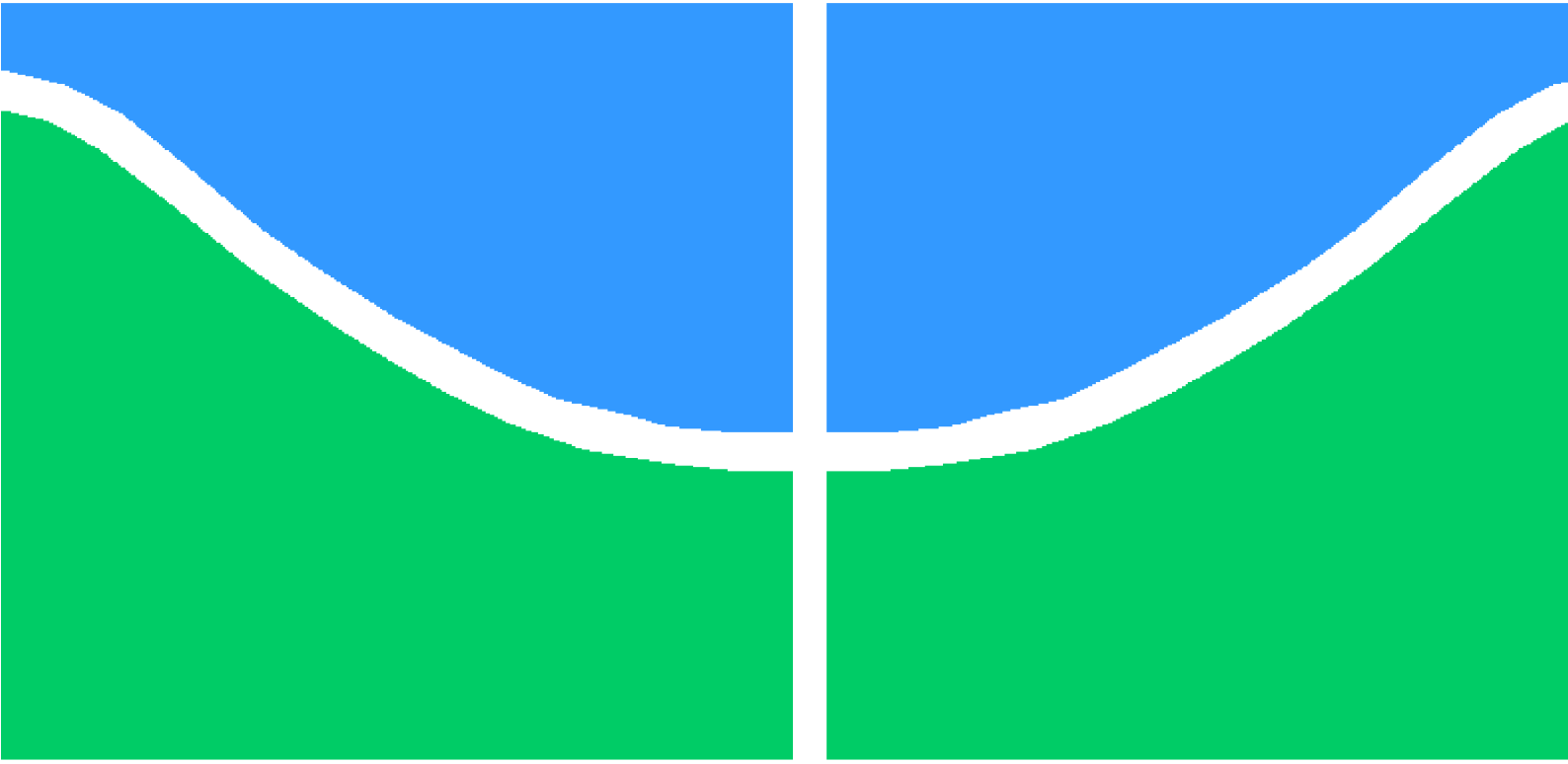

PROJETO DE GRADUAÇÃO

Comparação de Técnicas de Redução de Modelo

Por,

Carlos Lima Santoro

Brasília, 5 de julho de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO

Comparação de Técnicas de Redução de Modelo

Por,

Carlos Lima Santoro

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Mecânico

Banca Examinadora

Prof. Marcus Vinícius Girão de Morais,
UnB/ ENM (Orientador) _____

Profa. Marcela Rodrigues Machado_____

Prof. Adriano Todorovic Fabro _____

Brasília 5 de julho de 2018

Resumo

Em análise dinâmica de estruturas, a modelagem numérica com o uso do método de elementos finitos (MEF) se tornou popular devido a sua capacidade de produzir resultados satisfatórios em domínios de geometria complexa. Na busca de precisão dos resultados produzidos pelo MEF, modelos com muitos graus de liberdade e matrizes de elevada ordem são facilmente produzidos. No entanto, a manipulação de tais matrizes se torna dispendiosa e muitas vezes inviável. Para contornar tal problema surgem os métodos de redução de modelo, que tentam condensar sistemas grandes descaracterizando-os o mínimo possível. Este trabalho se dedica a elaborar e implementar um algoritmo que parte de um modelo inicial de n graus de liberdade, reduzindo-o por diferentes métodos e comparando o desempenho de cada método no cálculo de funções de resposta em frequência. Os métodos de redução comparados são a condensação estática (ou de Guyan), Improved Reduction System (IRS) e System Equivalent Reduction-Expansion Process (SEREP). O sistema utilizado nas reduções é uma turbina eólica, onde foi considerado apenas a vibração torcional não amortecida do trem de força, em que os nós do rotor e do gerador são os graus de liberdade mestres e os nós dos eixos e da caixa multiplicadora são graus de liberdade escravos. Os três métodos apresentaram boa precisão, porém o mais preciso foi o SEREP, tanto na amplitude quanto na frequência natural. O método que consumiu menor tempo de CPU foi o de Guyan, mas o IRS apresentou a melhor relação entre precisão e tempo de CPU.

Abstract

In structural dynamics analysis, the numerical modeling with finite element method (MEF) became popular because it capacity of yeld good results in domains with complex geometry. To get more accuracy in the results yelds by MEF, models with many degrees of freedom and large matrices are easily generated. But, the manipulation of this matrices is costly and sometimes impracticable. To solve this problem, arise the reduction methods, capable to reduce large systems with minimum information loss. This work is an elaboration of an algorithm capable to discretize an inicial model, reduce it by difrent methods comparing the performance of each reduction on the in the calculation of frequency response function. The system used in the reductions is a wind turbine, where only the undamped torsional vibration of the power train was considered, where the rotor and generator nodes are master degrees of freedom and the shafts and the gear box nodes are slaves degrees of freedom. The three methods presented good accuracy, but the most accurate was SEREP, both in amplitude and natural frequency. The method that consumed the least CPU time was Guyan, but the IRS presented the best relation between precision and CPU time.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Símbolos Latinos

J	Momento de Inércia	$[kg \cdot m^2]$
k_t	Rigidez	$[N/m]$
A	Amplitude	$[m]$
N	Número de Dentes	
n	Número de Graus de Liberdade	
E_c	Energia Cinética	$[J]$
$\det(w)$	Determinante Característico	
f_n	Frequência Natural	$[Hz]$
M	Matriz de Inércia	$[kgm^2]$
C	Matriz de Amortecimento	$[kgm^2/s]$
K	Matriz de Rigidez	$[Nm/rad]$
T	Torque	$[Nm]$
W	Base de Transformação	
F	Vetor de Forçamento	$[N]$
I	Matriz Identidade	
E	Energia Cinética	$[J]$
V	Energia Potencial Elástica	$[J]$
u	Vetor Normalizado de Amplitudes	
v	Componente de u_r Sobre u	
r	Coordenada Modal	
G	Módulo de Cisalhamento	$[Pa]$
L	Comprimento	$[m]$
$nepe$	Número de Elementos por Eixo	
ne	Número Total de Elementos do Sistema	
nn	Número Total de Nós do Sistema	

Símbolos Gregos

θ	Deslocamento Angular	$[rad]$
$\dot{\theta}$	Velocidade Angular	$[rad/s]$
$\ddot{\theta}$	Aceleração Angular	$[rad/s^2]$
ψ	Angulo de Fase	$[rad]$
ω	Frequência	$[rad/s]$
ω_n	Frequência Natural	$[rad/s]$
ξ	Coeficiente de Amortecimento	
γ	Desvio Relativo de Amplitude	
ϵ	Desvio Relativo de Frequência	
λ	Frequência de Ressonância ao Quadrado	$[rad^2/s^2]$
Ξ	Matriz Coeficiente de Amortecimento	
ρ	Massa Específica	$[kg/m^30]$
Φ	Modo de Vibração	

Sumário

1 INTRODUÇÃO

A análise dinâmica de estruturas, no intuito de atender aos objetivos de um projeto de engenharia, deve cumprir uma série de requisitos como: produzir resultados suficientemente precisos e seguros, necessitar de baixo custo e possibilitar uma execução em tempo hábil. Com o avanço e o surgimento de tecnologias complementares, como os computadores, a análise dinâmica é aprimorada a cada ano, o que não cessa a busca por melhor atender a estes requisitos.

Antes do surgimento e consolidação da mecânica computacional, a análise dinâmica contemplava apenas soluções analíticas de equações diferenciais para prever e entender o comportamento de estruturas. É claro que o entendimento analítico não perde sua importância, uma vez que constitui a base de qualquer método de análise. No entanto, a geometria das diversas estruturas, como um tambor de lavadora, uma ponte ou a pá de uma turbina, se tornaram complexas ao longo dos anos. Isso torna difícil, e em muitos casos inviável, o uso de soluções analíticas, uma vez que tais soluções são aplicáveis à sistemas de geometria simples, na prática.

Na tentativa de viabilizar o uso de soluções puramente analíticas, os projetistas recorriam à sucessivas simplificações dos sistemas analisados. Porém, em alguns casos essas simplificações comprometiam a precisam do resultado alcançado, limitando a capacidade de tecnológica dos produtos de engenharia. Neste cenário, os métodos numéricos se desenvolveram e ganharam notável importância quando puderam ser aplicados através dos computadores.

É neste ambiente que surge o método de elementos finitos (MEF), um método que permite contornar a dificuldade de descrever geometrias complexas com equações analíticas. Tal método consiste basicamente na decomposição das estruturas analisadas em diversas partes ou subestruturas. Ou seja, consiste na discretização dos domínios nos quais são aplicas as equações diferenciais. Então tais equações são transformadas para sua forma discreta e resolvidas através de métodos numéricos (KWON; BANG, 1997). O MEF pode ser utilizado para determinação de modos de vibração, de estados de tensão, de taxas de transferência de calor, de campos de velocidade em escoamento fluido e muitos outros parâmetros relevantes à engenharia como um todo.

As subestruturas produzidas no MEF são denominadas elementos, os quais são ligados entre si pelos nós. Em análise dinâmica de sistemas, os nós podem possuir de 1

a 6 graus de liberdade, sendo 3 de translação e 3 de rotação. Em certa medida, quanto maior o número de elementos, maior a precisão dos resultados produzidos pelo MEF. No entanto, realizar discretização de domínio com elevado número de elementos, resulta em matrizes de elevada ordem para descrever o modelo espacial do sistema. A manipulação de tais matrizes nos cálculos realizados se torna lenta e em alguns casos inviável a depender da ordem dessas matrizes.

Neste sentido, surge a importância da utilização dos métodos de redução de modelo, largamente utilizados pela comunidade de análise dinâmica de estruturas. Estes métodos são aplicado em trabalhos de otimização dinâmica, problemas de autovalores e autovetores, flambagem estrutural, sensitividade e definição de parâmetros de controle, atualização de modelo e identificação de danos (QU, 2013).

As técnicas de redução de modelo possuem a capacidade de reduzir a ordem das matrizes que descrevem um sistema com um mínimo de perda de informação, produzindo novas matrizes menores que as primeiras. Assim, os cálculos de análise dinâmica podem ser aplicados à matrizes menores diminuindo o custo computacional do trabalho. Muitas vezes não é apenas uma questão de aumentar a performance da análise, mas sim uma questão de viabilização da mesma.

Em atualização de modelo, por exemplo, os parâmetros de um modelo numérico produzidos com MEF são comparados com os parâmetros medidos em um ensaio experimental. Entretanto, não é conveniente instrumentar uma estrutura com o número de acelerômetros igual ao número de nós de um modelo computacional. Assim, um método de redução de modelo pode viabilizar a comparação de um modelo experimental com um modelo numérico.

1.1 Obejtivo e Metodologia

O objetivo deste trabalho é construir e implementar um algoritmo capaz de partir de um sistema discretizado de n graus de liberdade, aplicar diferentes métodos de redução sobre este sistema e comparar os diferentes modelos reduzidos com o primeiro, no que concerne à precisão e velocidade no cálculo de funções de resposta em frequência. O sistema discretizado utilizado nas análises é uma turbina eólica. O modelo que o descreve considera apenas a vibração torcional não amortecida de seu trem de força.

Para tal, o algoritmo será validado através da comparação da resposta de um modelo analítico com a resposta do modelo produzido inicialmente pelo algoritmo. Posteriormente, os diferentes métodos de redução serão comparados quanto à sua precisão e tempo de execução. A sobreposição das respostas no domínio da frequência e uma métrica proposta para análise de erro serão utilizadas para avaliar a precisão dos métodos. Tal precisão é avaliada quanto à amplitude de vibração e frequência de ressonância.

1.2 Estrutura do Relatório

- **Capítulo 1:** Neste capítulo há a introdução ao tema do trabalho, enquadramento deste no contexto de engenharia e apresentação do objetivo.
- **Capítulo 2:** Neste capítulo há um sumário da fundamentação teórica de dinâmica de estruturas, a origem do modelo escolhido, a formulação dos métodos de redução de modelo e da métrica de análise de erro.
- **Capítulo 3:** Neste capítulo há um detalhamento da implementação do algoritmo em Matlab, trazendo as escolhas realizadas para introduzir os conceitos apresentados no capítulo 2 em uma linguagem computacional.
- **Capítulo 4:** Neste capítulo são apresentados os resultados da validação do código e da performance dos métodos de redução de modelo diante da variação dos parâmetros inseridos no algoritmo.
- **Capítulo 5:** Neste capítulo é efetuada a conclusão, onde a análise dos resultados são sumarizadas e sugestões para trabalhos futuros são apresentadas.
- **Apêndice:** Esta seção contém os códigos integrais utilizados neste trabalho.

2 Formulação Teórica

Neste capítulo são apresentados alguns conceitos teóricos a fim de embasar o que é realizado computacionalmente com o código implementado. Aqui, é apresentado o modelo inicial escolhido para a realização das comparações, validação e análise. Há também a formulação teórica da métrica de análise de erro, que mais tarde é utilizada na comparação dos resultados produzidos pelas diferentes técnicas de redução de modelo.

2.1 Características Dinâmicas de um Sistema

Um sistema dinâmico pode ser caracterizado de três maneiras: modelo espacial, modelo modal e modelo de resposta (EWINS, 2000). No modelo espacial, o sistema é composto por um matriz de massa M, uma matriz de rigidez R e uma matriz de amortecimento C. Em análise de dinâmica de estruturas, a equação do movimento geralmente é escrita como em 2.1.

$$M\ddot{X}(t) + C\dot{X}(t) + KX(t) = F(t) \tag{2.1}$$

onde M, K e $C \in R^{n\times n}$; $\ddot{X}$, $\dot{X}$ e $X \in R^n$ são os vetores de aceleração, velocidade e deslocamento; $F \in R^n$ é o vetor de forçamento externo; n denota o número de graus de liberdade e t o tempo (QU, 2013). Neste trabalho, o sistema selecionado constitui um problema de vibração torcional pura, então a partir daqui, é utilizada a versão torcional da equação2.1 apresentada na equação 2.2. O modelo escolhido também despreza os efeitos de qualquer tipo de amortecimento, então o termo $C\dot{X}$ é desprezado.

$$M\ddot{\theta}(t) + K\theta(t) = T(t) \tag{2.2}$$

$\ddot{\theta}$ e θ são a aceleração e o deslocamento angular, enquanto T é o vetor de torques externos aplicados. Já o modelo modal pode ser conhecido através da matriz de frequências naturais ω_n^2 e dos vetores de modos de vibração Φ. Se assumida solução harmônica para a equação 2.2, pode-se deduzir:

$$\begin{aligned}\theta &= \theta_0 e^{i\omega t} \\ \dot{\theta} &= i\omega\theta_0 e^{i\omega t} \\ \ddot{\theta} &= -\omega^2\theta_0 e^{i\omega t}\end{aligned} \tag{2.3}$$

onde $i = \sqrt{-1}$. Substituindo as equações 2.3 em 2.2 e considerando vibração livre ($T(t) = 0$), produz-se a equação 2.4.

$$(K - \omega_n^2 M)\theta_0 = 0 \tag{2.4}$$

Tal forma constitui um problema de autovalores e autovetores, do qual podem ser extraídos os vetores Φ e a matriz ω_n^2. Assim, a matriz ω_n^2 terá a forma 2.5.

$$\omega_n = \begin{bmatrix} \omega_1^2 & 0 & \cdots & 0 \\ 0 & \omega_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_n^2 \end{bmatrix}_{n\times n} \tag{2.5}$$

e os vetores Φ possuirão a forma 2.6.

$$\theta_0 = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_n \end{bmatrix}_{n\times n} \tag{2.6}$$

Para conhecer o modelo resposta, é necessário conhecer a amplitude do sistema no domínio da frequência, ou seja, sua função resposta em frequência (FRF). Neste trabalho, esta função é determinada numericamente. Para tal, é necessário seguir um procedimento análogo ao de obtenção da equação 2.4, mas agora considerando um forçamento harmônico, como na equação 2.7.

$$T = T_0 \sin \omega t \tag{2.7}$$

Assim, pode-se reescrever a equação 2.4 como em 2.8.

$$(K - \omega^2 M)\theta_0 = T_0 \tag{2.8}$$

Agora, basta resolver a equação 2.8 para θ_0, produzindo a equação 2.9

$$\theta_0 = (K - \omega^2 M)^{-1} T_0 \tag{2.9}$$

Com esta equação, a amplitude de vibração pode ser calculada para cada frequência ω na determinação da FRF.

2.2 Modelo Inicial

Para realizar a discretização e os testes com os métodos de redução de modelo, foi escolhido um modelo apresentado por (FRISWELL; LEES; LITAK, 2011). Considere o sistema rotor, transmissão, gerador de uma turbina conectados entre si por dois eixos. Tal sistema pode ser aproximado pelo modelo apresentado na Figura 1. Esse modelo esquemático trata o rotor como um disco de inércia J_1, a transmissão como dois discos de inércia J_2 e J_3, os quais representam um par engrenado; e o gerador como um disco de inércia J_4. Tais elementos possuem deslocamentos angulares representados por θ_1, θ_2, θ_3 e θ_4, respectivamente. Os eixos são tratados como molas torcionais sem massa de constantes elásticas k_1 e k_2.

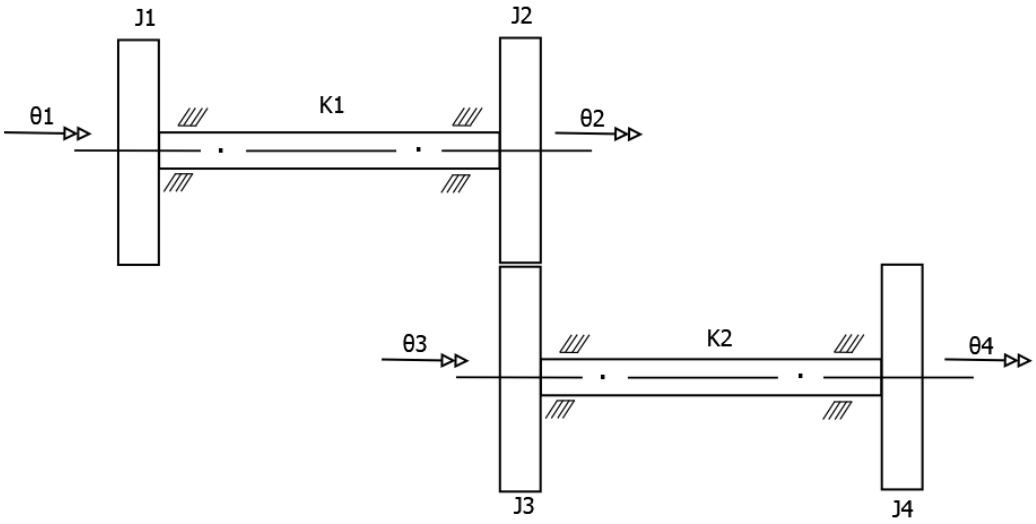

Figura 1 – Modelo completo.

A Figura 2 contém o diagrama de corpo livre do modelo. Utilizando a abordagem Newtoniana e aplicando o somatório de força sobre os corpos do sistema, deduz-se as equações do movimento 2.10.

$$\begin{aligned} J_1\ddot{\theta}_1 + k_1(\theta_1 - \theta_2) &= F_1 \\ J_2\ \ddot{\theta}_2 + k_1(\theta_2 - \theta_1) &= F_2 + r_2F_{23} \\ J_3\ddot{\theta}_3 + k_2(\theta_3 - \theta_4) &= F_3 + r_3F_{23} \\ J_4\ddot{\theta}_4 + k_2(\theta_4 - \theta_3) &= F_4 \end{aligned} \tag{2.10}$$

onde a relação de engrenamento é descrita por $n = N_2/N_3$. Admitindo-se mesmo deslocamento tangencial entre as engrenagens, tem-se:

$$\begin{aligned} r_2\theta_2 &= -r_3\theta_3 \\ \theta_3 &= -n\theta_2 \end{aligned} \tag{2.11}$$

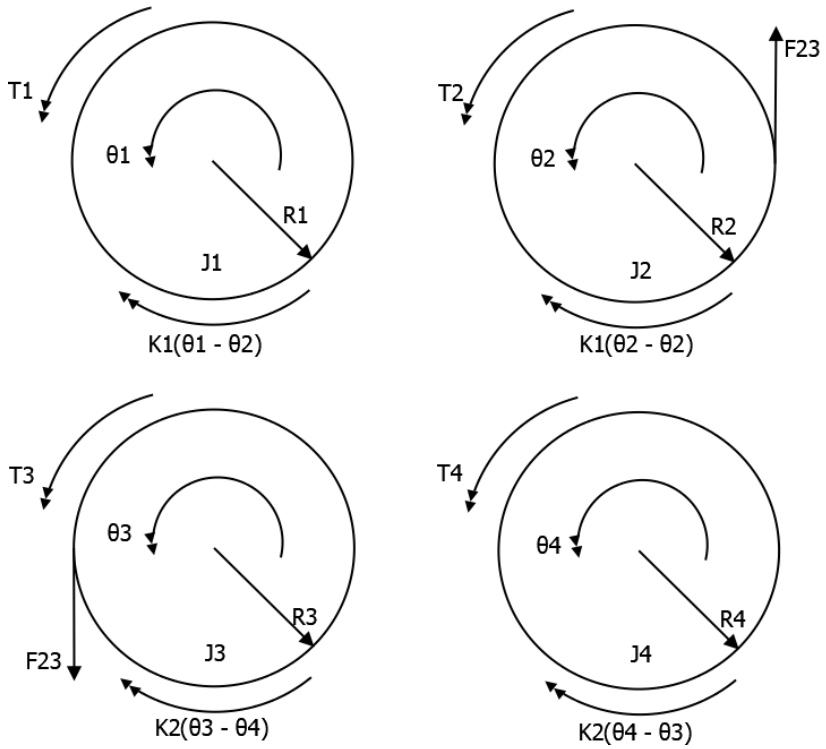

Figura 2 – Diagrama de corpo livre do modelo completo.

Logo, substituindo a equação 2.11 na 2.10, temos:

$$
\begin{aligned}
&J_1\ddot{\theta}_1 + k_1(\theta_1 - \theta_2) = F_1 \\
&J_2\ \ddot{\theta}_2 + k_1(\theta_2 - \theta_1) = F_2 + r_2F_{23} \\
&-J_3\ddot{\theta}_2 n - k_2(n\theta_2 + \theta_4) = F_3 + r_3F_{23} \\
&J_4\ddot{\theta}_4 + k_2(\theta_4 - \theta_3) = F_4 \qquad (2.12) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2.13)
\end{aligned}
$$

Multiplicando a equação 2.13.3 por n e subtraindo a equação 2.13.3 da 2.13.2, temos:

$$(J_2 + n^2J_3)\ddot{\theta}_2 - k_1\theta_1 + (k_1 + n^2k_2)\theta_2 + nk_2\theta_4 = F_2 - nF_3 \qquad (2.14)$$

Isso é possível graças ao fato de que os discos 2 e 3 estão engrenados e, portanto, seu movimento não é independente, podendo ser descrito por apenas um grau de liberdade. Desta maneira a inércia da caixa de engrenagem é representada por apenas um termo, então este modelo é conhecido como modelo de três massas. Então, podemos incluir a expressão 2.14 na 2.13 para obter um sistema em forma matricial:

$$
\begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 + n^2J_3 & 0 \\ 0 & 0 & J_4 \end{bmatrix}
\begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_4 \end{Bmatrix}
+
\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + n^2k_2 & nk_2 \\ 0 & nk_2 & k_2 \end{bmatrix}
\begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_4 \end{Bmatrix}
=
\begin{Bmatrix} F_1 \\ F_2 - nF_3 \\ F_4 \end{Bmatrix} \qquad (2.15)
$$

2.3 Técnicas de Redução de Modelo

Na análise dinâmica de sistemas mecânicos, a equação do movimento é resolvida para as coordenadas generalizadas de deslocamento do sistema $\theta^T = [\theta_1\ \theta_2\ ...\ \theta_n]$, para n graus de liberdade. Na modelagem e simulação de sistemas dinâmicos grandes e complexos, é desejável o desenvolvimento de modelos de subsistemas que são partições do sistema inicial, a fim de resolvê-lo para um número reduzido de graus de liberdade θ_r, onde $\theta = W\theta_r$ sendo W a matriz de transformação de um métodoe de redução de modelo (SELLGREN, 2003).

Assim, ao invés de resolver a equação $M\ddot{\theta} + C\dot{\theta} + K\theta = F$, resolve-se:

$$M_r\ddot{\theta}_r + C_r\dot{\theta}_r + K_r\theta_r = F_r \tag{2.16}$$

onde

$$\begin{aligned} M_r &= W^T M W \\ C_r &= W^T C W \\ K_r &= W^T K W \\ F_r &= W^T F \end{aligned} \tag{2.17}$$

Existem várias técnicas para obter tal base de redução. Esta seção discorre sobre Condensação Estática, IRS (*Improved Reduced System*) e SEREP (*System Equivalent Reduction Expansion Process*).

2.3.1 Condensação Estática

A condensação estática, ou de Guyan, pode reduzir o número de graus de liberdade do sistema sem a necessidade assumir novas suposições físicas. Para tal, o sistema deve ser organizado de maneira que sua equação do movimento possa ser escrita como na equação 2.18, onde as matrizes são particionadas em submatrizes mestres (subíndice m) e escravas (subíndice s).

$$\begin{bmatrix} M_{mm} & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_m \\ \ddot{\theta}_s \end{Bmatrix} + \begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \begin{Bmatrix} \theta_m \\ \theta_s \end{Bmatrix} = \begin{Bmatrix} F_m \\ F_s \end{Bmatrix} \tag{2.18}$$

Vale ressaltar que $K_{ms} = K_{sm}^{-1}$. A condensação estática ignora as forças de inércia no sistema e leva em conta apenas a parcela estática do problema. Então, para calcular sua matriz de transformação W_{Guyan}, considere o problema estático $K\theta = F$:

$$\begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \begin{Bmatrix} \theta_m \\ \theta_s \end{Bmatrix} = \begin{Bmatrix} F_m \\ F_s \end{Bmatrix} \tag{2.19}$$

onde θ_m são os graus de liberdade mestre (a serem mantidos) e θ_s são os graus de liberdade escravos (a serem removidos).

Isso é igual a:

$$K_{mm}\theta_m + K_{ms}\theta_s = F_m \tag{2.20}$$

$$k_{sm}\theta_m + K_{ss}\theta_s = F_s \tag{2.21}$$

Se reformularmos 2.21, obtemos para os graus de liberdade escravos:

$$\theta_s = K_{ss}^{-1}F_s - K_{ss}^{-1}K_{sm}\theta_m \tag{2.22}$$

Substituindo 2.22 em 2.20, obtemos o problema estático reduzido para os graus de liberdade mestres:

$$[K_{mm} - K_{sm}K_{ss}^{-1}K_{sm}]\theta_m = F_m - K_{ms}K_{ss}^{-1}F_s \tag{2.23}$$

que também pode ser escrito como $K_r u_r = F_r$. Assim, a base redutora pode ser escrita como:

$$W = \begin{bmatrix} I \\ -K_{ss}^{-1}K_{sm} \end{bmatrix} \tag{2.24}$$

As energias de um sistema dinâmico podem ser escritas como $E = \frac{1}{2}\dot{\theta}' M \dot{\theta}$ e $V = \frac{1}{2}\theta' K \theta$. A aplicação da base de transformação W sobre essas equações resulta em:

$$E = \frac{1}{2}\dot{\theta}_m' W' M W \dot{\theta}_m \tag{2.25}$$

$$V = \frac{1}{2}\theta_m' W' K W \theta_m \tag{2.26}$$

Então, a matriz de massa reduzida é $M_r = W'MW$ e a matriz de regidez reduzida é $K = W'KW$. Sendo a matriz de massa

$$M_r = \begin{bmatrix} M_m & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix} \tag{2.27}$$

a matriz de massa reduzida se torna

$$M_r = M_m - M_{ms}K_{ss}^{-1}K_{sm} - (K_{ss}^{-1}K_{sm})'(M_{sm} - M_{ss}K_{ss}^{-1}K_{sm}) \tag{2.28}$$

Com a condensação estática, a rigidez do sistema é conservada na medida em que os deslocamentos relativos entre os graus de liberdade mestres permanecem iguais

se deslocados estaticamente. Vale ressaltar que, se escolhidos adequadamente, os graus de liberdade escravos possuem forças de inércia desprezíveis. Principalmente em baixa frequência. Tal fato explica a nomenclatura "mestre"e "escravo", já que o movimento destes será governado pelas forças impostas por aqueles.

No caso da matriz de massa reduzida, nada da complexidade do sistema é perdida desde que todos os elementos da matriz de rigidez contribuam. Contudo, na matriz de massa reduzida, aparecem combinações dos termos de massa com os termos de rigidez. O resultado do problema de autovalores e autovetores é aproximadamente, mas não exatamente preservado (GUYAN, 1965).

2.3.2 Improved Reduction System

O'Callahan (1989) introduziu a técnica Redução de Sistema Melhorada, ou *Improved Reduction System* (IRS) como ela é mais conhecida. Este método consiste em um aperfeiçoamento da condensação estática (FRISWELL; MOTTERSHEAD, 1995). Obviamente, é impossível reproduzir o comportamento exato do sistema a partir de um sistema reduzido. Em algum momento, a acurácia será prejudicada em virtude da velocidade ou da possibilidade de validação experimental do modelo numérico.

Para conseguir a matriz de transformação W_{IRS}, é necessário partir da condensação estática. Na tentativa de minimizar a perda de precisão da condensação estática devido ao negligenciamento das forças de inércia, o IRS adiciona termos de inércia como forças pseudo-estáticas à formulação de Guyan (FRISWELL; GARVEY; PENNY, 1995). Então, partindo do sistema organizado como na equação 2.18 e assumindo solução harmônica, pode-se reescrever a parte inferior da equação.

$$[K_{ss} - \omega^2 M_{ss}]\theta_s = -[K_{sm} - \omega^2 M_{sm}]\theta_m \tag{2.29}$$

Reorganizando 2.29 e usando expansão de séries binomiais, tem-se

$$\begin{aligned}
\theta_s &= -[K_{ss} - \omega^2 M_{ss}]^{-1}[K_{sm} - \omega^2 M_{sm}]\theta_m \\
\theta_s &= -K_{ss}^{-1}[q - \omega^2 M_{ss}K_{ss}^{-1}]^{-1}[K_{sm} - \omega^2 M_{sm}]\theta_m \\
\theta_s &= -K_{ss}^{-1}[q + \omega^2 M_{ss}K_{ss}^{-1} + o(\omega^4)][K_{sm} - \omega^2 M_{sm}]\theta_m \\
\theta_s &= -K_{ss}^{-1}[K_{sm} + \omega^2(M_{ss}K_{ss}^{-1}K_{sm} - M_{sm}) + o(\omega^4)]\theta_m
\end{aligned} \tag{2.30}$$

onde $o(\omega^4)$ representa um erro de ordem ω^4. Como o objetivo é melhorar a resposta do sistema, estimada pelo modelo reduzido baseado na condensação estática, então a primeira ordem em ω^2 satisfaz a equação

$$\omega^2 M_R \theta_m = K_R \theta_m \tag{2.31}$$

onde M_R e K_R são as matrizes de massa e rigidez obtidas com a condensação estática, e ω e θ_m são uma frequência natural e um autovetor associado às coordenadas mestres, respectivamente. Ao ignorar os termos em ω^4 e em mais alta potência, a equação 2.30 se torna

$$\theta_s = [-K_{ss}^{-1}K_{sm} + K_{ss}^{-1}(M_{sm} - M_{ss}K_{ss}^{-}1K_{sm})M_R^{-1}K_R]\theta_m \tag{2.32}$$

A equação 2.32 define as coordenadas escravas a partir das coordenadas mestres. Assim, a matriz de transformação W_{IRS} pode ser convenientemente escrita da seguinte forma:

$$W_{IRS} = W_{Guyan} + SMW_{Guyan}M_R^{-1}K_R \tag{2.33}$$

onde

$$S = \begin{bmatrix} 0 & 0 \\ 0 & K_{ss}^{-1} \end{bmatrix} \tag{2.34}$$

Assim, as matrizes de massa e rigidez do sistema reduzido pela técnica IRS podem ser obtidas fazendo

$$M_{IRS} = W_{IRS}^T \; M \; W_{IRS}$$
$$K_{IRS} = W_{IRS}^T \; K \; W_{IRS} \tag{2.35}$$
$$\tag{2.36}$$

2.3.3 System Equivalent Reduction-Expansion Process

O Processo de Redução-Expansão de Sistema Equivalente, mais conhecido como System Equivalent Reduction-Expansion Process (SEREP), também foi proposto por O'Callahan (1989). Ao contrário das técnicas citadas nas seções anteriores, o SEREP não parte da abordagem de Guyan. Sua matriz de transformação W_{SEREP} é obtida a partir dos autovetores calculados com o sistema original.

A partir dos graus de liberdade mestres, já escolhidos, e do sistema organizado como na equação 2.18, considere a solução para os autovetores do sistema $\Phi = [\Phi_{am}\Phi_{as}]$, onde m e s denotam mestres e escravos, e a significa que todas as coordenadas foram mantidas no vetor. Os modos Φ_{am} formam uma matriz $n \times m$ onde n é o número de graus de liberdade e m é o número de graus de liberdade mestres. Já os modos Φ_{as} formam uma matriz $n \times (n - m)$ (ERNANDEZ, 2006). Considerando:

$$\theta = \begin{Bmatrix} \theta_m \\ \theta_s \end{Bmatrix} = \Phi\mathbf{r} = [\Phi_{am}\Phi_{as}] \begin{Bmatrix} r_m \\ r_s \end{Bmatrix} = \begin{bmatrix} \Phi_{mm} & \Phi_{ms} \\ \Phi_{sm} & \Phi_{ss} \end{bmatrix} \begin{Bmatrix} r_m \\ r_s \end{Bmatrix} \tag{2.37}$$

onde r é a coordenada modal, e multiplicando a equação 2.18 por

$$\Phi^T = [\Phi_{am}\Phi_{as}]^T \tag{2.38}$$

produz-se a equação do movimento em coordenadas modais

$$I\ddot{r} + \lambda r = \Phi^T F \tag{2.39}$$

onde a matriz de massa transformada é

$$I = \begin{bmatrix}\Phi_{am}^T \\ \Phi_{as}^T\end{bmatrix}\begin{bmatrix}M_{mm} & M_{ms} \\ M_{sm} & M_{ss}\end{bmatrix}\begin{bmatrix}\Phi_{am} & \Phi_{as}\end{bmatrix} = \begin{bmatrix}\Phi_{am}^T M\Phi_{am} & \Phi_{am}^T M\Phi_{as} \\ \Phi_{as}^T M\Phi_{am} & \Phi_{as}^T M\Phi_{as}\end{bmatrix} = \begin{bmatrix}I_m & 0 \\ 0 & I_s\end{bmatrix} \tag{2.40}$$

e a matriz de rigidez transformada é

$$\Lambda = \begin{bmatrix}\Phi_{am}^T \\ \Phi_{as}^T\end{bmatrix}\begin{bmatrix}K_{mm} & K_{ms} \\ K_{sm} & K_{ss}\end{bmatrix}\begin{bmatrix}\Phi_{am} & \Phi_{as}\end{bmatrix} = \begin{bmatrix}\Phi_{am}^T K\Phi_{am} & \Phi_{am}^T K\Phi_{as} \\ \Phi_{as}^T K\Phi_{am} & \Phi_{as}^T K\Phi_{as}\end{bmatrix} = \begin{bmatrix}\Lambda_m & 0 \\ 0 & \Lambda_s\end{bmatrix} \tag{2.41}$$

Em seguida, elimina-se o vetor modal $r = [r_m r_s]^T$ assumindo $r_s = 0$. Assim, a equação 2.39 é reduzida a

$$I_m\ddot{r}_m + \lambda_m r_m = \Phi_{am}^T F_m \tag{2.42}$$

A partir de agora, segundo a equação 2.37, é possível substituir $r_m = \Phi_{rr}^{-1}\theta_m$ na equação 2.42, produzindo

$${\Phi^{-1}}_{mm}^T \Phi_{am}^T M\Phi_{am}\Phi_{mm}^{-1}\ddot{x}_m + {\Phi^{-1}}_{mm}^T \Phi_{am}^T K\Phi_{am}\Phi_{mm}^{-1}x_m = {\Phi^{-1}}_{mm}^T \Phi_{am}^T F \tag{2.43}$$

Então, pode-se definir a matriz de transformação W_{SEREP} como

$$W_{SEREP} = \Phi_{am}\Phi_{mm}^{-1} = \begin{bmatrix}\Phi_{mm} \\ \Phi_{sm}\end{bmatrix}\Phi_{mm}^{-1} = \begin{bmatrix}I \\ \Phi_{sm}\Phi_{mm}^{-1}\end{bmatrix} \tag{2.44}$$

2.4 Métrica de Análise de Erro

Negligenciar alguns graus de liberdade induz algum erro no comportamento previsto para o sistema. Através da transformação T, onde $\theta_{0,r}^T = W\theta_{0,r}$ e $(K_r - \lambda_r M_r)\theta_{0,r} = 0$, é possível recuperar um sistema de mesmo número de graus de liberdade de antes da aplicação da redução de modelo.

O erro relativo γ_i do modo de vibrar e o erro relativo ϵ_i da frequência de ressonância, na frequência correspondente à posição i dentro do vetor de frequências, são calculados como (SELLGREN, 2003):

$$\gamma_i = 1 - \left| \frac{\theta_{0r,i}^T \theta_{0i}}{\left|\theta_{0r,i}^T\right| \left|\theta_{0i}\right|} \right| \tag{2.45}$$

$$\epsilon_i = \frac{\lambda_{r,i} - \lambda_i}{\lambda_{r,i}} \tag{2.46}$$

onde o subindice r denota valores do modelo reduzido pela condensação.

A equação 2.45 analisa os erros de amplitude do sistema como um todo. O vetor $\theta_{0r,i}$ quando dividido por sua norma, resulta em um vetor unitário de mesma direção e sentido u_r, onde cada componente corresponde a um grau de liberdade do sistema. De maneira análoga, o vetor u é formado a partir do vetor θ_{0i}.

Com os vetores normalizados, a única diferença entre eles será a direção. Tal desvio de direção é produzido pelos graus de liberdade escravos recuperados do sistema reduzido. Então, multiplicar os vetores u_r e u resulta em um valor entre 0 e 1 que representa a componente de u_r sobre u, chamada de v. Assim, $\gamma = 1 - v$ corresponde à diferença relativa entre os graus de liberdade escravos de $\theta_{0r,i}$ e $\theta_{0,i}$, como sugere a figura 3.

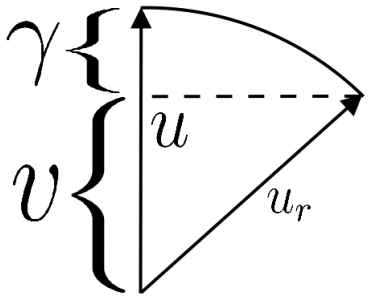

Figura 3 – Esquema de vetores da métrica de erro da amplitude de vibração.

3 Implementação

Este capítulo mostra os detalhes da implementação dos algoritmos deste trabalho. Para comparar as diferentes técnicas de redução de modelo, um algoritmo foi escrito em Matlab. Tal algoritmo gera um modelo de um sistema mecânico, em seguida discretiza algumas partes do sistema em n partes. Após a construção do modelo, este é reduzido através das diferentes técnicas de redução, produzindo subsistemas. Então, a resposta de cada subsistema é calculada numericamente e posteriormente comparada com as respostas dos demais sistemas.

Aqui, o algoritmo é explicado ao mesmo tempo que os trechos do código são mostrados. Para obter uma visualização não divida e completa do código, este encontra-se no apêndice.

3.1 Construção do Modelo Computacional com MEF

Para comparar as diferentes técnicas de redução, selecionamos o sistema de uma turbina eólica proposto por (FRISWELL; LEES; LITAK, 2011), representado na equação 2.15. Tal escolha permitiu validar a resposta calculada com a redução de Guyan, a qual serviu para validação dos resultados produzidos pelos demais métodos de redução. Os dados utilizados por Friswell para construção do modelo de uma turbina eólica são mostrados na expressão 3.1.

$$\begin{gathered} J_1 = 7000\ kgm^2 \qquad J_2 = 4000\ kgm^2 \qquad J_3 = 13000\ kgm^2 \qquad J_4 = 5000\ kgm^2 \\ k_1 = 1,6\ GNm/rad \quad k_2 = 0,29\ GNm/rad \\ c_1 = 2 \cdot 10^{-5} k_1 \qquad c_2 = 2 \cdot 10^{-5} k_2 \\ N_2 = 56 \qquad N_3 = 111 \end{gathered} \tag{3.1}$$

Tais dados são inseridos na memória do computador por simples atribuição, como mostra o código a seguir.

```
N2 = 56;
N3 = 111;   %numero de dentes
n = N2/N3;  %razão de engrenamento
```

```
J1 = 7e3;
J2 = 4e3;
J3 = 13e3;
J4 = 5e3;   %Inércias [kgm^2]

k1 = 1.6e9;
k2 = 0.29e9;    %Rigidez dos eixos [Nm/rad]
```

Os eixos do sistema proposto na equação 2.15 foram considerados apenas molas torcionais, até aqui. Porém, para cumprir os objetivos deste trabalho e construir um modelo de n graus de liberdade, o próximo passo é considerar o eixo como um elemento com massa e rigidez, para em seguida ter seu domínio discretizado em diversos elementos utilizando o método de elementos finitos (MEF).

Então, para dar início a este processo, antes é preciso definir os parâmetros geométricos, como comprimento e diâmetro; e os parâmetros que definem o material do qual o eixo seria fabricado, como massa específica e módulo de cisalhamento. Para os parâmetros do material, considerou-se os do aço, sendo o módulo de cisalhamento $G = 8 \cdot 10^4 MPa$ e a massa específica $\rho = 7860 kg/m^3$.

A rigidez torcional de um eixo é função de seu módulo de cisalhamento e de sua geometria. Mais especificamente, de seu comprimento e de seu diâmetro. Mas, a intensão é preservar as características físicas do sistema proposto por Friswell, então optou-se por definir um comprimento $L = 1\ m$ para os dois eixos e escrever seus diâmetros em função das rigidezes da seguinte maneira:

$$d = \sqrt[4]{\frac{32kL}{G\pi}} \tag{3.2}$$

Assim, os parâmetros dos eixos são adicionados da seguinte forma:

```
%Propriedades gerais do aço
ro = 7860; %Densidade do aço em kg/m^3
G = 80e9; %Módulo de cisalhamento em Pa

l1 = 1; %Comprimento do eixo 1 em metros
l2 = 1; %Comprimento do eixo 2 em metros

%Diâmetros em função de k, l e G
d1 = (32*k1*l1/(G*pi))^(1/4); %Diâmetro do eixo 1 em metros
d2 = (32*k2*l2/(G*pi))^(1/4); %Diâmetro do eixo 2 em metros
```

```
m_eixo1 = ro*l1*pi*d1^2/4; %Massa do eixo 1 em kg
m_eixo2 = ro*l2*pi*d2^2/4; %Massa do eixo 2 em kg

Je1 = m_eixo1*(d1/2)^2/2; %Inércias de giro dos eixos em kg.m^2
Je2 = m_eixo2*(d2/2)^2/2;
```

A partir daqui, haverá a inserção dos parâmetros necessários para a aplicação do MEF sobre um domínio unidimensional. Primeiro é necessário definir em quantos elementos o eixo será dividido, de maneira que cada elemento representa um pequeno eixo em série com os demais. Em seguida, calcula-se o número de nós, partido da ideia de que cada elemento possuirá um nó em cada extremidade, sendo os nós compartilhados quando estiverem entre dois elementos.

Neste trabalho, foi definido que cada eixo é discretizado a partir da mesma quantidade de elementos, uma vez que arbitrariamente ambos os eixos tem o mesmo comprimento. Tais simplificações são feitas aqui pois o interesse é construir um modelo de n graus de liberdade para explorar as técnicas de redução de modelo. Vale destacar que o número de elementos escolhidos nesta parte do código é o que deve ser e foi alterado para fins de observações utilizando a variação do número de graus de liberdade do sistema.

Neste código, o número de elementos por eixo ($nepe$) é atribuído primeiro e número de elementos do sistema completo (ne) é calculado em função do $nepe$, sendo $ne = 2\,nepe$. Em seguida, calcula-se o número de nós (nn), que devido ao tipo do elemento (cilíndrico com um nó em cada extremidade), é dado por $nn = ne + 1$.

Para montar as matrizes globais de massa e rigidez, optou-se por utilizar o elementos do tipo condensado. Assim, cada elemento possuirá sua matriz elementar de massa e rigidez, as quais são expressas por

$$M_e = \frac{\rho j L}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$
$$K_e = \frac{Gj}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.3}$$

onde j é o momento polar de área dado por

$$j = \frac{\pi d^4}{32} \tag{3.4}$$

já que se trata de um eixo cilíndrico. Estas atribuições e cálculos foram implementadas da seguinte maneira:

```
nepe = 6000; %Número de elementos por eixo
ne = 2*nepe; %Número de elementos
nn = ne + 1; %Número de nós
```

```
le1 = l1/nepe; %Comprimento do elemento do eixo 1
le2 = l2/nepe; %Comprimento do elemento do eixo 2

A1 = pi*d1^4/32; %Momento polar de área do eixo 1
A2 = pi*d2^4/32; %Momento polar de área do eixo 2

Ke1 = (G*A1/le1)*[1,-1;-1,1]; %Matriz de rigidez dos elementos do eixo 1
Ke2 = (G*A2/le2)*[1,-1;-1,1]; %Matriz de rigidez dos elementos do eixo 2

Me1 = (ro*A1*le1/6)*[2,1;1,2]; %Matriz de massa dos elementos do eixo 1
Me2 = (ro*A2*le2/6)*[2,1;1,2]; %Matriz de massa dos elementos do eixo 2
```

Após a definição das matrizes elementares, defini-se a matriz de nós, onde cada linha corresponde a um elemento e cada coluna a um dos nós presente em cada elemento. Como neste caso cada elemento possui apenas dois nós, um em cada extremidade, então a matriz de nós possuirá *ne* linhas e duas colunas. Esta matriz é inserida na memória a partir de um *loop* do tipo for, como mostrado a seguir:

```
nos = zeros(ne,2); %Matriz de nós (cada posição da matriz é...
 relativa a um elemento)
for i = 1:ne
nos(i, 1:2) = [i, i+1];
end
```

Depois da definição da matriz de nós, declara-se as matrizes de massa M e rigidez K do sistema para que seja alocado o espaço necessário na memória para recebê-las. Em seguida, essas matrizes são montadas a partir de um *loop for*, onde as matrizes elementares serão introduzidas nas matrizes globais do sistema. Neste *loop* indexamos as matrizes com a matriz de nós. Para cada ciclo do *loop*, há uma linha da matriz de nós. Assim, ao indexar as matrizes M e K, o índice de linha vai do valor da primeira coluna, da matriz de nós, ao valor da segunda coluna. O mesmo deve ser feito para o índice de coluna, o que resultará na matriz M e K sendo percorridas diagonalmente, como na Figura 4.

A construção das matrizes M e K são feitas em 2 loops diferentes, uma vez que metade da matriz será composta pelas matrizes do eixo 1 e a outra metade pelas matrizes do eixo 2. Em seguida, os dados dos nós especiais, ou seja, nós onde se encontram o rotor, a caixa multiplicadora de velocidades e o gerador; são inseridos. A construção destes dois *loops* é mostrada a seguir:

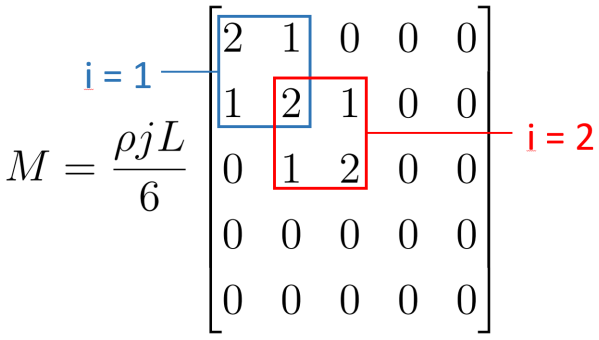

Figura 4 – Esquema de um *Loop* percorrendo a matriz de massa para inserção da matriz elementar.

```
M = zeros(nn,nn);
K = M;
for i = 1:(ne/2)
M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) =...
 M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) + Me1;
K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) =...
 K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) + Ke1;
end
for i = (ne/2 + 1):ne
M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) =...
 M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) + Me2;
K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) =...
 K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) + Ke2;
end
[M, K] = no_especial(nn, M, K, Ke2);
```

A inserção das valores nos nós especiais são feitos através de um função, para que o código fique melhor dividido. Nesta função, os dados de 3.1 são inseridos conforme os acoplamentos das engrenagens observados na equação 2.15. É necessário indexar a posição destes nós em função do número de elementos para que a função possa funcionar para um sistema de qualquer tamanho. Assim, o rotor sempre está no primeiro nó, a caixa multiplicadora no nó $ne/2 + 1$ e o gerador no último nó nn, sendo ne o número de elementos e nn o número de nós. Esta parte do código foi implementada da seguinte forma:

```
function [M, K] = no_especial(nn, M, K, Ke2)
%Esta função adiciona o valor dos nós especiais às matrizes M e K.

%Carrega os dados do sistema
Dados_Friswell
Dados_Eixos
```

```
ne = nn - 1;

%Escreve na matriz
M(1,1) = M(1,1) + J1;

M((ne/2 + 1), (ne/2 + 1)) = M((ne/2 + 1), (ne/2 + 1))*(1 +...
 n^2) + J2 + J3*n^2;
M((ne/2 + 1), (ne/2 + 1) + 1) = M((ne/2 + 1), (ne/2 + 1) + 1)*(-n);
M((ne/2 + 1) + 1, (ne/2 + 1)) = M((ne/2 + 1) + 1, (ne/2 + 1))*(-n);

K((ne/2 + 1), (ne/2 + 1)) = K((ne/2 + 1), (ne/2 + 1)) -...
 Ke2(1,1) + Ke2(1,1)*n^2;
K((ne/2 + 1), (ne/2 + 1) + 1) = K((ne/2 + 1), (ne/2 + 1) + 1)*(-n);
K((ne/2 + 1) + 1, (ne/2 + 1)) = K((ne/2 + 1) + 1, (ne/2 + 1))*(-n);

M(nn, nn) = M(nn, nn) + J4;
end
```

O sistema inicial construído e discretizado com elementos finitos já está pronta. Agora, só é preciso definir quem são os graus de liberdade escravos e atribuir um vetor de forçamento. A escolha dos graus de liberdade escravos podem ser feitas de maneira automática analisando-se a razão entre massa m e rigidez k de cada elemento. Deste modo, onde define-se um parâmetro y onde todos os nós com $m/k < y$ são considerados escravos. Entretanto, neste código, todos os nós entre o primeiro e o último nó, ou seja, entre o rotor e o gerador são considerados escravos de maneira arbitrária. Independente da forma escolhida para selecionar os escravos, um vetor com as posições destes graus de liberdade deve ser atribuído

Já o vetor de forçamento, possui valor 1 na posição do rotor e zero para as demais posições. Dois vetores de forçamento são definidos. Um com o tamanho ajustado ao sistema original de nn graus de liberdade e outro com o tamanho ajustado ao sistema reduzido, que neste caso possui apenas 2 graus de liberdade. Estes vetores foram implementados da seguinte forma:

```
%Definição dos graus de liberdade escravos
SlaveDofs = 2:(nn-1);

%Definição dos vetores de forçamento
F = [1; zeros(nn - 1, 1)];
FR = [1; zeros(nn - length(SlaveDofs) - 1, 1)];
```

3.2 Reduções e Comparação dos Modelos

Após a criação do modelo computacional de n graus de liberdade, as matrizes de massa M e de rigidez K devem passar por algoritmos de redução de modelo capazes de aplicar os conceitos vistos no capítulo 2.3. Cada técnica de redução deve produzir um modelo diferente, que posteriormente devem ter suas respostas calculadas no domínio da frequência. A FRF do modelo inicial também deve ser calculada para que todos possam ser comparados. A comparação se dará graficamente, com a plotagem das respostas calculadas no domínio da frequência; e numericamente, a partir da métrica mostrada na seção 2.4.

As diferentes técnicas de redução foram programadas em forma de função, para que possam receber as matrizes do sistema original, junto com o vetor de posições dos graus de liberdade escravos e retornar as matrizes do sistema reduzido. A escolha por programar em forma de função também facilita a compreensão do código e a medição do tempo de CPU de cada método de redução. Os detalhes da implementação dos algoritmos de redução serão tratados em subseções específicas para cada método.

Para armazenar as matrizes M e K retornadas por cada função de redução de maneira conveniente, optou-se por utilizar um tensor de terceira ordem, onde é armazenada uma matriz reduzida em cada posição. Após o cálculo de todos os modelos, inicia-se os cálculos das FRFs. Para tal, é definido um intervalo de frequência no qual deseja-se observar as amplitudes de vibração. Além do intervalo, também é preciso definir a diferença de frequência em cada ponto da curva. Tais parâmetros são atribuídos e passados para a função que calcula a FRF. A primeira resposta calculada é a do modelo completo.

```
%Aplica as reduções de modelo
[KR(:,:,1), MR(:,:,1), T(:,:,1), Kss] = GuyanReduction(K , M , SlaveDofs);
[KR(:,:,2), MR(:,:,2), T(:,:,2)] = ImprovedReducedSystem(K , M, SlaveDofs);
[KR(:,:,3), MR(:,:,3), T(:,:,3)] = SEREP(K , M,...
 SlaveDofs);

%Calcula a resposta do sistema completo
intervalo = [0 2000];
d_omega = 0.1;
[omega,x0] = calcula_resposta(M,K,F,intervalo,d_omega);
```

A FRF reproduz o método apresentado na seção 2.1 e retorna uma matriz, a qual foi escrita como $x0$, com a amplitudes por linha e nós por coluna. Essa função também retorna o vetor *omega* com as frequências nas quais as amplitudes foram calculadas. Esta função, por meio de um *loop* do tipo *for*, calcula a equação 2.9 em casa frequência do intervalo definido anteriormente.

```
function [omega,X0] = resposta_amortecida(M,K,F,intervalo,d_omega)
%Funcao para calcular a resposta em frequencia de um sistema

aux = 1;
for i = intervalo(1) : d_omega : intervalo(2)
omega(aux) = i;
X0(aux,:) = (K - (omega(aux)^2)*M)\F;
aux = aux+1;
end
end
```

Em seguida, um *loop* percorre o tensor de terceira ordem passando as matrizes de cada modelo para a função que calcula a FRF. As matrizes de amplitude são então gravadas em um outro tensor. Neste trabalho analisaremos as amplitudes de vibração apenas do primeiro nó do sistema (o nó que corresponde ao rotor). Portanto, a primeira coluna de cada matriz de amplitude dos diferentes modelos, inclusive do original, são gravados em uma única matriz, a qual foi chamada de X. Esta manobra é feita para plotar as diferentes curvas lado a lado utilizando a função semilogx do Matlab. Caso seja necessário observar a amplitude de outros nós, a indexação da matriz X deve ser alterada. Esta implementação pode ser conferida abaixo.

```
%Calcula a resposta dos sistemas reduzidos
for i = 1:(length(KR))
[omega,x0R(:,:,i)] = calcula_resposta(MR(:,:,i),KR(:,:,i),FR,intervalo,...
d_omega);
end

%Coloca tudo em um vetor para ser plotado no mesmo gráfico
X(:,1) = x0(:,1);
X(:, 2:(length(KR) + 1)) = x0R(:, 1, 1:length(KR));

%Plota os resultados
semilogx(omega,20*log10(abs(X)))
nomes = ["Completo", "Guyan", "IRS", "SEREP"];
legend(nomes)
%axis([1 10 -60 100])
xlabel('Frequência (rad/s)')
ylabel('Amplitude (dB)')
title('FRF Rotor')
```

3.3 Técnicas de Redução de Modelo

A implementação de todas as técnicas de redução de modelo apresentadas neste trabalho devem primeiro passar por alguns passos, que consistem basicamente em organizar as matrizes de rigidez K e massa M dos sistemas originais de maneira que elas fiquem com a estrutura apresentada na equação 2.18. Para tal, é necessário identificar quantos graus de liberdade há no sistema original. Esta identificação é realizada através da função *length* do Matlab.

Em seguida, um vetor é criado e chamado de *index* com o uma sequência consecutiva partindo de 1 até o número de graus de liberdade. Este vetor representa as posições de todos os nós do sistema. Então, organiza-se o vetor de posição dos nós escravos, para o caso destes serem passados fora de ordem para a função. Por fim, retira-se do vetor *index* as posições dos nós escravos de maneira que reste somente os graus de liberdade mestre.

```
function [M,K] = organiza_matrizes(M,K,SlaveDofs)
Dof = length(K(:,1));
SlaveDofs = sort(SlaveDofs);
index = 1 : Dof ;

index(SlaveDofs) = [];
```

Agora, temos dois vetores para indexação das matrizes no processo de reorganização, o vetor das posições dos nós escravos e o vetor das posições dos nós mestres. Primeiro, montam-se as matrizes K_{mm} e M_{mm}, ou seja, matrizes que possuem somente os nós mestres. Para tal, utiliza-se dois *loops for* aninhados, de maneira que um percorra as linhas das e outro as colunas das matrizes. As matrizes K_{mm} e M_{mm} são então iguais as matriz K e M indexadas com o vetor dos graus de liberdade mestres *index*.

O processo é análogo na produção da matriz K_{ss} e M_{ss}, trocando o vetor das posições mestres pelo das posições escravas. Posteriormente, o mesmo arranjo de *loops* é utilizado para o cálculo das matrizes K_{sm} e M_{sm}, porém utilizando os dois vetores de posição como indexadores. Para estas matrizes, o vetor de posições escravas indexa as linhas enquanto o vetor de posições mestres indexa as colunas. O cálculo das matrizes K_{ms} e M_{ms} é dispensável uma vez que

$$\begin{aligned} M_{sm} &= M_{ms}^T \\ K_{sm} &= K_{ms}^T \end{aligned} \tag{3.5}$$

Com as matrizes M_{mm}, M_{sm}, M_{ss}, K_{mm}, K_{sm} e k_{ss} calculadas, o processo de organização de matrizes é concluído com as matrizes M e K sendo reescritas como na equação 2.18.

```
for i = 1 : length(index)
for j = 1 : length(index)
Mmm(i,j) = M(index(i),index(j));
Kmm(i,j) = K(index(i),index(j));
end
end


for i = 1 : length(SlaveDofs)
for j = 1 : length(SlaveDofs)
Mss(i,j) = M(SlaveDofs(i),SlaveDofs(j));
Kss(i,j) = K(SlaveDofs(i),SlaveDofs(j));
end
end

for i = 1 : length(SlaveDofs)
for j = 1 : length(index)
Msm(i,j) =  M(SlaveDofs(i),index(j));
Ksm(i,j) =  K(SlaveDofs(i),index(j));
end
end

M = [Mmm, Msm'; Msm, Mss];
K = [Kmm, Ksm'; Ksm, Kss];
end
```

3.3.1 Condensação Estática

Após a organização das matrizes de massa M e rigidez K, calcula-se a parte de baixo da matriz 2.24, a qual foi nomeada P no código. Em seguida, atribui-se uma matriz identidade com tamanho igual ao número de colunas da matriz P, utilizando a função *eye* do Matlab. Então, escreve-se a matriz de transformação de Guyan como na equação 2.24.

Com a matriz de transformação W calculada, basta aplica-la às matrizes de rigidez e massa como na equação 2.17. Vale ressaltar, que as multiplicações realizadas com a matriz W e sua transposta devem ser realizada sobre as matrizes M e K organizadas e não sobre as que foram passadas para a função. Este trecho do algoritmo foi implementado da seguinte forma:

```
[M,K] = organiza_matrizes(M,K,SlaveDofs)
P = - inv(Kss) * Ksm;
tamanhos = size(P);
```

```
W = [ eye(tamanhos(2)); P ];
MR = W'*M*W;
KR = W'*K*W;
```

3.3.2 Improved Reduction System

Tal qual a condensação estática, o IRS necessita que as matrizes de massa M e rigidez K sejam organizadas com o procedimento descrito no início da seção 3.3. Em seguida, o procedimento da transformação de Guyan deve ser realizado na íntegra para que a matriz W_{IRS} possa ser calculada como na equação 2.33. No entanto, antes de calcular W_{IRS}, a matriz S deve ser calculada como na equação 2.34.

Vale ressaltar que os zeros da matriz S da equação 2.34, são matrizes zero, então para conseguir calcular a matriz S, não bata inserir os zeros nas outras posições que não a de K_{ss}, pois a matriz S deve ter as mesmas dimensões da matriz K do modelo inicial. Então, ao invés de inserir apenas zeros, deve-se inserir matrizes de zeros com as dimensões de K_{mm}, K_{ms} e K_{sm}, respectivamente. Isso pode ser realizado com a função *zeros* do Matlab, mas, neste caso, foram utilizadas as próprias submatrizes da Matriz K multiplicadas por zero.

```
[M,K] = organiza_matrizes(M,K,SlaveDofs)
P = - inv(Kss) * Ksm;
tamanhos = size(P);
W = [ eye(tamanhos(2)); P ];
MR = W'*M*W;
KR = W'*K*W;

S = [Kmm*0, Ksm'*0; Ksm*0, inv(Kss)];
Wirs = W + S*M*W*inv(MR)*KR ;
MR = Wirs'*M*Wirs;
KR = Wirs'*K*Wirs;
```

Por fim, as matrizes de M e K **organizadas** como na equação 2.18 são multiplicadas por W_{IRS}^T e W_{IRS} como na equação 2.17 para obtenção do modelo reduzido pelo método IRS.

3.3.3 System Equivalent Reduction-Expansion Process

Seguindo o mesmo procedimento inicial dos demais algoritmos de redução, o primeiro passo é organizar as matrizes de massa M e K tal qual a equação 2.18, seguindo o procedimento adotado no início da seção 3.3. Com as matrizes do modelo espacial organizadas, calcula-se os autovetores e os autovalores do sistema utilizando a função *eig* do

Matlab. Posteriormente, separa-se a matriz modal com as coordenadas mestres Φ_{mm} da matriz modal com linhas escravas e colunas mestres Phi_{sm}.

Feito isso, a matriz de transformação W_{SEREP} deve ser calculada como na equação 2.44 e aplicada às matrizes **organizadas** do modelo espacial original. Tal aplicação deve ser realizada como proposto na equação 2.17. O algoritmo para a redução de modelo do tipo SEREP foi implementado da seguinte maneira:

```
[M,K] = organiza_matrizes(M,K,SlaveDofs)
[autovetores,autovalores] = eig(K,M);
tam = length(index);
PHImm = autovetores(1:tam, 1:tam);
PHIsm = autovetores((tam + 1):length(autovetores), 1:tam);
W = [PHImm;PHIsm]*PHImm^(-1);
MR = W'*M*W;
KR = W'*K*W;
```

4 Resultados

Com o algoritmo para comparar as respostas no domínio da frequência dos diferentes modelos produzidos a partir de um sistema de n graus de liberdade, algumas análises podem ser realizadas para caracterizar as diferentes técnicas de redução de modelo. Este capítulo se dedica a detalhar cada estudo realizado com o código e apresentar os dados resultantes. Foram realizadas:

- Validação
- Medidas de precisão com a métrica de análise de erro da seção 2.4.
- Testes de consumo computacional.
- Testes de variação da inércia dos graus de liberdade escravos.
- Testes de variação do número de elementos do modelo espacial discretizado.

4.1 Validação

Para saber se discretização do modelo inicial, apresentado em 2.2 foi realizada corretamente e entender seus efeitos, realiza-se um estudo sobrepondo as FRFs produzidas pelo modelo analítico e pelo modelo discretizado. Tal comparação pode validar o algoritmo de discretização, uma vez o modelo analítico é um caso particular do modelo discretizado com massa específica $\rho = 0$ e número de elementos por eixo $nepe = 1$.

Então, utilizando um código com o modelo analítico e o código apresentado na seção 3.1, extrai-se as duas respostas no domínio da frequência sobrepondo-as, como na FIGURA 5.

Como é possível observar, a curva da resposta de um dos graus de liberdade produzida pelo novo algoritmo se ajusta perfeitamente à curva do modelo analítico. Também podemos observar este comportamento para todos os graus de liberdade do sistema utilizando a métrica de erro apresentada na seção 2.4. O erro relativo γ entre as duas respostas foi calculado em toda a gama de frequência da FIGURA 5 e plotada na FIGURA 6. Na teoria, as FRFs apresentadas não possuiriam pontos máximos como é observado na FIGURA 5. Isto só ocorre pois existe um erro de discretização de frequência no cálculo

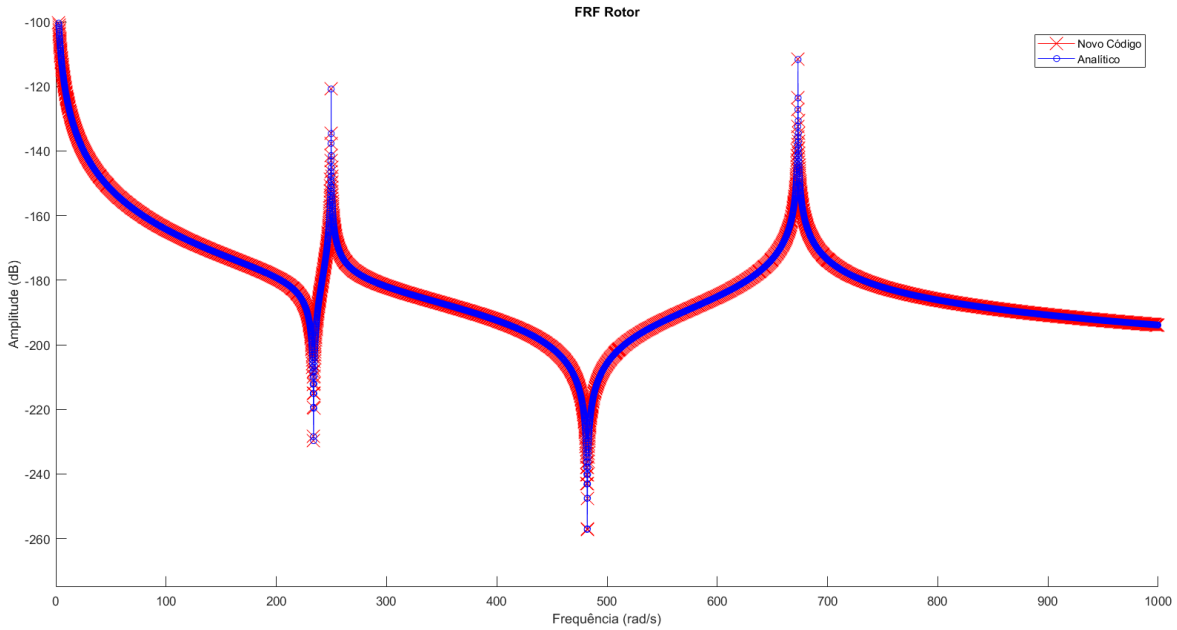

Figura 5 – FRF do modelo analítico sobreposta à FRF produzida pelo código para o caso particular.

numérico da FRF. Neste trabalho, cada ponto de frequência calculado possui uma diferença de $0,1 rad/s$ dos pontos vizinhos para todos os gráficos.

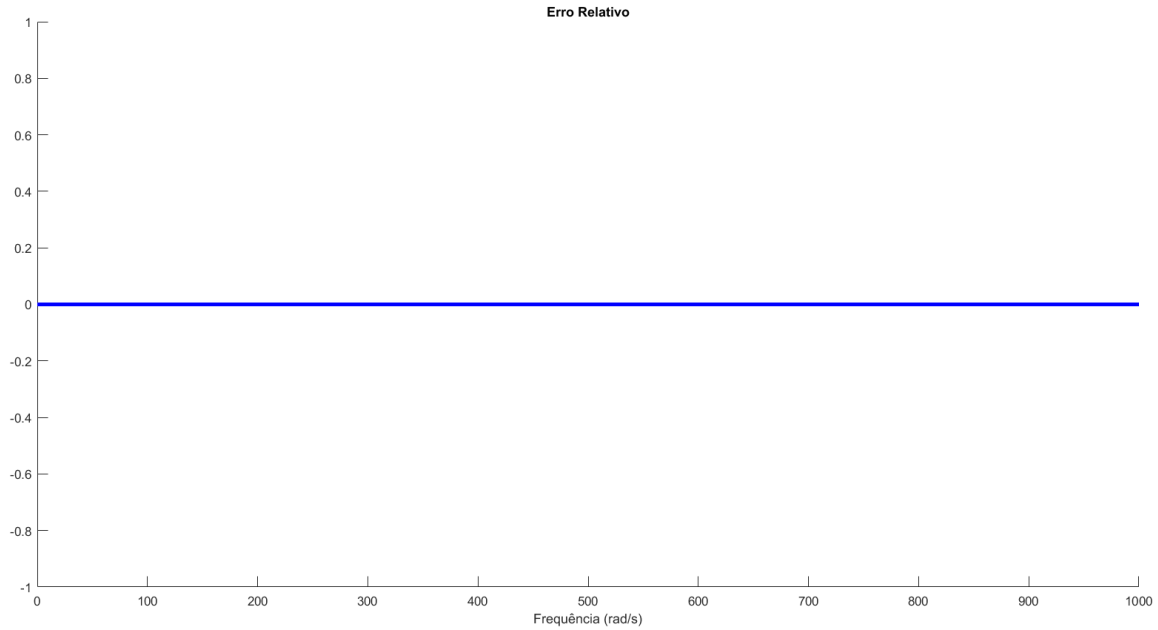

Figura 6 – Erro relativo γ entre a resposta do modelo analítico e o modelo produzido pelo código.

4.2 Variação do Número de Graus de Liberdade

Como mostrado na seção 3.1, alterando o valor de *nepe* (número de elementos por eixo), é possível variar o número de graus de liberdade do sistema a ser formado, através

da inserção de eixos discretizados. Então, foram produzidas duas FRFs, uma com 3 e outra com 1001 graus de liberdade. Tais curvas correspondem ao modelo reduzido por condensação estática e foram plotadas e sobrepostas na FIGURA 7.

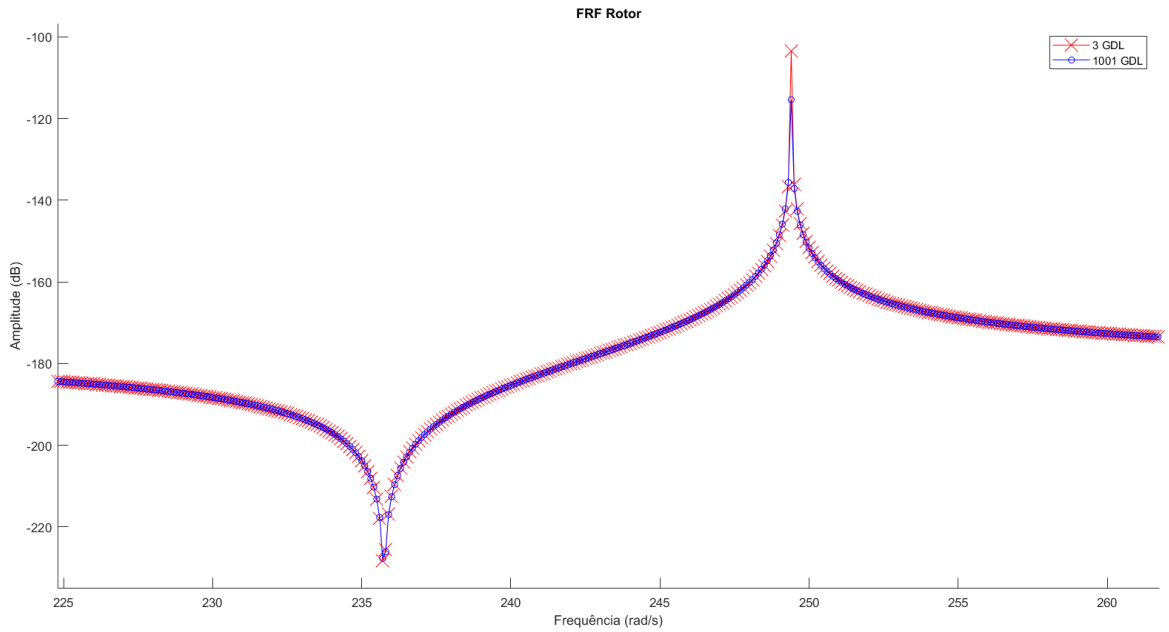

Figura 7 – Sobreposição das FRFs produzidas com 3 e 1001 graus de liberdade.

Como é possível observar, as duas curvas se ajustam entre si, indicando que a resposta do rotor não sofre influência do aumento do número de graus de liberdade escravos. Isso se deve à sua grande inércia. De maneira análoga, o mesmo ocorre com o gerador.

Porém, o mesmo não ocorre com os graus de liberdade escravos. Utilizando a métrica de análise de erro da seção 2.4 percebemos grande diferença entre entre os dois casos (com poucos e muitos graus de liberdade). Antes de aplicar a métrica, os graus de liberdade desprezados pelas reduções são recuperados multiplicando suas respostas por cada matriz de transformação W. Foram plotadas seis curvas do erro γ entre modelos reduzidos e modelo completo, sendo três com 11 graus de liberdade (mostradas na FIGURA 8) e três com 801 graus de liberdade (mostradas na FIGURA 9).

Pode-se notar que no intervalo entre 0 e 300 rad/s a curva é bem sensível ao número de elementos do sistema, de maneira que o erro γ diminui gradativamente a medida que se aumenta os graus de liberdade. Entretanto, a partir de 400 rad/s γ cresce drasticamente independentemente do número de elementos. Isso se dá porque trata-se de uma comparação entre modelos reduzidos e modelo completo, de tal forma que a resposta dos modelos reduzidos seguem sem grandes flutuações após o primeiro pico de ressonância enquanto o modelo completo ainda possuirá uma região de ressonância e anti-ressonância. Tal afastamento pode ser observado na figura 10. Vale ressaltar que a métrica de análise de erro γ não analisa somete a resposta de um grau de liberdade e sim a resposta todos os graus.

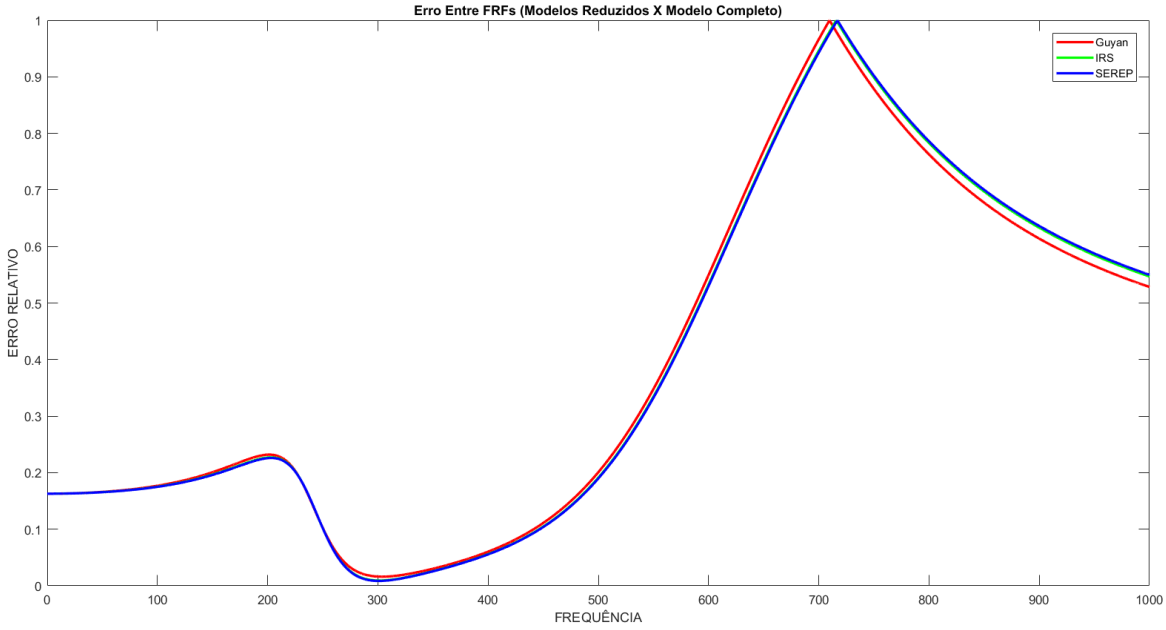

Figura 8 – Erro relativo γ entre modelos reduzidos e modelo completo de 11 graus de liberdade.

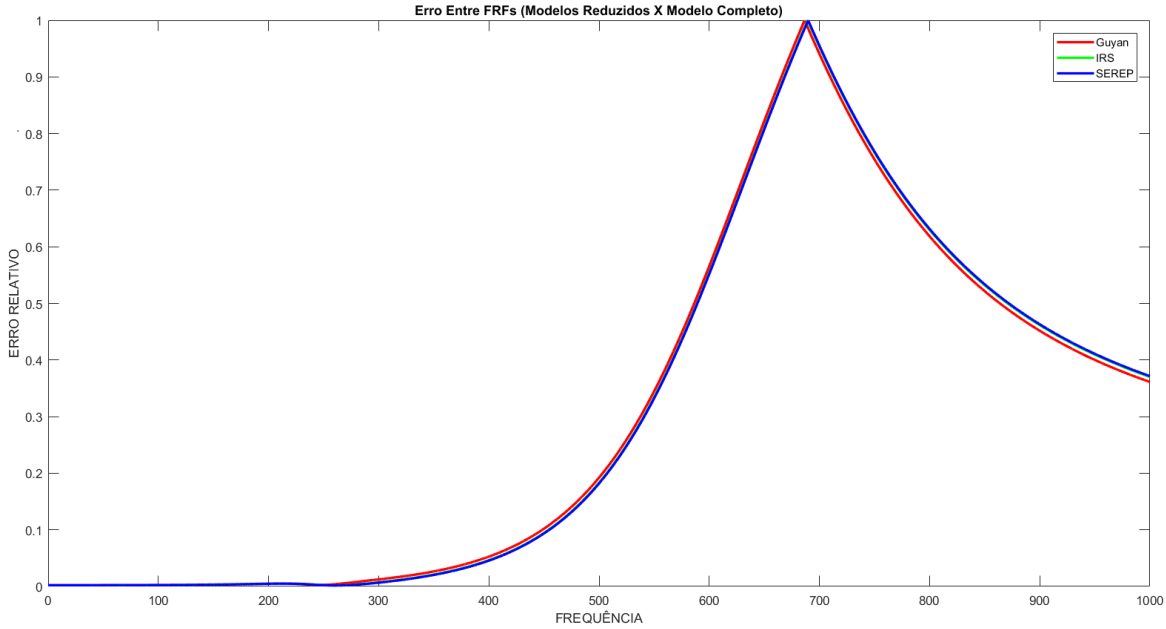

Figura 9 – Erro relativo γ entre modelos reduzidos e modelo completo de 801 graus de liberdade.

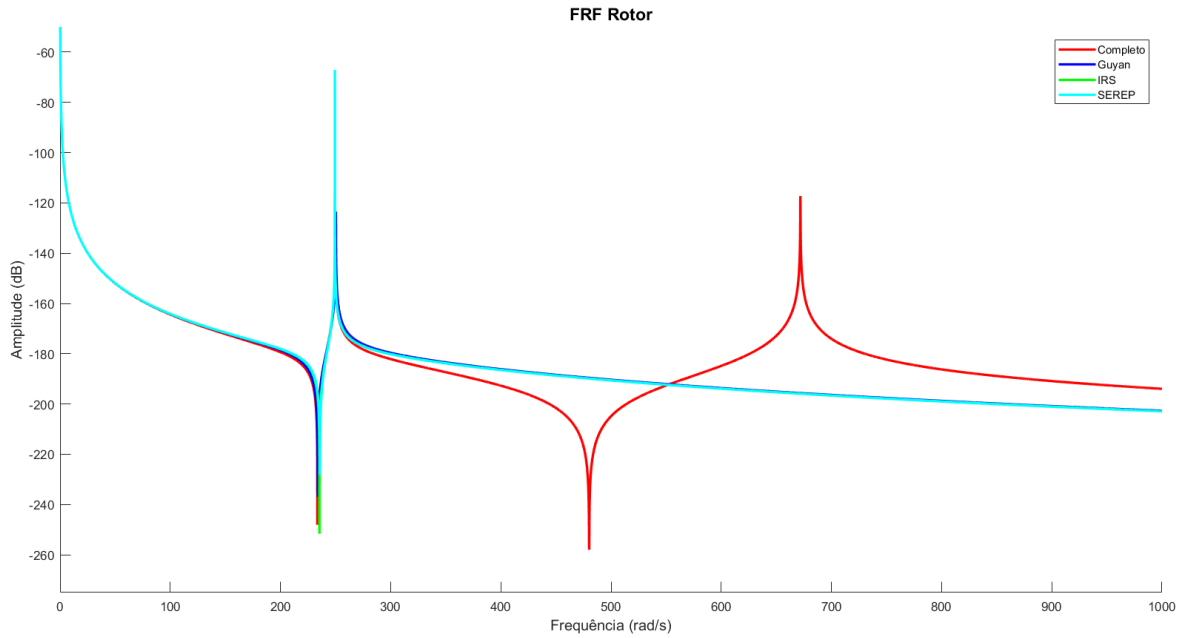

Figura 10 – FRFs do modelos reduzidos e do modelo completo.

4.3 Variação da Inércia dos Graus de Liberdade Escravos

Na formulação do algoritmo de discretização dos eixos do modelo inicial, há a inserção das matrizes elementares de massa com a utilização da massa específica do material de que seria composto o eixo. Como os eixos são os escolhidos como graus de liberdade escravos, sua massa foi variada através da alteração da massa específica para produzir variação em sua inércia de rotação. Em seguida, o algoritmo foi utilizado para fornecer a primeira frequência de ressonância de cada modelo (completo, reduzido por Guyan, por IRS e por SEREP) para cada razão de inércia dos eixos por inércia do rotor. O resultado foi plotado na figura 11.

A curva mostra uma diminuição da primeira frequência de ressonância a medida que a inércia dos graus de liberdade escravos aumentam. A queda da frequência de ressonância foi de aproximandamente 10 rad/s variando a inércia dos eixos a até 100% da inércia do rotor.

4.4 Tempo de CPU

Afim de estudar o custo computacional de cada redução de modelo, o algoritmo desenvolvido foi utilizado variando-se o número de elementos do sistema a ser reduzido enquanto o tempo de CPU foi medido para cada técnica de redução de modelo. Tal medição foi realizada através da função *cputime* do Matlab, como mostrado a seguir:

```
t1 = cputime;
```

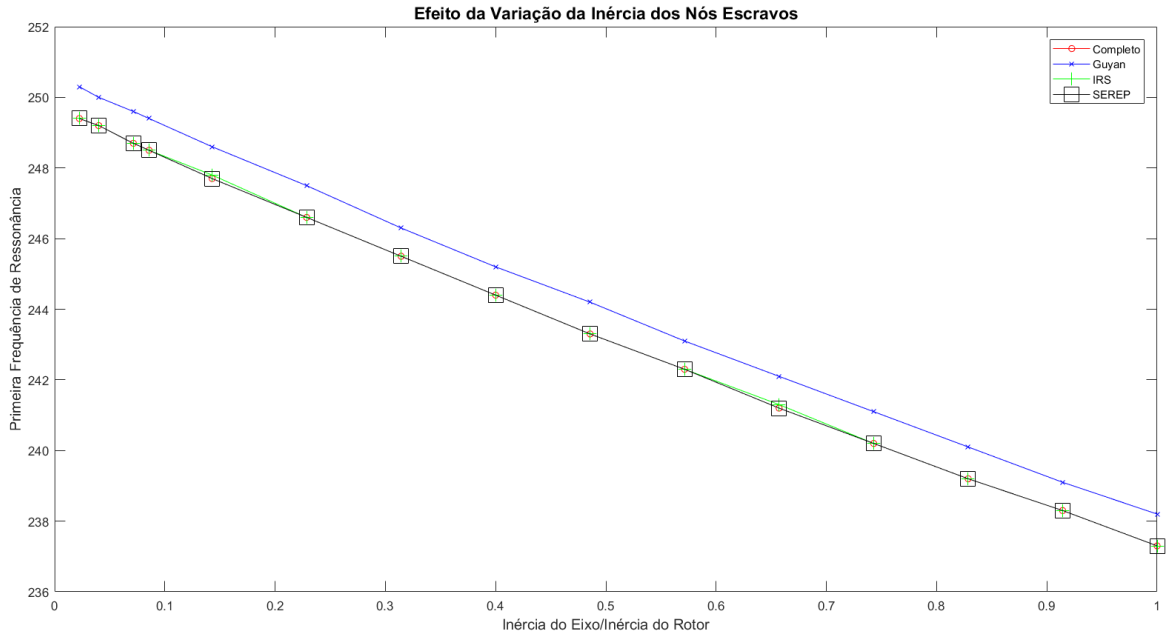

Figura 11 – Variação da inércia dos graus de liberdade escravos.

```
%Função ou operações de redução de modelo
tempo_decorrido = cputime - t1
```

Posteriormente, os dados de tempo por número de elementos foram plotados na FIGURA 12 para cada método de redução de modelo. A CPU utilizada foi uma Intel Core I7 de sétima geração com velocidade base de $2,8GHz$, 4 *cores* e 8 processadores lógicos. Nenhum tipo de processamento paralelo foi utilizado.

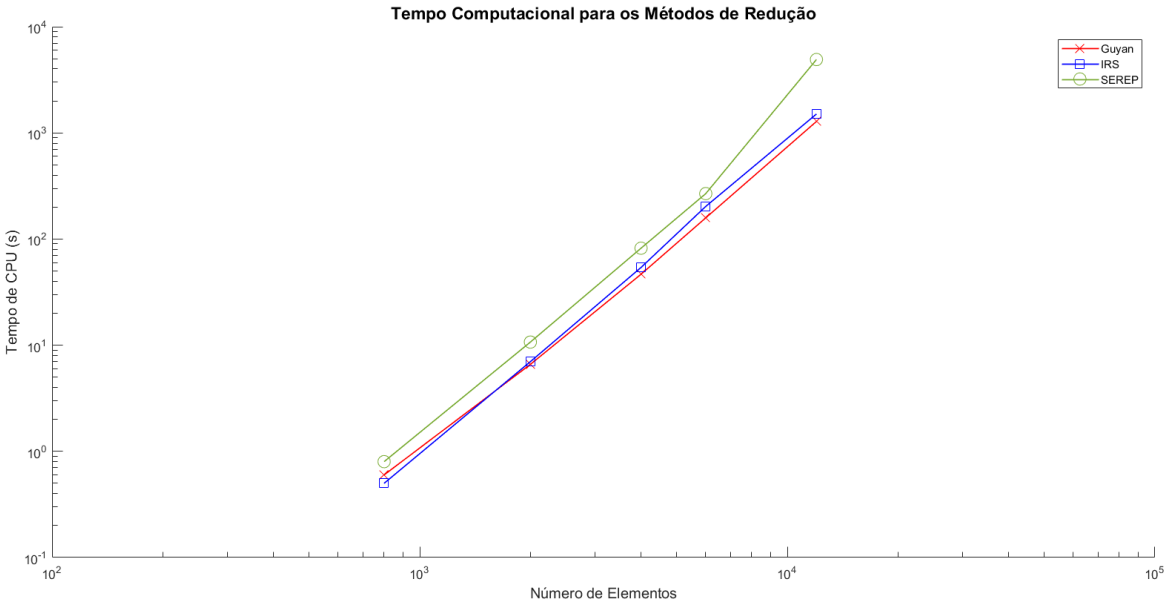

Figura 12 – Efeito da variação do número de elementos do sistema no tempo de CPU em escala logarítmica.

O método SEREP se mostra mais lento que os demais. Já a redução de Guyan, é a mais rápida, seguida da da redução por IRS, o qual se mostra quase tão rápida. As curvas

de IRS e Guyan se interceptam porque para baixo número de graus de liberdade, há grande flutuação relativa do tempo consumido pelo computador para executar as operações.

É possível observar que o tempo de CPU aumenta exponencialmente com o número de elementos adotados para o sistema. Isso se deve ao fato de que todos as técnicas envolvem uma ou duas inversões de matrizes. Vale ressaltar que para cada ponto calculado nas FRFs, há uma operação de inversão de matriz, de tal sorte que calcular a resposta em frequência de grandes sistemas se torna inviável caso não seja utilizada alguma técnica de redução de modelo.

5 Conclusão

Foi implementado o algoritmo com capacidade de discretizar o modelo proposto 2.2 na seção em n graus de liberdade, de reduzir tal sistema através de diferentes técnicas de redução de modelo e calcular a resposta no domínio da frequência de cada modelo produzido para a devida comparação. O modelo inicial foi validado por meio de comparação com a resposta calculada a partir de uma solução analítica (sem discretização). Com este código implementado em Matlab, foi possível realizar estudos de precisão e tempo utilizando a métrica de análise de erro da seção 2.4 e a medição do tempo de CPU para cada redução.

Foi verificado que, para o modelo explorado, todos os métodos de redução apresentam boa precisão. No entanto, SEREP foi mais preciso que os demais métodos no que se refere à amplitude de vibração, seguido do IRS (quase tão preciso quanto) e do método de Guyan, o menos preciso de todos. Quanto a frequência de ressonância, SEREP e IRS foram igualmente precisos, seguidos por Guyan, o menos preciso.

Verificou-se também que o tempo de execução da técnica que Guyan foi o menor, seguido por IRS e SEREP. Tais observações revelam o IRS como uma boa alternativa, já que possui precisão comparável ao SEREP, porém com um menor tempo de execução. Vale lembrar que com os métodos de redução de modelo, há a inversão de matrizes da ordem do sistema original apenas uma ou duas vezes antes do cálculo da resposta, a qual se dará efetuando inversões em matrizes de ordem reduzida. Caso não se utilize nenhuma técnica de redução, o cálculo da resposta se torna inviável para grandes sistemas, uma vez que haverá inversão de matrizes de elevada ordem para cada frequência calculada. Vale ressaltar que não foram utilizadas matrizes esparsas e nenhum tipo de processamento paralelo durante a execução dos cálculos. Os resultados obtidos poderiam ser alterados caso as reduções fossem feitas com outros modelos, com outras formas de implementar ou com outras linguagens de programação. As ordens dos erros numéricos não foram determinadas, com exceção do erro por discretização de frequência, uma vez que os pontos das FRFs possuem $0,1 rad/s$ de distância entre os pontos vizinhos.

O código apresentado ainda pode ser explorado de outras maneiras. O modelo inicial proposto possibilita uma escolha conveniente dos graus de liberdade escravos. Talvez um sistema mais homogêneo, como uma viga em vibração transversal, revele maiores diferenças entre as técnicas de redução de modelo. A estrutura do código permite que suas

primeiras linhas sejam eliminadas e quaisquer matrizes de massa M e de rigidez K sejam escritas na memória para sofrer os processos de redução, constituindo uma facilidade de realizar o mesmo estudo com outros modelos. Além disso, se programada outra função para outro método de redução de modelo, esta função poderia ser inserida no *script* principal com apenas uma linha de código. Uma melhoria significativa poderia ser realizada no código para investigar se há vantagem no uso de métodos de redução de modelo em se tratando de espaço consumido na memória do computador.

Referências

ERNANDEZ, F. Serep: System equivalent reduction-expansion process. 2006. Citado na página 11.

EWINS, D. *Modal Testing: Theory, Practice, and Application*. [S.l.]: Research Studies Press, 2000. Citado na página 4.

FRISWELL, M.; GARVEY, S.; PENNY, J. Model reduction using dynamic and iterated irs techniques. *Journal of sound and vibration*, Elsevier, v. 186, n. 2, p. 311–323, 1995. Citado na página 10.

FRISWELL, M.; MOTTERSHEAD, J. E. *Finite element model updating in structural dynamics*. [S.l.]: Springer Science & Business Media, 1995. v. 38. Citado na página 10.

FRISWELL, M. I.; LEES, A.; LITAK, G. Torsional vibration of machines with gear errors. *Journal of Physics: Conference Series*, v. 305, p. 07, 2011. Citado 2 vezes nas páginas 6 e 14.

GUYAN, R. J. Reduction of stiffness and mass matrices. *AIAA journal*, v. 3, n. 2, p. 380–380, 1965. Citado na página 10.

KWON, Y. W.; BANG, H. *The Finite Element Method Using Matlab*. [S.l.]: CRC Press, 1997. Citado na página 1.

QU, Z.-Q. *Model order reduction techniques with applications in finite element analysis*. [S.l.]: Springer Science & Business Media, 2013. Citado 2 vezes nas páginas 2 e 4.

SELLGREN, U. Fe condensation methods. *Component Mode Synthesis - A method for efficient dynamic simulation of complex technical systems*, p. 5, 2003. Citado 2 vezes nas páginas 8 e 13.

Apêndices

A Código Matlab - Comparação Entre as Técnicas de Redução de Modelo

```
clc; close all; clear;
%Carlos Lima Santoro
%Script para comparar diferentes métodos de redução de modelo

%Monta o sistema
DadosFriswell
DadosEixos
Sistema1Eixos

%Aplica as reduções de modelo
[KR(:,:,1), MR(:,:,1), T(:,:,1), Kss] = GuyanReduction(K , M ,... SlaveDofs);
[KR(:,:,2), MR(:,:,2), T(:,:,2)] = ImprovedReducedSystem(K , M ,... SlaveDofs);
[KR(:,:,3), MR(:,:,3), T(:,:,3)] = SEREP(K , M , SlaveDofs);

%Calcula a resposta do sistema completo
intervalo = [0, 1000];
d_omega = 0.1;
[omega,x0] = calcularesposta(M,K,F,intervalo,d_omega);

%Calcula a resposta dos sistemas reduzidos
for i = 1:(length(KR))
[omega,x0R(:,:,i)] = calcularesposta(MR(:,:,i),KR(:,:,i),FR,intervalo,d_omega);
end

%Coloca tudo em um vetor para ser plotado no mesmo gráfico
X(:,1) = x0(:,1);
X(:, 2:(length(KR) + 1)) = x0R(:, 1, 1:length(KR));
```

```
%Plota os resultados
plot(omega,20*log10(abs(X)))
nomes = ["Completo", "Guyan", "IRS", "SEREP"];
legend(nomes)
xlabel('Frequência (rad/s)')
ylabel('Amplitude (dB)')
title('FRF Rotor')
```

B Código Matlab - DadosFriswell

```
N2 = 56;
N3 = 111;   %numero de dentes
n = N2/N3;  %razão de engrenamento

J1 = 7e3;
J2 = 4e3;
J3 = 13e3;
J4 = 5e3;   %Inércias [kgm^2]

k1 = 1.6e9;
k2 = 0.29e9;    %Rigidez dos eixos [Nm/rad]
```

C Código Matlab - DadosEixos

```
%Propriedades gerais do aço
ro = 7860; %Densidade do aço em kg/m^3

G = 80e9; %Módulo de cisalhamento em Pa

l1 = 1; %Comprimento do eixo 1 em metros
l2 = 1; %Comprimento do eixo 2 em metros

%Diâmetros em função de k, l e G
d1 = (32*k1*l1/(G*pi))^(1/4); %Diâmetro do eixo 1 em metros
d2 = (32*k2*l2/(G*pi))^(1/4); %Diâmetro do eixo 2 em metros

m_eixo1 = ro*l1*pi*d1^2/4; %Massa do eixo 1 em kg
m_eixo2 = ro*l2*pi*d2^2/4; %Massa do eixo 2 em kg

Je1 = m_eixo1*(d1/2)^2/2; %Inércias de giro dos eixos em kg.m^2
Je2 = m_eixo2*(d2/2)^2/2;

disp(Je1/J1);
```

D Código Matlab - Sistema1Eixos

```
%Esta rotina define o sistema com base nos dados de Friswell e nos dados
%dos eixos, discretizando o sistema em diversos pedaços.
nepe = 5; %Número de elementos por eixo
ne = 2*nepe; %Número de elementos
nn = ne + 1; %Número de nós

le1 = l1/nepe; %Comprimento do elemento do eixo 1
le2 = l2/nepe; %Comprimento do elemento do eixo 2

A1 = pi*d1^4/32; %Momento polar de área do eixo 1
A2 = pi*d2^4/32; %Momento polar de área do eixo 2

Ke1 = (G*A1/le1)*[1,-1;-1,1]; %Matriz de rigidez dos elementos do eixo 1
Ke2 = (G*A2/le2)*[1,-1;-1,1]; %Matriz de rigidez dos elementos do eixo 2

Me1 = (ro*A1*le1/6)*[2,1;1,2]; %Matriz de massa dos elementos do eixo 1
Me2 = (ro*A2*le2/6)*[2,1;1,2]; %Matriz de massa dos elementos do eixo 2

nos = zeros(ne,2); %Matriz de nós (cada posição da matriz é relativa...
 a um elemento)
for i = 1:ne
nos(i, 1:2) = [i, i+1];
end

%Montando as matrizes de massa e rigidez
M = zeros(nn,nn);
K = M;
for i = 1:(ne/2)
M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) = M(nos(i,1):nos(i,2),...
 nos(i,1):nos(i,2)) + Me1;
K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) = K(nos(i,1):nos(i,2),...
 nos(i,1):nos(i,2)) + Ke1;
```

```
end
for i = (ne/2 + 1):ne
M(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) = M(nos(i,1):nos(i,2),...
 nos(i,1):nos(i,2)) + Me2;
K(nos(i,1):nos(i,2), nos(i,1):nos(i,2)) = K(nos(i,1):nos(i,2),...
 nos(i,1):nos(i,2)) + Ke2;
end
[M, K] = noespecial(nn, M, K, Ke2);

%Definição dos graus de liberdade escravos
SlaveDofs = 2:(nn-1);

%Definição dos vetores de forçamento
F = [1; zeros(nn - 1, 1)];
FR = [1; zeros(nn - length(SlaveDofs) - 1, 1)];
```

E Código Matlab - noespecial

```
function [M, K] = no_especial(nn, M, K, Ke2)
%Esta função adiciona o valor dos nós especiais às matrizes M e K.

%Carrega os dados do sistema
Dados_Friswell
Dados_Eixos
ne = nn - 1;

%Escreve na matriz
M(1,1) = M(1,1) + J1;

M((ne/2 + 1), (ne/2 + 1)) = M((ne/2 + 1), (ne/2 + 1))*(1 + n^2)...
 + J2 + J3*n^2;
M((ne/2 + 1), (ne/2 + 1) + 1) = M((ne/2 + 1), (ne/2 + 1) + 1)*(-n);
M((ne/2 + 1) + 1, (ne/2 + 1)) = M((ne/2 + 1) + 1, (ne/2 + 1))*(-n);

K((ne/2 + 1), (ne/2 + 1)) = K((ne/2 + 1), (ne/2 + 1)) - Ke2(1,1)...
 + Ke2(1,1)*n^2;
K((ne/2 + 1), (ne/2 + 1) + 1) = K((ne/2 + 1), (ne/2 + 1) + 1)*(-n);
K((ne/2 + 1) + 1, (ne/2 + 1)) = K((ne/2 + 1) + 1, (ne/2 + 1))*(-n);

M(nn, nn) = M(nn, nn) + J4;
end
```

F Código Matlab - GuyanReduction

```
function [KR,MR,W,Kss]=GuyanReduction(K , M , SlaveDofs)
%
%   Redução de Guyan

Dof = length(K(:,1));
SlaveDofs = sort(SlaveDofs);
index = 1 : Dof ;

index(SlaveDofs) = [];

for i = 1 : length(index)
for j = 1 : length(index)
Mmm(i,j) = M(index(i),index(j));
Kmm(i,j) = K(index(i),index(j));
end
end


for i = 1 : length(SlaveDofs)
for j = 1 : length(SlaveDofs)
Mss(i,j) = M(SlaveDofs(i),SlaveDofs(j));
Kss(i,j) = K(SlaveDofs(i),SlaveDofs(j));
end
end

for i = 1 : length(SlaveDofs)
for j = 1 : length(index)
Msm(i,j) =  M(SlaveDofs(i),index(j));
Ksm(i,j) =  K(SlaveDofs(i),index(j));
end
```

```
end

%-----------------------------------------------------------------%
%Reorganizando as matrizes do sistema original
M = [Mmm, Msm'; Msm, Mss];
K = [Kmm, Ksm'; Ksm, Kss];
%-----------------------------------------------------------------%

P = - inv(Kss) * Ksm;
tamanhos = size(P);
W = [ eye(tamanhos(2)); P ];
MR = W'*M*W;
KR = W'*K*W;
```

G Código Matlab - ImprovedReducedSystem

```
function [KR,MR,Wirs] = ImprovedReducedSystem(K,M,SlaveDofs)
%Guyan----------------------------------------------------------%
Dof = length(K(:,1));
SlaveDofs = sort(SlaveDofs);
index = 1 : Dof ;

index(SlaveDofs) = [];

for i = 1 : length(index)
for j = 1 : length(index)
Mmm(i,j) = M(index(i),index(j));
Kmm(i,j) = K(index(i),index(j));
end
end


for i = 1 : length(SlaveDofs)
for j = 1 : length(SlaveDofs)
Mss(i,j) = M(SlaveDofs(i),SlaveDofs(j));
Kss(i,j) = K(SlaveDofs(i),SlaveDofs(j));
end
end

for i = 1 : length(SlaveDofs)
for j = 1 : length(index)
Msm(i,j) =  M(SlaveDofs(i),index(j));
Ksm(i,j) =  K(SlaveDofs(i),index(j));
end
end
```

```
%-----------------------------------------------------------------%
%Reorganizando as matrizes do sistema original
M = [Mmm, Msm'; Msm, Mss];
K = [Kmm, Ksm'; Ksm, Kss];
%-----------------------------------------------------------------%

P = - inv(Kss) * Ksm;
tamanhos = size(P);
W = [ eye(tamanhos(2)); P ];
MR = W'*M*W;
KR = W'*K*W;
%-----------------------------------------------------------------%

S = [Kmm*0, Ksm'*0; Ksm*0, inv(Kss)];

Wirs = W + S*M*W*inv(MR)*KR ;

MR = Wirs'*M*Wirs;
KR = Wirs'*K*Wirs;
```

H Código Matlab - SEREP

```
function [KR,MR,W] = SEREP3(K, M, SlaveDofs)

Dof = length(K(:,1));
SlaveDofs = sort(SlaveDofs);
index = 1 : Dof ;

index(SlaveDofs) = [];

for i = 1 : length(index)
for j = 1 : length(index)
Mmm(i,j) = M(index(i),index(j));
Kmm(i,j) = K(index(i),index(j));
end
end



for i = 1 : length(SlaveDofs)
for j = 1 : length(SlaveDofs)
Mss(i,j) = M(SlaveDofs(i),SlaveDofs(j));
Kss(i,j) = K(SlaveDofs(i),SlaveDofs(j));
end
end

for i = 1 : length(SlaveDofs)
for j = 1 : length(index)
Msm(i,j) =  M(SlaveDofs(i),index(j));
Ksm(i,j) =  K(SlaveDofs(i),index(j));
end
end

%-------------------------------------------------------------%
%Reorganizando PHIsm matrizes do sistema original
```

```
M = [Mmm, Msm'; Msm, Mss];
K = [Kmm, Ksm'; Ksm, Kss];
%-----------------------------------------------------------------%

[autovetores,autovalores] = eig(K,M);

tam = length(index);
PHImm = autovetores(1:tam, 1:tam);
PHIsm = autovetores((tam + 1):length(autovetores), 1:tam);

W = [PHImm;PHIsm]*PHImm^(-1);
MR = W'*M*W;
KR = W'*K*W;
```

I Código Matlab - calcularesposta

```
function [omega,X0] = resposta_amortecida(M,K,F,intervalo,d_omega)
%Funcao para calcular a resposta em frequencia de um sistema amortecido
% Problema: (K + w*C*j - w^2M)*X0 = F

aux = 1;
for i = intervalo(1) : d_omega : intervalo(2)
omega(aux) = i;
X0(aux,:) = (K - (omega(aux)^2)*M)\F;
aux = aux+1;
end
end
```

J Código Matlab - Calcula Erros

```
clc; close all; clear;
%Carlos Lima Santoro
%Script para comparar diferentes métodos de redução de modelo

%Monta o sistema
DadosFriswell
DadosEixos
Sistema1Eixos

%Aplica as reduções de modelo
[KR(:,:,1), MR(:,:,1), T(:,:,1), Kss] = GuyanReduction(K , M,...
 SlaveDofs);
[KR(:,:,2), MR(:,:,2), T(:,:,2)] = ImprovedReducedSystem(K , M,...
 SlaveDofs);
[KR(:,:,3), MR(:,:,3), T(:,:,3)] = SEREP(K , M , SlaveDofs);

%Calcula a resposta do sistema completo
intervalo = [0, 1000];
d_omega = 0.1;
[omega,x0] = calcularesposta(M,K,F,intervalo,d_omega);

%Calcula a resposta dos sistemas reduzidos
for i = 1:(length(KR))
[omega,x0R(:,:,i)] = calcularesposta(MR(:,:,i),KR(:,:,i),...
FR,intervalo,d_omega);
end

%Coloca tudo em um vetor para ser plotado no mesmo gráfico
X(:,1) = x0(:,1);
X(:, 2:(length(KR) + 1)) = x0R(:, 1, 1:length(KR));

%Calcula os Dr e orgniza os graus de liberdade em seus respectivos...
```

```
 lugares
tam = size(x0R);
for i = 1:tam(3)
Dr(:,:,i) = T(:,:,i)*x0R(:,:,i)';
aux = Dr(2,:,i);
tam = size(Dr);
for j = 3:(tam(1) - 1)
Dr(j,:,i) = Dr(j + 1, :, i);
end
Dr(tam(1),:,i) = aux;
end

%Aplicação da euqação 11 do artigo do Sellgren
tam = size(Dr);
for j = 1:tam(3)
for i = 1:length(Dr(:,:,1))
gamma(i,j) = 1 - abs(Dr(:,i,j)'*x0(i,:)'/(norm(Dr(:,i,j))*norm(x0(i,:))));
end
end

plot1 = plot(omega, gamma)
set(plot1(1), 'Color', [1 0 0])
set(plot1(2), 'Color', [0 1 0])
set(plot1(3), 'Color', [0 0 1])

nomes = ["Guyan", "IRS", "SEREP"];
legend(nomes)
xlabel('FREQUÊNCIA')
ylabel('ERRO RELATIVO')
```