

 ysav(i)=y(i)

 dysav(i)=dydx(i)

 11 continue

 h=htry

 1 hh=0.5\*h

 call rk4(ysav,dysav,n,xsav,hh,ytemp,derivs,nd)

 x=xsav+hh

 call derivs(n,x,ytemp,dydx,nd)

 call rk4(ytemp,dydx,n,x,hh,y,derivs,nd)

 x=xsav+h

 if (x.eq.xsav) then

 write(\*,\*) 'stepsize not significant in rkqc',x

 stop

 endif

 call rk4(ysav,dysav,n,xsav,h,ytemp,derivs,nd)

 errmax=0.

 do 12 i=1,n

 ytemp(i)=y(i)-ytemp(i)

 dummy=abs(ytemp(i)/yscal(i))

 errmax=max(errmax,dummy)

 12 continue

 errmax=errmax/eps

 if(errmax.gt.one) then

 h=safety\*h\*(errmax\*\*pshrnk)

 goto 1

 else

 hdid=h

 if (errmax.gt.errcon) then

 hnext=safety\*h\*(errmax\*\*pgrow)

 else

 hnext=4.d0\*h

 endif

 endif

 do 13 i=1,n

 y(i)=y(i)+ytemp(i)\*fcor

 13 continue

 return

 end

!---------------------------------------------------------------------

 subroutine rk4(y,dydx,n,x,h,yout,derivs,nd)

!

! rk4

!

 parameter (nmax=100,nd3=100)

 dimension y(nd),dydx(nd),yout(nd),yt(nd3),dyt(nd3),dym(nd3)

 external derivs

 hh=h\*.5

 h6=h/6

 xh=x+hh

 do 11 i=1,n

 yt(i)=y(i)+hh\*dydx(i)

 11 continue

 call derivs(n,xh,yt,dyt,nd)

 do 12 i=1,n

 yt(i)=y(i)+hh\*dyt(i)

 12 continue

 call derivs(n,xh,yt,dym,nd)

 do 13 i=1,n

 yt(i)=y(i)+h\*dym(i)

 dym(i)=dyt(i)+dym(i)

 13 continue

 call derivs(n,x+h,yt,dyt,nd)

 do 14 i=1,n

 yout(i)=y(i)+h6\*(dydx(i)+dyt(i)+2\*dym(i))

 14 continue

 return

 end

!234567890123456789012345678901234567890123456789012345678901234567890

**rk4ord.f90**

 subroutine rk4ord(y,n,x,h,derivs,nd)

!

! rk4

!

 parameter (nmax=100,nd3=100)

 dimension y(nd),dydx(nd3),yt(nd3),dyt(nd3),dym(nd3)

 external derivs

 hh=h\*.5

 h6=h/6

 xh=x+hh

 call derivs(n,x,y,dydx,nd)

 do 11 i=1,n

 yt(i)=y(i)+hh\*dydx(i)

 11 continue

 call derivs(n,xh,yt,dyt,nd)

 do 12 i=1,n

 yt(i)=y(i)+hh\*dyt(i)

 12 continue

 call derivs(n,xh,yt,dym,nd)

 do 13 i=1,n

 yt(i)=y(i)+h\*dym(i)

 dym(i)=dyt(i)+dym(i)

 13 continue

 call derivs(n,x+h,yt,dyt,nd)

 do 14 i=1,n

 y(i)=y(i)+h6\*(dydx(i)+dyt(i)+2\*dym(i))

 14 continue

 return

 end

!234567890123456789012345678901234567890123456789012345678901234567890

**secante.f90**

 subroutine secante(xa0,xa1,maxit,tol,xatual,ff)

 common /convergence/ auxi3,rtol

 common /flag/ iflag

!

 iflag=0

 rtol=tol

 xantes = xa0

 auxi1 = ff(xantes)

 xpos = xa1

 auxi2 = ff(xpos)

 if (abs(auxi1).le.tol) then

 xatual = xantes

 auxi3=auxi1

 goto 333

 endif

 if (abs(auxi2).le.tol) then

 xatual = xpos

 auxi3 = auxi2

 goto 333

 endif

 do i=1,maxit

 xatual=xpos-auxi2\*(xpos-xantes)/(auxi2-auxi1)

 auxi3 = ff(xatual)

 ! write(10,\*)i,auxi3

 ! write(12,\*)i,xatual,auxi3

 if (abs(auxi3).le.tol) goto 333

 xantes = xpos

 auxi1=auxi2

 xpos = xatual

 auxi2 = auxi3

 enddo

 if (i.ge.maxit) then

 write (\*,\*)'Nao convergiu, F=',abs(auxi3)

 return

 endif

333 CONTINUE

 iflag=1

 aux5=ff(xatual)

 write (\*,\*)'A solucao secante eh x=',xatual,' f=',abs(auxi3)

 RETURN

 END

!--------------------------------------------------------------

**tfunc.f90**

!------------------------------------------------------------------

 function tf(x)

 parameter (nmax=100)

 dimension tp(nmax)

 external tfcn,rkqc

 common /noeqt/ n

 common /flag/ iflag

 common /const/ ht2

 common /param/ eta0,deta,etaend,f10,f20

 common /param1/ t10

!

! initial values

!

 eta=eta0

 tp(1)=t10

 tp(2)=x

!

 k=0

 if (iflag.eq.1) then

 write(5,\*)eta,tp(1)

 write(6,\*)eta,tp(2)

 write(8,\*)'--------------------------------------------'

 write(8,\*)'Nr do passo eta t1 t2 '

 write(8,\*)'--------------------------------------------'

 write(8,\*)k,eta,(tp(l),l=1,n)

 endif

!

 ht2=deta/1

 nd=nmax

!

! beginning of time loop

!

 50 k=k+1

 tendi=eta+deta

! write(\*,\*)'-------------eta=',tendi

!

! call odeint(tp,n,eta,tendi,1.e-6,ht2,1.e-20,id1,id2,nmax,tfcn,rkqc)

 call rk4ord(tp,n,eta,deta,tfcn,nd)

! call fore(n,tfcn,eta,tp,tendi,nmax)

 if (iflag.eq.1) then

 if (k.eq.100.or.k.eq.200.or.k.eq.300.or.k.eq.400.or.k.eq.500.or.k.eq.600.or.k.eq.700.or.k.eq.800.or.k.eq.900.or.k.eq.1000) write(8,\*)k,tendi,(tp(l),l=1,n)

 write(5,\*)tendi,tp(1)

 write(6,\*)tendi,tp(2)

 endif

!

 if (tendi.lt.etaend) then

 eta=tendi

 goto 50

 endif

 aux=tp(1)-1.

 tf=aux

 return

 end

!------------------------------------------------------

**data.gnu**

set data style linespoints

set grid

set xlabel 'Espessura adimensional da camada limite (eta)'

set ylabel 'Solução numérica f2(eta)'

set title 'Velocidade adimensional (f)'

plot 'f2.txt'

pause -1

**datat.gnu**

set data style linespoints

set grid

set xlabel 'Espessura adimensional da camada limite (eta)'

set ylabel 'Solução numérica teta(eta)'

set title 'Temperatura adimensional (teta)'

plot 't1.txt'

pause -1

**inpclass.txt**

3 ! n=number of equations (momentum)

0. ! eta0 = initial eta

10. ! etaend = eta at infinity

0.01 ! deta = eta-stepsize

0. ! f10 = f initial value

0. ! f20 = f-prime initial value

0.1 ! f30a = first f-2prime guessed initial value

1.2 ! f30b = second f-2prime guessed initial value

100 ! maxit - max number of iterations

1.e-6 ! tol = tolerance for secant method

2 ! nt=number of equations (energy)

1. ! pr = Prandtl number

0. ! t10 = teta initial value

0.1 ! t20a = first teta-1prime guessed initial value

1.2 ! t20b = second teta-1prime guessed initial value