Universidade de Brasília

Implementação de Controle com Redução Modal

Ataias Pereira Reis
Emanuel Pereira Barroso Neto

19 de junho de 2016

1 Introdução

O objetivo deste documento é apresentar os procedimentos necessários para implementar o método de controle apresentado no artigo "*Modal Reduction Based Tracking Control for Installation of Subsea Equipments*", desenvolvido por Fabrício et al, em um controlador industrial da Rockwell. Nem todos os detalhes estão presentes no artigo, o que torna difícil simplesmente lê-lo e realizar o sistema. Algumas modificações no controle serão feitas com base no trabalho do aluno de mestrado Rafael Simões <rafael.domenici@hotmail.com>.

2 Equações Governantes

Para o riser, a Equação 2.1 representa o deslocamento horizontal $\Upsilon(z,t)$ do tubo — um barbante, no caso da bancada de laboratório — sob a ação de forças hidrodinâmicas externas $F_n(z,t)$ (força linear com unidade N/m) e tração $T(z)$ (unidade N):

$$m_s \frac{\partial^2 \Upsilon}{\partial t^2} = -EJ\frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z}\left(T(z)\frac{\partial \Upsilon}{\partial z}\right) + F_n(z,t) \tag{2.1}$$

Antes de prosseguir, é importante definir termos desta equação:

- m_s é a massa linear do barbante (densidade linear, kg/m)
- E é o módulo de Young do barbante e ele é desconhecido

- J é o segundo momento de área e representa a resistência do barbante à flexão. O barbante não apresenta tal resistência, daí $J = 0$
- $T(z)$ é a força de tração e é dada por

$$T(z) = (m_b + zm_s)\, g,$$

sendo m_b a massa da bolinha (kg), $m_s = m_{\text{barbante,kg}}/L$, sendo L o comprimento do barbante, z a posição vertical a partir do carrinho e g é a força da gravidade.

A força externa resultante, $F_n(z,t)$, é dada por

$$F_n(z,t) = -m_{fbar}\frac{\partial^2 \Upsilon}{\partial t^2} - \mu \left|\frac{\partial \Upsilon}{\partial t}\right| \frac{\partial \Upsilon}{\partial t}, \tag{2.2}$$

na qual μ é o coeficiente de arrasto (unidade 1/s) e m_{fbar} é a massa do fluido adicionado, que será posteriormente pormenorizada. Fazendo $m = m_s + m_{fbar}$ e substituindo a Equação 2.2 na 2.1, obtém-se:

$$\frac{\partial^2 \Upsilon}{\partial t^2} = -\frac{EJ}{m}\frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z}\left(\frac{T(z)}{m}\frac{\partial \Upsilon}{\partial z}\right) - \frac{\mu}{m}\left|\frac{\partial \Upsilon}{\partial t}\right| \frac{\partial \Upsilon}{\partial t} \tag{2.3}$$

2.1 Constantes

Significado	**Símbolo**	**Valor**	**Unidade**
Massa	m_{bar}	0.492	g
Comprimento	L	0.82	m
Massa linear	m_s	0.6	g/m
Raio	r_{bar}	1	mm
Densidade	ρ_{bar}	191	kg/m^3

Tabela 2.1: Constantes do barbante

Significado	**Símbolo**	**Valor**	**Unidade**
Massa	m_b	0.492	g
Raio	r_b	15.3	mm
Coeficiente de inércia	C_m	1.2	-
Coeficiente de arrasto	C_d	0.6	-
Volume	V_b	$\frac{4}{3}\pi r_b^3$	m^3
Área da seção transversal	A_b	πr_b^2	m^2

Tabela 2.2: Constantes da bolinha de isopor

Como visto anteriormente, $m = m_s + m_{fbar}$. A massa linear m_{fbar} do fluido adicionado ao redor do barbante é dada por

$$\begin{aligned} m_{fbar} &= 2\pi r_{bar}^2 \rho_{\text{ar}} \\ &= 0.00770 \text{ g/m}. \end{aligned} \tag{2.4}$$

Já que $m_{fbar} \ll m_s$, consideraremos $m = m_s$ nos cálculos. Em relação à massa m_{fb} do fluido adicionado ao redor da bolinha de isopor, ela é dada por

$$\begin{aligned} m_{fb} &= 1.2 V_b \rho_{\text{ar}} \\ &= 1.2 \left(\frac{4}{3} \pi r_b^3 \right) \rho_{\text{ar}} \\ &= 0.0220 \text{ g} \end{aligned} \tag{2.5}$$

$m_{fb} \ll m_b$ também, de forma que os cálculos consideraram $m' = m_b$.

Nota-se que o barbante pesa mais que o isopor, o que faz com que a tração não seja principalmente devida pela bolinha, mas sim pelo barbante. Neste caso, não se utiliza um valor médio para $T(z)$ como no artigo do Fabrício, mas ainda se pode usar um valor médio para as constantes τ e τ', que substituem o termo $\frac{\mu}{m} \left| \frac{\partial \Upsilon}{\partial t} \right|$ para o barbante e para a bolinha, respectivamente. Essas constantes são definidas de acordo com a trajetória prevista, uma vez que a velocidad média depende dessa trajetória. Já levando em conta um valor médio para $\left| \frac{\partial \Upsilon}{\partial t} \right|$, tem-se

$$\frac{\partial^2 \Upsilon}{\partial t^2} = -\frac{EJ}{m} \frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z} \left(\frac{T(z)}{m} \frac{\partial \Upsilon}{\partial z} \right) - \tau \frac{\partial \Upsilon}{\partial t} \tag{2.6}$$

Antes de prosseguirmos para a discretização e obtenção das matrizes em espaço de estados, é importante pensar nas condições de contorno. No topo, $z = L$, tem-se $\Upsilon(L, t) = u(t)$, ou seja, o carrinho se move conforme uma trajetória $u(t)$ definida. Neste mesmo ponto, $\frac{\partial \Upsilon}{\partial z}(L, t) = 0$. Para a ponta na qual a carga está situada, $z = 0$, tem-se $\frac{\partial \Upsilon}{\partial z}(0, t) = \frac{F_L}{T}$, sendo F_L a força aplicada pela ponta do riser na carga. **(Outra coisa que confundi, eu entendi $u(t)$ sendo uma trajetória, pois Υ é deslocamento, mas no artigo do Fabrício está escrito em uma momento que é uma força)**.

2.2 Discretização

De forma a se realizar o controle proposto, o sistema deve ter um espaço de estados finito. Para isso, aplica-se o método de diferenças finitas na coordenada z de maneira a se aproximar a EDP governante em um número finito de EDOs. No espaço discreto, a

equação do k-ésimo elemento é dada por

$$\frac{d^2\Upsilon_k}{dt^2} = -\frac{EJ}{ml^4}\left(\Upsilon_{k-2} - 4\Upsilon_{k-1} + 6\Upsilon_k - 4\Upsilon_{k+1} + \Upsilon_{k+2}\right) + \frac{T_0 + mg(k-1)l}{ml^2}\left(\Upsilon_{k-1} - 2\Upsilon_k + \Upsilon_{k+1}\right) + g\frac{-\Upsilon_{k-1} + \Upsilon_{k+1}}{2l} - \tau\frac{d\Upsilon_k}{dt}, \quad (2.7)$$

sendo N o número de pontos de discretização e l a distância entre dois pontos de discretização ($l = L/N$).

Note que $k \in \mathbb{N} : 2 \leq k \leq N-1$, pois um dos extremos é a bolinha e a equação do pêndulo rege seu movimento enquanto que a outra ponta se aplica uma condição de contorno. O que aconteceria quando $k = 2$ e se precisasse de Υ_{k-2}? Para nosso experimento, $J = 0$ e esse problema não ocorre. Caso se façam testes com um valor de $J \neq 0$, teríamos de resolver esse problema primeiro.

Para simplificar, definem-se as constantes

$$a = -\frac{EJ}{ml^4} \quad (2.8)$$

$$b_k = \frac{T_0 + mg(k-1)l}{ml^2},\ k \geq 2 \quad (2.9)$$

$$c = \frac{g}{2l} \quad (2.10)$$

$$d_k = b_k - c,\ k \geq 2 \quad (2.11)$$

$$e_k = b_k + c,\ k \geq 2 \quad (2.12)$$

A meu ver, a melhor forma de se analisar como as matrizes do sistema ficarão é expandir o sistema para casos com N pequeno e ver o que está ocorrendo. Observe que $a = 0$ para o barbante, o que simplifica os próximos passos.

Para o caso $N = 6$, tem-se

$$\mathbf{x} = \left(\Upsilon_1\ \Upsilon_2\ \Upsilon_3\ \Upsilon_4\ \Upsilon_5\ \Upsilon_6\ \dot{\Upsilon}_1\ \dot{\Upsilon}_2\ \dot{\Upsilon}_3\ \dot{\Upsilon}_4\ \dot{\Upsilon}_5\ \dot{\Upsilon}_6\right)^T \quad (2.13)$$

$$u = \Upsilon(L,t) = \Upsilon_7 \quad (2.14)$$

$$y = \Upsilon(0,t) = \Upsilon_1 \quad (2.15)$$

e as equações são

$$\begin{aligned}
\ddot{\Upsilon}_2 &= b_2\left(\Upsilon_1 - 2\Upsilon_2 + \Upsilon_3\right) + c(-\Upsilon_1 + \Upsilon_3) - \tau\dot{\Upsilon}_2 \\
&= d_2\Upsilon_1 - 2b_2\Upsilon_2 + e_2\Upsilon_3 - \tau\dot{\Upsilon}_2 \qquad (2.16)\\
\ddot{\Upsilon}_3 &= b_3\left(\Upsilon_2 - 2\Upsilon_3 + \Upsilon_4\right) + c(-\Upsilon_2 + \Upsilon_4) - \tau\dot{\Upsilon}_3 \\
&= d_3\Upsilon_2 - 2b_3\Upsilon_3 + e_3\Upsilon_4 - \tau\dot{\Upsilon}_3 \qquad (2.17)\\
\ddot{\Upsilon}_4 &= b_4\left(\Upsilon_3 - 2\Upsilon_4 + \Upsilon_5\right) + c(-\Upsilon_3 + \Upsilon_5) - \tau\dot{\Upsilon}_4 \\
&= d_4\Upsilon_3 - 2b_4\Upsilon_4 + e_4\Upsilon_5 - \tau\dot{\Upsilon}_4 \qquad (2.18)\\
\ddot{\Upsilon}_5 &= b_5\left(\Upsilon_4 - 2\Upsilon_5 + \Upsilon_6\right) + c(-\Upsilon_4 + \Upsilon_6) - \tau\dot{\Upsilon}_5 \\
&= d_5\Upsilon_4 - 2b_5\Upsilon_5 + e_5\Upsilon_6 - \tau\dot{\Upsilon}_5 \qquad (2.19)\\
\ddot{\Upsilon}_6 &= b_6\left(\Upsilon_5 - 2\Upsilon_6 + u\right) + c(-\Upsilon_5 + u) - \tau\dot{\Upsilon}_6 \\
&= d_6\Upsilon_5 - 2b_6\Upsilon_6 + e_6u - \tau\dot{\Upsilon}_6 \qquad (2.20)
\end{aligned}$$

A equação para a posição da carga Υ_1 leva em conta a massa da bolinha e a força de Morison:

$$m_b\ddot{\Upsilon}_1 = \frac{m_b g}{l}\left(\Upsilon_2 - \Upsilon_1\right) + \rho_{\text{ar}}C_m V_b\ddot{\Upsilon}_1 - \frac{1}{2}\rho_{\text{ar}}C_d A_b\dot{\Upsilon}_1\left|\dot{\Upsilon}_1\right|, \qquad (2.21)$$

e, isolando-se $\ddot{\Upsilon}_1$, tem-se

$$\ddot{\Upsilon}_1 = \frac{m_b g}{m' l}\left(\Upsilon_2 - \Upsilon_1\right) - \frac{1}{2m'}\rho C_d A_b\dot{\Upsilon}_1\left|\dot{\Upsilon}_1\right|. \qquad (2.22)$$

Note que $m' = m_b + \rho_{\text{ar}}C_m V_b = m_b + m_{fb} \approx m_b$. Assim, assume-se $m' = m_b$ para os cálculos.

Anteriormente, foi apresentada a linearização τ para o termo $\frac{1}{2m}\rho C_d A\left|\dot{\Upsilon}_k\right|$ do cabo. Assumo que isso também seja necessário para a bola, resultando em um τ':

$$\ddot{\Upsilon}_1 = b_1\left(-\Upsilon_1 + \Upsilon_2\right) - \tau'\dot{\Upsilon}_1, \qquad (2.23)$$

com

$$b_1 = \frac{m_b g}{m' l} = \frac{g}{l}. \qquad (2.24)$$

Desta forma, pode-se definir o sistema linear em forma matricial

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -b_1 & b_1 & 0 & 0 & 0 & 0 & -\tau' & 0 & 0 & 0 & 0 & 0 \\ d_2 & -2b_2 & e_2 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 & 0 & 0 \\ 0 & d_3 & -2b_3 & e_3 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 & 0 \\ 0 & 0 & d_4 & -2b_4 & e_4 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 \\ 0 & 0 & 0 & d_5 & -2b_5 & e_5 & 0 & 0 & 0 & 0 & -\tau & 0 \\ 0 & 0 & 0 & 0 & d_6 & -2b_6 & 0 & 0 & 0 & 0 & 0 & -\tau \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ e_6 \end{bmatrix} u \tag{2.25}$$

que pode ser representado concisamente como

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{6\times6} & \mathbf{I}_{6\times6} \\ \mathbf{M}_{6\times6} & \mathbf{L}_{6\times6} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{11\times1} \\ e_6 \end{bmatrix} u \tag{2.26}$$

Para o caso de uma discretização com N pontos, tem-se

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \\ \mathbf{M}_{N\times N} & \mathbf{L}_{N\times N} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{2N-1\times1} \\ e_N \end{bmatrix} u \tag{2.27}$$

$$y = \begin{bmatrix} 1 & \mathbf{0}_{1\times2N-1} \end{bmatrix} \mathbf{x} \tag{2.28}$$

2.3 Código para calcular matrizes

Listing 1: Código para gerar matrizes A, B e C

```
module ModalReduction
export generateA, generateB, generateC
export generateABC, getABC_M, getABCD_R
export manuscript_p48, simulation
export generateMATLABSimulationScript


function simulation(n,original=false,nout=4)
  A, B, C = generateABC(n)
  A_M, B_M, C_M = getABC_M(n, A, B, C)
  A_R, B_R, C_R, D_R = getABCD_R(n, A_M, B_M, C_M,nout)
  if original
    generateMATLABSimulationScriptToCompare("simulacaoN" * string(n)
        * "Compare.m", A, B, C, A_R, B_R, C_R, D_R)
    generateMATLABSimulationScript(n, "simulacaoN" * string(n) *
        "Reduced.m", A_R, B_R, C_R, D_R)
```

```
  else
    generateMATLABSimulationScript(n, "simulacaoN" * string(n) *
        "Reduced.m", A_R, B_R, C_R, D_R)
  end
end

#Gera A, B, C to sistema completo
function generateABC(n)
  tau = 0.2426      # tau do barbante (1/s) para excursão de 30cm
  taul = 0.1133     # tau da bolinha (1/s) para excursão de 30cm
  ms = 0.0006       # massa linear do barbante (kg/m)
  mb = 0.00015      # massa da bolinha (kg)
  g = 9.80665       # aceleração da gravidade (m/s^2)
  L = 0.82        # Comprimento total do barbante (m)
  l = L/n         # distância entre dois pontos de discretização (m)
  T0 = mb*g       # Tração no ponto 0 (logo acima da bolinha) -
      considerando peso da bolinha (N)

  b = zeros(n)
  c = g/(2l)
  d = zeros(n)
  e = zeros(n)

  b[1] = g/l
  for k = 2:n
    b[k] = (T0 + ms*g*(k-1)*l)/(ms*l^2)
    d[k] = b[k] - c
    e[k] = b[k] + c
  end

  A = generateA(n, b, d, e, tau, taul)
  B = generateB(n,e[n])
  C = generateC(n)

  return A, B, C
end

function generateA(n, b, d, e, tau, taul)
  M = zeros(n,n)
  #Primeira linha de M
  M[1,1] = -b[1]
  M[1,2] = b[1]

  #Linhas 2 ate n-1
  for i = 2:n-1
    M[i,i-1] = d[i]
    M[i,i] = -2*b[i]
    M[i,i+1] = e[i]
  end
```

```
62
63    #Linha n
64    M[n,n-1] = d[n]
65    M[n,n] = -2*b[n]
66
67    L = eye(n)
68    for i = 1:n
69      L[i,i] = i == 1 ? -taul : -tau
70    end
71
72    #Concatenar matrizes, gerando matriz (2n,2n)
73    A = [[zeros(n,n) eye(n)]; [M L]]
74
75    return A
76  end
77
78  function generateB(n, eN)
79    B = zeros(2*n)
80    B[2*n] = eN
81
82    return B
83  end
84
85  function generateC(n)
86    C = zeros(1,2*n)
87    C[1,1] = 1
88
89    return C
90  end
91
92  function getT(n, A)
93    eig_A = eigvals(A)
94    Tcomplex = eigvecs(A)
95
96    T = zeros(2*n, 2*n)
97    i = 1
98    while i <= 2*n #será que tem algo errado? tem de testar
99      if abs(imag(eig_A[i])) > 1e-10
100       T[:,i] = real(Tcomplex[:,i])
101       T[:,i+1] = -imag(Tcomplex[:,i])
102       i = i + 2
103     else
104       T[:,i] = Tcomplex[:,i]
105       i = i + 1
106     end
107   end
108
109   return T
110 end
```

```
111
112 function getABC_M(n, A, B, C)
113   T = getT(n,A)
114
115   A_M = T \ A * T
116   B_M = T \ B
117   C_M = C * T
118
119   return A_M, B_M, C_M
120 end
121
122 function getABCD_R(n, A_M, B_M, C_M,n_out=4)
123   C_M_diag = diagm(vec(C_M)) #matriz diagonal
124   G = C_M_diag / A_M * B_M #ganhos
125   subsystems = getSubsystems(n, eigvals(A_M), G)
126   A_R = zeros(n_out,n_out)
127   B_R = zeros(n_out)
128   C_R = zeros(1,n_out)
129
130   #Construir matrizes
131   i = 1
132   j = 1
133   while i <= n_out
134     index = subsystems[j][2]
135     if  length(index) == 2 #complexo
136       A_R[i:i+1,i:i+1] = A_M[index, index]
137       B_R[[i,i+1]] = B_M[index]
138       C_R[1,[i,i+1]] = C_M[index]
139       i = i + 2
140     else
141       A_R[i,i] = A_M[index[1],index[1]]
142       B_R[i] = B_M[index[1]]
143       C_R[1,i] = C_M[index[1]]
144       i = i + 1
145     end
146     j = j + 1
147   end
148
149   D_R = C_M / A_M * B_M - C_R / A_R * B_R
150   return A_R, B_R, C_R, D_R
151 end
152
153 function getSubsystems(n, eig_A, G)
154   i = 1
155   subsystems = []
156   while i <= 2*n
157     if abs(imag(eig_A[i])) > 1e-10 #complexo
158       gain = abs(G[i] + G[i+1]) #considera sinal na soma como aqui
          ou soma os módulos?
```

```
      append!(subsystems, [(gain, [i,i+1])])
      i = i + 2
    else #real
      gain = abs(G[i])
      append!(subsystems, [(gain, [i])])
      i = i + 1
    end
  end
  #ordena pelo primeiro elemento da tupla
  #ordem descendente
  sort!(subsystems,rev=true)
  return subsystems
end

function manuscript_p48()
  n = 2
  M = [[-1 1]; [1 -3]]
  L = -2 * eye(n)
  A = [[zeros(n,n) eye(n)];[M L]]
  B = generateB(2,2)
  C = generateC(n)

  A_M, B_M, C_M = getABC_M(n,A,B,C)
  A_R, B_R, C_R, D_R = getABCD_R(n, A_M, B_M, C_M, 4)

  return A_R, B_R, C_R, D_R
end

function generateMATLABSimulationScript(n, filename, A, B, C, D)
  output = "A = " * string(A) * ";\n\n"
  output = output * "B = " * string(B) * "';\n\n"
  output = output * "C = " * string(C) * ";\n\n"
  output = output * "D = " * string(D) * ";\n\n"
  output = output * "sys = ss(A, B, C, D);\n"
  output = output * "opt = stepDataOptions;"
  output = output * "opt.InputOffset = 0;\n"
  output = output * "opt.StepAmplitude = 0.3;\n"
  output = output * "t = (0:0.01:50)';\n"
  output = output * "y = step(sys, t, opt);"

  output = output * "fig = figure;\n"
  output = output * "hold on;\n"
  output = output * "plot(t,y,'k-');\n"
  output = output * "xlabel('Time (s)'), ylabel('Position (m)');\n"
  output = output * "title('Sistema original N = " * string(n) * ",
     simulacao reduzida para ordem 4');\n"
  output = output * "print('SimulationReduceN" * string(n) * "',
     '-dpng', '-r300');\n"
  output = output * "close(fig);\n"
```

```
206
207   file = open(filename, "w")
208   write(file, output)
209   close(file)
210 end
211
212 function generateMATLABSimulationScriptToCompare(filename, A, B, C,
      A_R, B_R, C_R, D_R)
213   output = "A_R = " * string(A_R) * ";\n\n"
214   output = output * "B_R = " * string(B_R) * "';\n\n"
215   output = output * "C_R = " * string(C_R) * ";\n\n"
216   output = output * "D_R = " * string(D_R) * ";\n\n"
217   output = output * "sysR = ss(A_R, B_R, C_R, D_R);\n\n"
218
219   output = output * "A = " * string(A) * ";\n\n"
220   output = output * "B = " * string(B) * "';\n\n"
221   output = output * "C = " * string(C) * ";\n\n"
222   output = output * "sysO = ss(A, B, C, [0]);\n\n"
223
224   output = output * "opt = stepDataOptions;"
225   output = output * "opt.InputOffset = 0;\n"
226   output = output * "opt.StepAmplitude = 0.3;\n\n"
227
228   output = output * "t = (0:0.01:50)';\n"
229   output = output * "yR = step(sysR, t, opt);\n"
230   output = output * "yO = step(sysO, t, opt);\n\n"
231
232   output = output * "fig = figure;\n"
233   output = output * "hold on;\n"
234   output = output * "plot(t,yR,'k--');\n"
235   output = output * "plot(t,yO,'k');\n"
236   output = output * "legend('Reduzido', 'Original');\n"
237   output = output * "xlabel('Time (s)'), ylabel('Position (m)');\n"
238   n = round(Int,size(A)[1]/2)
239   output = output * "title('Simulacao com sistema original N = " *
        string(n) * " e reduzido N = 4');\n"
240   output = output * "print('SimulationN" * string(n) * "', '-dpng',
        '-r300');\n"
241   output = output * "close(fig);\n"
242
243   file = open(filename, "w")
244   write(file, output)
245   close(file)
246 end
247
248
249 end
```

3 Uma Estratégia de Redução da Ordem do Modelo

A maior parte da teoria clássica de controle lida com sistemas representados por um pequeno número de variáveis de estado. Portanto, uma forma de aplicar métodos clássicos de controle da literatura para sistemas de parâmetros distribuídos discretos é por meio de uma redução da ordem do modelo.

Tal redução do modelo será feita em duas etapas: primeiro, uma transformação modal é aplicada nas equações originais do espaço de estados, resultando em uma nova representação em variáveis modais. Nesta forma, o sistema pode ser visto como um conjunto de subsistemas dissociados em paralelo, cuja influência na saída pode ser calculada individualmente. Então, os subsistemas com os maiores ganhos estáticos são escolhidos para criar um modelo de ordem reduzida.

3.1 Decomposição Modal

Primeiro, deve-se obter os autovalores do espaço de estados do *riser*. Observa-se que eles são sempre distintos entre si, uma condição suficiente para a diagonalização da matriz do espaço de estados. Assim, calcula-se a matriz modal $\mathbf{T}$, cuja i-ésima coluna é o i-ésimo autovetor do sistema:

$$\mathbf{T} = (\ \mathbf{v_1} \mid \mathbf{v_2} \mid \ \ldots \ \mid \mathbf{v_{2N}}\ )_{1\times 2N} \tag{3.1}$$

Como a matriz $\mathbf{T}$ provavelmente tem valores complexos devidos a autovalores complexos. Isso é um problema para a representação em espaço de estados e sua simulação. A solução é criar uma matriz $\tilde{\mathbf{T}}$ que tenha só números reais. Antes de explicar como criá-la, lembre que os autovalores complexos sempre aparecem em pares conjugados já que a matriz $\mathbf{A}$ só tem valores reais. Quando a primeira coluna de um autovetor de um par complexo conjugado for encontrada, a coluna respectiva de $\tilde{\mathbf{T}}$ será sua parte real. A segunda coluna desse par complexo conjugado será a parte imaginária da coluna de $\mathbf{T}$.

A matriz $\tilde{\mathbf{T}}$ é utilizada para uma transformação de similaridade no sistema original:

$$\mathbf{A_M} = \tilde{\mathbf{T}}^{-1}\mathbf{A}\tilde{\mathbf{T}}, \tag{3.2}$$

$$\mathbf{x_M} = \tilde{\mathbf{T}}^{-1}\mathbf{x}, \tag{3.3}$$

$$\mathbf{B_M} = \tilde{\mathbf{T}}^{-1}\mathbf{B}, \text{ e} \tag{3.4}$$

$$\mathbf{C_M} = \mathbf{C}\tilde{\mathbf{T}}. \tag{3.5}$$

O sistema transformado, denotado pelo subscrito $\mathbf{M}$, é mais adequado à análise. $\mathbf{A_M}$ é uma matriz diagonal, com seus autovalores explícitos, e permitindo o desacoplamento do sistema original em N subsistemas de segunda ordem formados por pares de autovalores reais ou complexo-conjugados.

3.2 Redução Modal

Neste estágio, procura-se determinar quais dos subsistemas são mais adequados para aproximar o modelo original por meio do cálculo do ganho estático de cada um. Este método depende da predominância de uns poucos autovalores na resposta do sistema, já que altas frequências são muito atenuadas pelas forças hidrodinâmicas e pela suavidade da entrada.

Os subsistemas selecionados são combinados em um modelo reduzido

$$\dot{\mathbf{z}} = \mathbf{A_R z} + \mathbf{B_R} u \tag{3.6}$$
$$y = \mathbf{C_R z} + \mathbf{D_R} u \tag{3.7}$$

cuja ordem é escolhida considerando o custo-benefício entre a acurácia da dinâmica reduzida e a simplicidade da estrutura de controle exigida. Além disso, o sistema reduzido deve compensar o ganho estático perdido nos autovalores desconsiderados. Isto é feito por meio de uma matriz de transferência direta $\mathbf{D_R}$, que é a diferença dos ganhos dos sistemas original e reduzido:

$$\mathbf{D_R} = \mathbf{C A^{-1} B} - \mathbf{C_R A_R^{-1} B_R} \tag{3.8}$$

O subsistemas são de ordem 1 ou 2 dependendo se o autovalor é real ou um par complexo conjugado.

A matriz de transferência direta $\mathbf{D_R}$ introduz novas dinâmicas: uma saída não-nula que não leva em conta o atraso de propagação da entrada e um ganho em altas frequências. Conforme mostrado por Fortaleza (2009), podemos refinar o modelo reduzido introduzindo um atraso de entrada ϵ que minimiza a transferência direta e garante dinâmica nula para $t < \epsilon$:

$$\begin{aligned} \dot{\mathbf{z}} &= \mathbf{A_R z} + \mathbf{B_D} u(t-\epsilon) \\ y &= \mathbf{C_R z} + \mathbf{D_D} u(t-\epsilon) \end{aligned} \tag{3.9}$$

sendo

$$\mathbf{B_D} = \mathbf{A_M} \left(e^{\epsilon \mathbf{A_M}} \right) \mathbf{A_M^{-1} B_M} \tag{3.10}$$
$$\mathbf{D_D} = \mathbf{C_M} \left(e^{\epsilon \mathbf{A_M}} - \mathbf{I} \right) \mathbf{A_M^{-1} B_M} + \mathbf{D_M} \tag{3.11}$$

O novo modelo reduzido (3.9) é tal que, para uma entrada degrau no instante t', a saída mantém seu valor inicial enquanto $t < t' + \epsilon$. Para $t \geq t' + \epsilon$, ambos os modelos reduzidos produzem a mesma saída. O atraso ϵ pode ser visto como uma aproximação para o atraso natural de propagação da estrutura.