**main.f90**

! program amortecimento

! use msflib ! biblioteca que contem o comando systemqq

! logical chamada

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

 parameter (nelmax=100)

 dimension tp(nelmax)

 external fcn,rkqc

 common /const/ ht2

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

 open(2,file='x1.txt')

 open(3,file='x-all.txt')

 open(4,file='x2.txt')

!

! integrador de EDO's com passo adaptativo - RK 4a/5a ordem

! integrador de EDO's com passo fixo - RK 4a ordem e For. Euler

!

 read(1,\*)iflag

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

 read(1,\*)n,tau0

 write(\*,\*)'n=',n,'tau0=',tau0

 read(1,\*) tend,dtau

 write(\*,\*)'tend=',tend,'dtau=',dtau

!

! initial values

!

 read(1,\*)(tp(l),l=1,n)

 write(\*,\*)(tp(l),l=1,n)

! initial time

 time=tau0

 k=0

 write(3,\*) ' Table of results'

 write(3,\*)'-----------------------------'

 write(3,\*)' Passo Nr time T(i) '

 write(3,\*)'-----------------------------'

 write(3,\*)k,time,(tp(l),l=1,n)

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

 write(4,\*)time,tp(2)

 ht2=dtau/100

!

! beginning of time loop

!

 50 k=k+1

 tendi=time+dtau

 write(\*,\*)'-------------time=',tendi

 if(iflag.eq.0) then ! RK-adaptativo

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

 endif

 if(iflag.eq.1) then ! RK-passo fixo

 kk=0

 500 kk=kk+1

 time=min(time+ht2,tendi)

 call rk4ord(tp,n,time,ht2,fcn,nelmax)

 if (time.lt.tendi) goto 500

 endif

 if(iflag.eq.2) then ! Forward Euler

 call fore(n,fcn,time,tp,tendi,nelmax)

 endif

 write(3,\*)k,tendi,(tp(l),l=1,n)

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

 write(4,\*)tendi,tp(2)

 if (tendi.lt.tend) then

 time=tendi

 goto 50

 endif

 close(2)

 close(3)

 close(4)

call system('notepad x-all.txt') ! listagem de todas as variaveis

! chamada = systemqq('notepad x2.txt') ! listagem 2a variavel

call system('wgnuplot dados.gnu') ! gráfico geral

 stop

 end

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

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

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

 dimension fi(nelmax),f(nelmax)

 ! f(1)=(t\*fi(1)-fi(1)\*\*2)/t/t+fi(2)

 ! f(2)=fi(2)-fi(1)

 ! Exemplo 7.8 do livro - sistema sub-, critico e superamortecido

 zeta=0.1

 w\_n=0.1

 f(1)=fi(2)

 f(2)=w\_n\*\*2\*(-fi(1)-2\*zeta\*fi(2)/w\_n)

 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

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

**dados.gnu**

set data style linespoints

set grid

set xlabel 't(s)'

set ylabel 'x(t) e xlinha(t)'

set title 'Resposta de sistema de segunda ordem'

plot 'x1.txt','x2.txt'

pause -1

**inpaula.txt**

0 ! iflag = 0 (RK-adaptativo), 1 (RK-fixo), 2 (Forward Euler)

2 0. ! n = numero de equacoes, tau0 = tempo inicial

500. 5. ! tend = tempo final, dtau = passo de tempo externo

100. 0. ! tp0(i) = valores iniciais (i=1,n)

**Ode.txt**

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

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

 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

**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

!

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

 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

**rk4ord.f90**

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

!

! rk4

!

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

 parameter (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