6
Referências Bibliográficas

ALESSANDRONI, S., ANDREAUS, U., DELL´ISOLA, F., PORFIRI, M. **A passive electric controller for multimodal vibrations of thin plates.** Computers and Structures 83 (2005) 1236-1250.

BASSILY, S.F; DICKINSON, S.M. **Buckling and Lateral Vibration of Rectangular Plates Subject to Inplane Loads – A Ritz Aproach.** Journal of Sound and Vibrations 24 (2), 219-239, 1972.

BEDAIR, O.K.; SHERBOURNE, A. **On the Stability of Plates Under Combined Compression and In-Plane Bending.** Computer & Structures. Vol. 53 (6) 1453-1464, 1994.

BOYCE, William E.; DIPRIMA, Richard C. **Elementary differential equations and boundary value problems** /. 6th ed. - New York : J. Wiley & Sons, c1997. 749p.

BRUSH, Don Orr; ALMROTH, Bo O. **Buckling of bars, plates, and shells.** New York: McGraw-Hill, c1975. 379p.

CURADELLI, R.O., AMBROSINI, R.D., DANESI, R.F. **Vibration control by attaching masses to a plate excited by rotating machinery.** Journal of Sound and Vibration 273 (2004) 1087-1100.

DICKINSON, S.M. **Lateral Vibration of Rectangular Plates Subject to in-plane Forces.** Journal of Sound and Vibration. Vol. 16 (4), 465-472, 1971.

_____. **The Buckling and Frequency of Flexural Vibration of Rectangular isotropic and orthotropic plates using Rayleighs's Method.** Journal of Sound and Vibration. Vol 61 (1) 1-8, 1978.

HOUSNER, et al. **Structural Control: Past, Present and Future.** Journal of Engineering Mechanics – ASCE. Vol. 123, (9), 897-971, 1997.

HUI, D., LEISSA, A.W. **Effects of Geometric Imperfections on Vibrations of Biaxially Compressed Rectangular Flat Plates.** Journal of Applied Mechanics – ASME, v. 50, n. 4A, p. 750-756, 1983.

ILANKO, S. **Vibration and Post-Buckling of In-Plane Loaded Rectangular Plates Using a Multiterm Galerkin's Method.** Journal of Applied Mechanics – ASME. Vol. 69. 589-592, 2002.

KALDAS, M.M.; DICKINSON, S.M. **Vibration and Buckling Calculations for Rectangular Plates Subject to Complicated in-plane Stress Distributions by using numerical integration in a Rayleigh-Ritz Analysis.** Journal of Sound and Vibration. Vol 75(2) 151-162, 1981.

KEIR, J., KESSISSOGLOU, N.J., NORWOOD, C.J. **Active control of connected plates using single and multiple actuators and error sensors.** Journal of Sound and Vibration 281 (2005) 73-97.

KREYSZIG, Erwin. **Advanced engineering mathematics**. 8th ed. New York : Wiley c1999 [1273] p.

KREYSZIG, Erwin.; NORMINTON, E. J.; KREYSZIG, Erwin. **Maple computer manual** /. New York : Wiley c1994. 417p., [45]f.

LEISSA, A.W., **Vibration of Plates**. NASA-SP-160, Washinton, D.C. Office of Technology Utilization – NASA, 1969, 159 p. Technical Report.

LIU, W., HOU, Z., DEMETRIOU, M.A. **A computational scheme for the optimal sensor/actuator placement of flexible structures using spatial H_2 measures.** Mechanical Systems and Signal Processing 20 (2006) 881-895.

LURIE, Harold,. **Lateral Vibrations as Related to Structural Stability**. Journal of Applied Mechanics – ASME, v. 19, n. 2, p. 195-204, 1952.

MAT DARUS, I.Z., TOKHI, M.O.. **Soft computing-based active vibration control of a flexible structure**. Engineering Applications of Artificial Intelligence 18 (2005), 93-114.

MUNAKATA, K. **On the Vibration and Elastic Instability of a Rectangular Plate Clamped at its four edges.** Journal of Mathematics and Phisics. Vol. 31 (1) 69-75, 1952.

NOVOZHILOV, Valentin Valentinovich. **Foundations of the nonlinear theory of elasticity.** Rochester, N. Y.: Graylock, 1953. 233 p.

RAO, Singiresu S. **Mechanical Vibrations** – Menlo Park: Addison-Wesley, c1986. 600p.

SOONG, T.T., SPENCER Jr, B.F. **Supplemental energy dissipation: state-of-the-art and state-of-the practice**. Engineering Structures 24 (2002), 243–259.

SPENCER Jr, B.F.; NAGARAJAIAH, S. **State of the Art of Structural Control.** Journal of Structural Engineering – ASCE. Vol. 129, (9), 845-856, 2003.

SPENCER Jr, B.F.; SAIN, M.K. **Controlling buildings: a new frontier in feedback**. In: T. Samad, guest editor. Emerging Technologies. IEEE Control Systems Magazine 1997; 17(6):19–35 [Special issue].

STANISIC, M. **An Approximate Method Applied to the Solution of the Problem of Vibrating Rectangular Plates.** Journal of the Aeronautical Sciences. Vol. 24(2) 159-160 1957.

SZILARD, Rudolph,. **Theory and analysis of plates ::** classical and numerical methods /. Englewood Cliffs, N. J. : Prentice-Hall, c1974. 724p.

TIMOSHENKO, Stephen,; WOINOWSKY-KRIEGER, S. **Theory of plates and shells** /. 2nd. ed. - New York : McGraw-Hill, Inc. c1959. 580p.

WARBURTON, G.B. **The Vibration of Rectangular Plates.** Proceeding of the Institute of Mechanical Engineers, ser. A. Vol 168. 371-384, 1954.

YOUNG, D. **Vibration of Rectangular Plates by the Ritz Method.** Journal of Applied Mechanics – ASME. Vol. 17. 448-453 1950.

7
Apêndice

Neste apêndice são apresentadas as rotinas utilizadas para aplicação do método de Galerkin Iterativo usando o programa Maple© versão 10. A rotina apresentada é utilizada no cálculo das freqüências naturais e dos modos de vibração, abrangendo as duas primeiras iterações, sendo que as demais são idênticas a estas.

7.1.
Primeira iteração

Na primeira iteração, utiliza-se como função de entrada uma função polinomial obtida para uma viga com as condições de apoio iguais a da placa considerada na direção x, uma vez que, na primeira iteração será obtida uma função para o modo de vibração na direção y.

```
> restart: with(plots): with(LinearAlgebra):
with(Student[Calculus1]): Digits:=32:
```

#E-E-E-E - 1ª Iteração#
#Equação diferencial de placas#

```
> PDE:=(diff(W(x,y),x,x,x,x))+2*(diff(W(x,y),y,y,x,x))+
(diff(W(x,y),y,y,y,y))-M*omega^2*W(x,y)/De:
```

#Modos#

```
> m:=1: n:=1:
```

#Solução polinomial na direção x – m=1, 2.#

```
> EQ:=[-16*x^4/a^4+32*x^3/a^3-16*x^2/a^2,
-8192/27*x^7/a^7+28672/27*x^6/a^6-1536*x^5/a^5+
32000/27*x^4/a^4-13312/27*x^3/a^3+256/3*x^2/a^2]:
> X(x):=EQ[m]:
```

#Solução proposta para a equação diferencial#

```
> W(x,y):=Y(y)*X(x):
> PDE:=simplify(PDE):
```

#Método de Galerkin#

```
> PDE1:=PDE*X(x):
> ODE:=int(PDE1, x=0..a):
```

#Propriedades da placa considerada:#

```
> a:=12:
> b:=12:
> E:=30E9:
> h:=0.2:
> nu:=0.3:
> M:=2500*h:
> De:=E*h^3/(12*(1-nu^2)):
> ODE:=simplify(ODE):
```

#Para facilitar os cálculos serão adotas as constantes na equação a seguir:#

```
> ODE1:=c1*diff(Y(y),y,y,y,y)+c2*diff(Y(y),y,y)+
c3*omega^2*Y(y)+c4*Y(y):
```

#Valores numéricos das constantes empregadas no problema:#

```
> ODE2:=subs(diff(Y(y),y,y,y,y)=A1,diff(Y(y),y,y)=A2,
omega^2*Y(y)=A3,Y(y)=A4,ODE):
> c1:=coeff(ODE2,A1): c2:=coeff(ODE2,A2):
> c3:=coeff(ODE2,A3): c4:=coeff(ODE2,A4):
> Y(y):=_C1*sin(lambda1*y)+_C2*cos(lambda1*y)+
+_C3*sinh(lambda2*y)+_C4*cosh(lambda2*y):
```

#Condições de contorno para dois lados opostos: Engastado-Engastado.#

```
> cond1:=eval(subs(y=0,(eval(W(x,y)/X(x)))):
> cond2:=eval(subs(y=0,eval((diff(W(x,y),y))/X(x)))):
> cond3:=eval(subs(y=b,(eval(W(x,y)/X(x)))):
> cond4:=eval(subs(y=b,eval((diff(W(x,y),y))/X(x)))):
```

#Formulação do problema de autovalor:#

```
> a11:=coeff(cond1, _C1):
> a12:=coeff(cond1, _C2):
> a13:=coeff(cond1, _C3):
> a14:=coeff(cond1, _C4):
> a21:=coeff(cond2, _C1):
> a22:=coeff(cond2, _C2):
> a23:=coeff(cond2, _C3):
> a24:=coeff(cond2, _C4):
> a31:=coeff(cond3, _C1):
> a32:=coeff(cond3, _C2):
> a33:=coeff(cond3, _C3):
> a34:=coeff(cond3, _C4):
> a41:=coeff(cond4, _C1):
> a42:=coeff(cond4, _C2):
> a43:=coeff(cond4, _C3):
> a44:=coeff(cond4, _C4):
> A:=Matrix([[a11,a12,a13,a14],[a21,a22,a23,a24],
[a31,a32,a33,a34],[a41,a42,a43,a44]]):
> detA:=simplify(Determinant(A)):
```

#Soluções do problema de autovalor (Valores de lambda1):#

```
> lambda:=evalf(g=1/2*(2*c1*(c2+(c2^2-4*c1*c3*omega^2-
4*c1*c4)^(1/2)))^(1/2)/c1):
> omega:=abs(rhs(isolate(evalf(lambda), omega))):
```

#Deve ser adotado um valor inicial para lambda2:#

```
> lambda2:=0.5:
```

#Cálculo dos valores de lambda1, lambda2 e determinação das freqüências naturais:#

```
> R:=Vector([0,0,0,0,0,0]):
> R1:=Vector([0,0,0,0,0,0]):
> V:=Vector([0,0,0,0,0,0]):
> R2:=Vector([0,0,0,0,0,0]):
> u1:=0: u2:=0: i:=1:
```

```
> for lambda1 from 0 by 0.001 while i <= 6 do
> u2:=evalf(detA):
> if ((u1/u2)<0) then
> R[i]:=lambda1;
> i:=i+1:
> end if:
> u1:=u2:
> end do:
> unassign('lambda1');
> for i from 1 by 1 to 6 do for j from 1 by 1 to 5 do
> r1:=NewtonsMethod(detA,lambda1=R[i],iterations=10,output
=value):
> g:=r1:
> v:=evalf(omega):
> r2:=1/2*(-2*c1*(c2-(c2^2-4*c1*c3*v^2-
4*c1*c4)^(1/2)))^(1/2)/c1:
> lambda2:=r2:
> unassign('g'):
> end do:
> R1[i]:=r1: R2[i]:=r2: V[i]:=v:
> end do:
```

#Calculos dos parâmetros de freqüência e conversão de unidades:#

```
> K:=Vector([0,0,0,0,0,0]):
> for i from 1 to 6 do
> K[i]:=V[i]*a^2/(De/M)^(1/2):
> end do:
> H:=Vector([0,0,0,0,0,0]):
> for i from 1 to 6 do
> H[i]:=evalf(V[i]/Pi):
> end do:
```

#Resultados#

```
> print(R1,R2,V,K,H):
```

#Determinação dos modos de vibração:#

#Substituindo os valores de lambda1 e lambda2 na matriz A, obtem-se:#

```
> lambda1:=R1[n]:
> lambda2:=R2[n]:
> b11:=evalf(a11):
> b12:=evalf(a12):
> b13:=evalf(a13):
> b14:=evalf(a14):
> b21:=evalf(a21):
> b22:=evalf(a22):
> b23:=evalf(a23):
> b24:=evalf(a24):
> b31:=evalf(a31):
> b32:=evalf(a32):
> b33:=evalf(a33):
> b34:=evalf(a34):
> b41:=evalf(a41):
> b42:=evalf(a42):
> b43:=evalf(a43):
> b44:=evalf(a44):
> B:=Matrix([[b11,b12,b13,b14],[b21,b22,b23,b24],
[b31,b32,b33,b34],[b41,b42,b43,b44]]):
```

#Eliminação da 1ª linha e da 1ª coluna:#

```
> B:=Matrix([[b22,b23,b24],[b32,b33,b34],[b42,b43,b44]]):
> C:=Vector([_C2,_C3,_C4]):
> Q:=MatrixVectorMultiply(B,C):
> U:=Vector([-b21,-b31,-b41]):
> sols:=solve({Q[1]=U[1],
Q[2]=U[2],Q[3]=U[3]},{_C2,_C3,_C4}):
> C:=subs(sols[1],sols[2],sols[3],C):
> norma:=sqrt(((C[1])^2)+((C[2])^2)+((C[3])^2)+1):
> Cons:=Vector([1,C[1],C[2],C[3]])/norma:
> aux1:=_C1=Cons[1]:
> aux2:=_C2=Cons[2]:
```

```
> aux3:=_C3=Cons[3]:
> aux4:=_C4=Cons[4]:
> Y(y):=_C1*sin(lambda1*y)+_C2*cos(lambda1*y)+
_C3*sinh(lambda2*y)+_C4*cosh(lambda2*y):
> Y(y):=subs(aux1,aux2,aux3,aux4,Y(y)):
> W(x,y):=Y(y)*X(x):
> plot3d(W(x,y),y=0..b, x=0..a);
```

#Gravação dos resultados para serem utilizados nas outras iterações:#

```
> arq:= fopen(`C:\\c_c_c_c\\c_c_c_c.txt`,WRITE):
> fprintf (arq, `%1.32f %1.32f\n %1.32f %1.32f %1.32f
%1.32f\n`,lambda1,lambda2,Cons[1],Cons[2],Cons[3],Cons[4]):
> fclose(arq);
> arq1:= fopen(`C:\\c_c_c_c\\dados.txt`,WRITE):
> fprintf (arq1, `%1.2f %1.2f %1.2f\n %11.0f %1.1f %5.2f\n
%2d %2d\n`, a, b, h, E, nu, M, m, n):
> fclose(arq1);
> arq2:= fopen(`C:\\ c_c_c_c\\Freqn.txt`,WRITE):
> for i from 1 to 6 do
> fprintf(arq2, `%2d %3.32f %3.32f\n`, i, K[i], H[i]):
> end do:
> fclose(arq2);
```

7.2.
Segunda iteração

Na segunda iteração são utilizados os dados gravados pela primeira e a função de entrada torna-se a função do modo de vibração obtida na primeira iteração. A partir desta, os resultados já se tornam mais precisos, uma vez que o erro calculado nas demais iterações se torna praticamente nulo.

```
> restart: with(plots): with(LinearAlgebra):
with(Student[Calculus1]): Digits:=32:
```

#E-E-E-E - 2ª Iteração#
#Equação diferencial de placas#

```
> PDE:=(diff(W(x,y),x,x,x,x))+2*(diff(W(x,y),y,y,x,x))+
(diff(W(x,y),y,y,y,y))-M*omega^2*W(x,y)/De:
```

#Solução na direção y – Leitura dos dados gravados#

```
> arq:= fopen(`C:\\c_c_c_c\\c_c_c_c.txt`,READ):
> lista:=fscanf(arq,"%f %f %f %f %f %f/n"):
> lambda1:=lista[1]: lambda2:=lista[2]:
> _C1:=lista[3]:
> _C2:=lista[4]:
> _C3:=lista[5]:
> _C4:=lista[6]:
> fclose(arq);
> Y(y):=_C1*sin(lambda1*y)+_C2*cos(lambda1*y)+
_C3*sinh(lambda2*y)+_C4*cosh(lambda2*y):
> unassign('lambda1,lambda2,_C1,_C2,_C3,_C4');
```

#Solução proposta para a equação diferencial#

```
> W(x,y):=Y(y)*X(x):
> PDE:=simplify(PDE):
```

#Método de Galerkin#

```
> PDE1:=PDE*Y(y):
```

```
> ODE:=ApproximateInt(PDE1, y=0..b, method =
newtoncotes[6]):
```

#Propriedades da placa considerada – Leitura dos dados gravados#

```
> arq1:= fopen(`C:\\c_c_c_c\\dados.txt`,READ):
> lista1:=fscanf(arq,"%f %f %f %f %f %f %d %d/n"):
> a:=lista1[1]:
> b:=lista1[2]:
> h:=lista1[3]:
> E:=lista1[4]:
> nu:=lista1[5]:
> M:=lista1[6]:
> m:=lista1[7]:
> n:=lista1[8]:
> fclose(arq1);
> De:=E*h^3/(12*(1-nu^2)):
> ODE:=simplify(ODE):
> ODE1:=c1*diff(X(x),x,x,x,x)+c2*diff(X(x),x,x)+
c3*omega^2*X(x)+c4*X(x):
```

#Valores numéricos das constantes empregadas no problema:#

```
> ODE2:=subs(diff(X(x),x,x,x,x)=A1,diff(X(x),x,x)=A2,
omega^2*X(x)=A3,X(x)=A4,ODE):
> c1:=coeff(ODE2,A1): c2:=coeff(ODE2,A2):
> c3:=coeff(ODE2,A3): c4:=coeff(ODE2,A4):
> X(x):=_C1*sin(lambda1*x)+_C2*cos(lambda1*x)+
_C3*sinh(lambda2*x)+_C4*cosh(lambda2*x):
```

#Condições de contorno para dois lados opostos: Engastado-Engastado#

```
> cond1:=eval(subs(x=0,(eval(W(x,y)/Y(y))))):
> cond2:=eval(subs(x=0,eval((diff(W(x,y),x))/Y(y)))):
> cond3:=eval(subs(x=a,(eval(W(x,y)/Y(y))))):
> cond4:=eval(subs(x=a,eval((diff(W(x,y),x))/Y(y)))):
```

#Formulação do problema de autovalor:#

```
> a11:=coeff(cond1, _C1):
```

```
> a12:=coeff(cond1, _C2):
> a13:=coeff(cond1, _C3):
> a14:=coeff(cond1, _C4):
> a21:=coeff(cond2, _C1):
> a22:=coeff(cond2, _C2):
> a23:=coeff(cond2, _C3):
> a24:=coeff(cond2, _C4):
> a31:=coeff(cond3, _C1):
> a32:=coeff(cond3, _C2):
> a33:=coeff(cond3, _C3):
> a34:=coeff(cond3, _C4):
> a41:=coeff(cond4, _C1):
> a42:=coeff(cond4, _C2):
> a43:=coeff(cond4, _C3):
> a44:=coeff(cond4, _C4):
> A:=Matrix([[a11,a12,a13,a14],[a21,a22,a23,a24],
[a31,a32,a33,a34],[a41,a42,a43,a44]]):
> detA:=simplify(Determinant(A)):
```

#Soluções do problema de autovalor (Valores de lambda1):#

```
> lambda:=evalf(g=1/2*(2*c1*(c2+(c2^2-4*c1*c3*omega^2-
4*c1*c4)^(1/2)))^(1/2)/c1):
> omega:=abs(rhs(isolate(evalf(lambda), omega))):
```

#Deve ser adotado um valor inicial para lambda2:#

```
> lambda2:=0.5:
```

#Cálculo dos valores de lambda1, lambda2 e determinação das freqüências naturais:#

```
> R:=Vector([0,0,0,0,0,0]):
> R1:=Vector([0,0,0,0,0,0]):
> V:=Vector([0,0,0,0,0,0]):
> R2:=Vector([0,0,0,0,0,0]):
> u1:=0: u2:=0: i:=1:
> for lambda1 from 0 by 0.001 while i <= 6 do
> u2:=evalf(detA):
```

```
> if ((u1/u2)<0) then
> R[i]:=lambda1;
> i:=i+1:
> end if:
> u1:=u2:
> end do:
> unassign('lambda1');
> for i from 1 by 1 to 6 do for j from 1 by 1 to 5 do
> r1:=NewtonsMethod(detA, lambda1=R[i],iterations=10,output
= value):
> g:=r1:
> v:=evalf(omega):
> r2:=1/2*(-2*c1*(c2-(c2^2-4*c1*c3*v^2-
4*c1*c4)^(1/2)))^(1/2)/c1:
> lambda2:=r2:
> unassign('g'):
> end do:
> R1[i]:=r1: R2[i]:=r2: V[i]:=v:
> end do:
```

#Calculos dos parâmetros de freqüência e conversão de unidades:#

```
> K:=Vector([0,0,0,0,0,0]):
> for i from 1 to 6 do
> K[i]:=V[i]*a^2/(De/M)^(1/2):
> end do:
> H:=Vector([0,0,0,0,0,0]):
> for i from 1 to 6 do
> H[i]:=evalf(V[i]/Pi):
> end do:
```

#Resultados#

```
> #print(R1,R2,V,K,H);
```

#Determinação dos modos de vibração:#

#Substituindo os valores de lambda1 e lambda2 na matriz A, obtém-se:#

```
> lambda1:=R1[m]:
```

```
> lambda2:=R2[m]:
> b11:=evalf(a11):
> b12:=evalf(a12):
> b13:=evalf(a13):
> b14:=evalf(a14):
> b21:=evalf(a21):
> b22:=evalf(a22):
> b23:=evalf(a23):
> b24:=evalf(a24):
> b31:=evalf(a31):
> b32:=evalf(a32):
> b33:=evalf(a33):
> b34:=evalf(a34):
> b41:=evalf(a41):
> b42:=evalf(a42):
> b43:=evalf(a43):
> b44:=evalf(a44):
> B:=Matrix([[b11,b12,b13,b14],[b21,b22,b23,b24],
[b31,b32,b33,b34],[b41,b42,b43,b44]]):
```

#Eliminação da 4ª linha e da 4ª coluna:#

```
> B:=Matrix([[b11,b12,b13],[b21,b22,b23],[b31,b32,b33]]):
> C:=Vector([_C1,_C2,_C3]):
> Q:=MatrixVectorMultiply(B,C):
> U:=Vector([-b14,-b24,-b34]):
> sols:=solve({Q[1]=U[1],
Q[2]=U[2],Q[3]=U[3]},{_C1,_C2,_C3}):
> C:=subs(sols[1],sols[2],sols[3],C):
> norma:=sqrt(((C[1])^2)+((C[2])^2)+((C[3])^2)+1):
> Cons:=Vector([C[1],C[2],C[3],1])/norma:
> aux1:=_C1=Cons[1]:
> aux2:=_C2=Cons[2]:
> aux3:=_C3=Cons[3]:
> aux4:=_C4=Cons[4]:
```

```
> X(x):=_C1*sin(lambda1*x)+_C2*cos(lambda1*x)+
_C3*sinh(lambda2*x)+_C4*cosh(lambda2*x):
> X(x):=subs(aux1,aux2,aux3,aux4,X(x)):
> W(x,y):=Y(y)*X(x):
> plot3d(W(x,y),y=0..b, x=0..a);
```

#Gravação dos resultados para serem utilizados nas outras iterações:#

```
> arq:= fopen(`C:\\c_c_c_c\\c_c_c_c2.txt`,WRITE):
> fprintf (arq, `%1.32f %1.32f\n %1.32f %1.32f %1.32f
%1.32f\n`,lambda1,lambda2,Cons[1],Cons[2],Cons[3],Cons[4]):
> fclose(arq);
> arq2:= fopen(`C:\\c_c_c_c\\Freqm2.txt`,WRITE):
> for i from 1 to 6 do
> fprintf (arq2, `%2d %3.32f %3.32f\n`, i, K[i], H[i]):
> end do:
> fclose(arq2);
```