Universidade de Brasília

Implementação de Controle com Redução Modal

Ataias Pereira Reis
Emanuel Pereira Barroso Neto

8 de abril 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 (adimensional ? deveria ser, mas há algo que não bate) 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 substitui o termo $\frac{\mu}{m} \left| \frac{\partial \Upsilon}{\partial t} \right|$ para o barbante e para a bolinha, respectivamente. 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

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

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

$$a = -\frac{EJ}{ml^4} \tag{2.8}$$

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

$$c = \frac{g}{2l} \tag{2.10}$$

$$d_k = b_k - c,\ k \geq 2 \tag{2.11}$$

$$e_k = b_k + c,\ k \geq 2 \tag{2.12}$$

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

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

$$u = \Upsilon(L,t) = \Upsilon_7 \tag{2.14}$$

$$y = \Upsilon(0,t) = \Upsilon_1 \tag{2.15}$$

e as equações são

$$\begin{aligned}\ddot{\Upsilon}_2 &= b_2\left(\Upsilon_1 - 2\Upsilon_2 + \Upsilon_3\right) + c(-\Upsilon_1 + \Upsilon_3) - \tau\dot{\Upsilon}_2 \\ &= d_2\Upsilon_1 - 2b_2\Upsilon_2 + e_2\Upsilon_3 - \tau\dot{\Upsilon}_2 \end{aligned} \tag{2.16}$$

$$\begin{aligned}\ddot{\Upsilon}_3 &= b_3\left(\Upsilon_2 - 2\Upsilon_3 + \Upsilon_4\right) + c(-\Upsilon_2 + \Upsilon_4) - \tau\dot{\Upsilon}_3 \\ &= d_3\Upsilon_2 - 2b_3\Upsilon_3 + e_3\Upsilon_4 - \tau\dot{\Upsilon}_3 \end{aligned} \tag{2.17}$$

$$\begin{aligned}\ddot{\Upsilon}_4 &= b_4\left(\Upsilon_3 - 2\Upsilon_4 + \Upsilon_5\right) + c(-\Upsilon_3 + \Upsilon_5) - \tau\dot{\Upsilon}_4 \\ &= d_4\Upsilon_3 - 2b_4\Upsilon_4 + e_4\Upsilon_5 - \tau\dot{\Upsilon}_4 \end{aligned} \tag{2.18}$$

$$\begin{aligned}\ddot{\Upsilon}_5 &= b_5\left(\Upsilon_4 - 2\Upsilon_5 + \Upsilon_6\right) + c(-\Upsilon_4 + \Upsilon_6) - \tau\dot{\Upsilon}_5 \\ &= d_5\Upsilon_4 - 2b_5\Upsilon_5 + e_5\Upsilon_6 - \tau\dot{\Upsilon}_5 \end{aligned} \tag{2.19}$$

$$\begin{aligned}\ddot{\Upsilon}_6 &= b_6\left(\Upsilon_5 - 2\Upsilon_6 + u\right) + c(-\Upsilon_5 + u) - \tau\dot{\Upsilon}_6 \\ &= d_6\Upsilon_5 - 2b_6\Upsilon_6 + e_6 u - \tau\dot{\Upsilon}_6 \end{aligned} \tag{2.20}$$

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

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

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

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

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':

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

com

$$b_1 = \frac{m_b g}{m'(N-1)l} = \frac{g}{(N-1)l}. \tag{2.24}$$

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

```
# -*- coding: utf-8 -*-

from numpy import *
from numpy.linalg import *

#a is a numpy matrix
def print_matriz(a):
  (I,J) = a.shape # Determina as dimensões da matriz a ser impressa
  for i in range(0,I):
    st = ''
    for j in range(0,J):
      #Números de valor absoluto menor que um threshold são
         mostrados como 0 para facilitar visualização
      st += '{:12.2}'.format((a[i,j] if (abs(a[i,j]) > 1e-10) else
         0.0))
      if (j != J - 1):
        st += ' '
    print(st)
  print()

def generateA(n, b, d, e, tau, taul):
  L = (-tau) * eye(n)
  L[0][0] = (-taul)
  M = zeros((n,n))
  for k in range(0,n):
    M[k][k] = -2*b[k] if k != 0 else -b[k]
    if k != n-1:
      M[k+1][k] = d[k+1]
      M[k][k+1] = e[k] if k != 0 else b[k]
  A = vstack((hstack((zeros((n,n)),eye(n))),hstack((M,L))))
  return matrix(A)

def generateB(n, e):
  B = zeros((2*n,1))
  B[2*n-1,0] = e[n-1]
  return matrix(B)

def generateC(n):
  C = zeros((1,2*n))
  C[0,0] = 1        # Como C é uma matriz linha, as duas dimensões
     são necessárias; senão, toda a matriz C valerá 1.
  return matrix(C)

def getReducedSystem(A,B,C,n):
  eig_A,T = eig(A)  # eig_A são os autovalores de A, e T é a matriz
     de autovetores
```

```
43
44  # print('Autovalores de A')
45  # print_matriz(matrix(eig_A))
46
47  # print('Matriz T')
48  # print_matriz(T)
49  # print()
50
51  #A Matriz T não é a que deve ser utilizada para a transformação de
        similaridade!
52  #Ela tem números complexos e isso é ruim
53  #Ela deve ser convertida em uma matriz que tenha só números reais
54
55  Tnew = matrix(zeros((2*n,2*n))) #real! não é complexo
56  i = 0
57  while i < 2*n: #será que tem algo errado? tem de testar
58    if imag(eig_A[i]) > 1e-10: #complexo
59      Tnew[:,i] = real(T[:,i])
60      Tnew[:,i+1] = -imag(T[:,i])
61      i = i + 2
62    else:
63      Tnew[:,i] = T[:,i]
64      i = i + 1
65
66  # print('Matriz T nova')
67  # print_matriz(Tnew)
68  # print()
69
70  del T
71  T = Tnew
72
73  T_inv = matrix(inv(T))
74  A_M = T_inv * A * T
75  B_M = T_inv * B
76  C_M = C * T
77  C_M_diag = matrix(zeros((2*n,2*n), complex))
78
79
80  # print('Matriz A_M')
81  # print_matriz(A_M)
82  # print()
83  #
84  # print('Matrix B_M')
85  # print_matriz(B_M)
86  # print()
87  #
88  # print('Matrix C_M transposta')
89  # print_matriz(C_M.transpose())
90  # print()
```

```
91
92    for i in range(0,2*n):
93      C_M_diag[i,i] = C_M[0,i]
94    A_M_inv = inv(A_M)
95    Gains = C_M_diag * A_M_inv * B_M
96    # print('Ganhos:')
97    # print_matriz(Gains)
98    # print()
99
100   gdim = (Gains.shape[0] // 2)
101   GainSum = zeros((gdim,1))
102
103   #Como todos os autovalores são complexos, cada subsistema 2x2 é
          composto
104   #de um autovalor e seu conjugado que é um outro autovalor
105   for i in range(0,Gains.shape[0],2):
106     GainSum[i//2] = real(abs(Gains[i] + Gains[i+1]))
107
108   print('Ganhos dos subsistemas 2x2 (todos os autovalores são
          complexos):')
109   print_matriz(GainSum)
110   print()
111
112   #Obter maiores ganhos e índices
113   gain = array([max(GainSum)])
114   lines = array([argmax(GainSum)*2-2, argmax(GainSum)*2-1])
115   GainSum = delete(GainSum, argmax(GainSum))
116   beginning = 0
117   gain = insert(gain, beginning, max(GainSum))
118   lines = insert(lines, beginning, array([argmax(GainSum)*2-2,
          argmax(GainSum)*2-1]))
119   print('Maiores ganhos')
120   print(gain)
121   print('Índices')
122   print(lines)
123   print()
124
125   print('Autovalores')
126   print_matriz(matrix(eig_A[lines]))
127   print()
128
129   print('Matriz A_R')
130   eig1 = eig_A[lines[0]]
131   eig2 = eig_A[lines[2]]
132   A_R = matrix(array([[real(eig1), imag(eig1),  0,
                          0],
133             [-imag(eig1), real(eig1), 0,                    0],
134             [0,           0,          real(eig2),  imag(eig2)],
135             [0,           0,         -imag(eig2),  real(eig2)]]))
```

```
136   print_matriz(A_R)
137   print()
138
139   print('Vetor B_R')
140   B_R = B_M[lines]
141   print_matriz(B_R)
142   print()
143
144   print('Vetor C_R')
145   C_R = C_M[0,lines]
146   print_matriz(C_R)
147   print()
148
149   print('Constante D_R')
150   D_R = C*inv(A)*B - C_R*inv(A_R)*B_R
151   print_matriz(D_R)
152   print()
153
154 def main():
155   n = 300
156   tau = 0.2426  # tau do barbante
157   taul = 0.1133 # tau da bolinha
158   ms = 0.0006   # massa linear do barbante (kg/m)
159   mb = 0.00015  # massa da bolinha (kg)
160   g = 9.807   # aceleração da gravidade (m/s^2)
161   L = 0.82    # Comprimento total do barbante (m)
162   l = L/n     # distância entre dois pontos de discretização
163   T0 = mb*g   # Tração no ponto 0 (logo acima da bolinha) -
          considerando peso da bolinha (N)
164
165   b = zeros((n,1))
166   c = g/(2*l)
167   d = zeros((n,1))
168   e = zeros((n,1))
169   b[0] = g/((n-1)*l)
170   for k in range(2,n+1):
171     b[k-1] = (T0 + ms*g*(k-1)*l)/(ms*l**2)
172     d[k-1] = b[k-1] - c
173     e[k-1] = b[k-1] + c
174
175   # b = [1,2,3,4,5,6]
176   # d = [0,0.2,0.3,0.4,0.5,0.6] # Teste
177   # e = [0,12,13,14,15,16]
178
179   A = generateA(n, b, d, e, tau, taul)
180   # print('Matriz A:')
181   # print_matriz(A)
182   # print()
183   B = generateB(n,e)
```

```
184   # print('Matriz B:')
185   # print_matriz(B)
186   # print()
187   C = generateC(n)
188   # print('Matriz C:')
189   # print_matriz(C)
190   # print()
191   getReducedSystem(A,B,C,n)
192
193 if __name__ == "__main__":
194   main()
```

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:

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

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

$$\dot{\mathbf{z}} = \mathbf{A_R}\mathbf{z} + \mathbf{B_R}u \tag{3.6}$$

$$y = \mathbf{C_R}\mathbf{z} + \mathbf{D_R}u \tag{3.7}$$

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:

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

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 $t < \epsilon$:

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

sendo

$$\mathbf{B_D} = \mathbf{A_M}\left(e^{\epsilon \mathbf{A_M}}\right)\mathbf{A_M^{-1}}\mathbf{B_M} \tag{3.10}$$

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

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.

4 Projeto de Controle

4.1 Planejamento Offline de Trajetória

A dinâmica do modelo de ordem reduzida (3.9) é comparada com aquela do sistema original. Escolhendo uma redução para ordem 4, com dois pares de autovalores complexo-conjugados ($\lambda_i, \overline{\lambda}_i$, com $\lambda_i = \sigma_i + jw_i$), a equação do sistema em espaço de estados é

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} \sigma_1 & w_1 & 0 & 0 \\ -w_1 & \sigma_1 & 0 & 0 \\ 0 & 0 & \sigma_2 & w_2 \\ 0 & 0 & -w_2 & \sigma_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} u(t-\epsilon) \tag{4.1}$$

$$y = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} + (d)u(t-\epsilon) \tag{4.2}$$

As constantes b_i e c_i são diferentes das definidas anteriormente.

Normalmente o problema de planejamento de trajetória consiste em encontrar um controle em malha aberta $u^*(t)$, de forma que as variáveis de estado sigam uma trajetória $z^*(t)$. O problema aqui é que $z^*(t)$ carece de interpretação física; ou seja, $z^*(t)$ não se relaciona de maneira clara com nenhuma variável de estado original do *riser*, e, portanto, com a operação de re-entrada. Uma forma de se lidar com isso é lidar com a trajetória da saída *flat* do sistema.

Para um sistema ser *flat*, é condição suficiente ele ser plenamente controlável, ou seja, a matriz de controlabilidade $K = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}$ deve ser de posto pleno. A partir de z, podemos obter uma saída *flat* (f), dada por uma combinação linear das variáveis de estado (z_1, z_2, z_3, z_4) obtida utilizando-se a última linha de K^{-1}:

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} K^{-1} z \tag{4.3}$$

$$f = \alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3 + \alpha_4 z_4 \tag{4.4}$$

Deseja-se que a entrada u dependa apenas de f e suas derivadas temporais. Para isso, é necessário que f seja tantas vezes derivável quanto indica a ordem do sistema. No presente estudo, f deve ser ao menos 4 vezes derivável. Derivando f de acordo com a Equação 4.4, temos:

$$\begin{array}{lcl} \dot{f} & = & \alpha_5 z_1 + \alpha_6 z_2 + \alpha_7 z_3 + \alpha_8 z_4 \\ \ddot{f} & = & \alpha_9 z_1 + \alpha_{10} z_2 + \alpha_{11} z_3 + \alpha_{12} z_4 \\ \dddot{f} & = & \alpha_{13} z_1 + \alpha_{14} z_2 + \alpha_{15} z_3 + \alpha_{16} z_4 \\ f^{(4)} & = & \alpha_{17} z_1 + \alpha_{18} z_2 + \alpha_{19} z_3 + \alpha_{20} z_4 + \gamma u \end{array} \tag{4.5}$$

Das Equações 4.4 e 4.5, podemos definir a seguinte matriz M, sabendo que

$$\begin{pmatrix} f \\ \dot{f} \\ \ddot{f} \\ \dddot{f} \\ f^{(4)} \end{pmatrix} = M \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ u \end{pmatrix} \tag{4.6}$$

Pela Equação 4.6:

$$M = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 \\ \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 \\ \alpha_9 & \alpha_{10} & \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} & 0 \\ \alpha_{17} & \alpha_{18} & \alpha_{19} & \alpha_{20} & \gamma \end{pmatrix} \tag{4.7}$$

Para se obter u em função de uma trajetória f, podemos utilizar a Equação 4.6. Multiplicando os dois membros da equação à esquerda por M^{-1}, vem

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ u \end{pmatrix} = M^{-1} \begin{pmatrix} f \\ \dot{f} \\ \ddot{f} \\ \dddot{f} \\ f^{(4)} \end{pmatrix} \tag{4.8}$$

Observando-se a Equação 4.8, é trivial notar que u será uma combinação linear de f e suas derivadas temporais, sendo os coeficientes fornecidos pela última linha da matriz M^{-1}. COm efeito, podemos escrever:

$$u = \beta_0 f + \beta_1 \dot{f} + \beta_2 \ddot{f} + \beta_3 \dddot{f} + \beta_4 f^{(4)} \tag{4.9}$$

O próximo passo, agora, é planejar a trajetória por meio de $f^*(t)$. O modo mais simples é utilizar uma interpolação polinomial, com coeficientes a_i escolhidos de tal forma que o valor de $f^*(t)$ seja nulo até o tempo de início da operação ($t = t'$); também é necessário

que $f^*(t)$ atinja um valor constante c_f após um tempo de operação t_f ($t = t' + t_f$). Além disso, é necessário que as cinco primeiras derivadas temporais de $f^*(t)$ sejam nulas nos extremos da operação, de forma a assumir uma trajetória suave de referência para $f^{(4)}$ na vizinhança de c_f. Assumindo-se um polinômio de grau 11 para $f^*(t)$, vem

$$f^*(t) = \begin{cases} 0, \forall t < t' \\ \sum_{k=0}^{11} a_i t^k \\ c_f, \forall t > t_f + t' \end{cases} \tag{4.10}$$

4.2 Trajetória em Malha Fechada

Após o planejamento da trajetória, com a obtenção de $f^*(t)$, o próximo passo é obter uma lei de controle em malha fechada para que se corrija o erro de trajetória dado por $e = f(t) - f^*(t)$. Fazendo uma mudança de coordenadas:

$$\upsilon = -\frac{\beta_0}{\beta_4} f - \frac{\beta_1}{\beta_4} \dot{f} - \frac{\beta_2}{\beta_4} \ddot{f} - \frac{\beta_3}{\beta_4} \dddot{f} + \frac{1}{\beta_4} u \tag{4.11}$$

Tomando-se a Equação 4.9, temos que $\upsilon = f^{(4)}$. Com essa informação, podemos configurar um controlador com realimentação *tracking* (Nota: verificar esse conceito!) com a expressão que se segue

$$\upsilon = f^{*(4)} - k_4 \dddot{e} - k_3 \ddot{e} - k_2 \dot{e} - k_1 e - k_0 \int_0^t e dt \tag{4.12}$$

Os valores dos ganhos k_i devem ser tais que o polinômio característico em malha fechada $s^5 + k_4 s^4 + k_3 s^3 + k_2 s^2 + k_1 s + k_0$ seja estável (ver Fabrício et al.). Portanto, o erro e convergirá exponencialmente para 0, e $f(t)$, além de suas derivadas até a 4ª ordem, convergirão para suas trajetórias de referência: $f^*(t), ..., f^{*(4)}(t)$. Na Equação 4.12, o termo $k_0 \int_0^t e dt$ é introduzido de forma a se corrigir erros estáticos causados por perturbações externas.

A expressão final para a entrada original u, portanto, fica

$$u = \beta_0 f + \beta_1 \dot{f} + \beta_1 \ddot{f} + \beta_1 \ddot{\ddot{f}} + \beta_4 \upsilon \tag{4.13}$$

Em que υ é dado pela Equação 4.11.