Mauro Alexandre Janeiro Guerreiro

Licenciado em Ciências de Engenharia Civil

Identificação de Dano Estrutural com Base em Análise Modal

Dissertação para obtenção do Grau de Mestre
em Engenharia Civil - Perfil Estruturas

Orientador: Professor Doutor Corneliu Cismasiu

Júri:

Presidente:	Professor Doutor António Pinto Ramos
Arguente:	Professor Doutor Filipe Santos
Vogal:	Professor Doutor Corneliu Cismasiu

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro de 2014

Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu orientador, Professor Doutor Corneliu Cismasiu, pelo constante incentivo ao longo deste trabalho, pelo optimismo e confiança que prontamente demonstrou quando tive dificuldades e pela constante disponibilidade para me receber.

Ao meu amigo Carlos Calmeiro, quero agradecer não só a amizade partilhada ao longo dos últimos anos mas também as inúmeras horas que disponibilizou para me ajudar no desenvolvimento da aplicação deste projecto, sem a qual teria dificuldades acrescidas.

Aos amigos e colegas que me acompanharam durante o percurso académico, pela amizade, pelo apoio e por todos os bons momentos que ao longo destes anos me proporcionaram as condições necessárias para chegar aqui. Agradeço em particular ao António Ramalho, Bruno Lino Alves, Carlos Prazeres, Cláudia Dias, Diogo Brito, Diogo Pires, Frederico Abegão, João Carvalho Cruz, João Rodrigues, Marli Silva, Mário Sanchez, Miguel Almeida, Ricardo Cross e Rui Simões. Em especial, quero agradecer à Margarida Ferreira por ter sido o meu braço direito ao longo do curso.

Aos meus amigos de longa data André Palma, David Nunes, Diogo Estevão, Hugo Mariani, Miguel Ribeiro, Pedro Maurício, Pedro Leonardo, Ricardo Monteiro e Tiago Jacob por estarem presentes incondicionalmente na minha vida e por tudo o que a sua amizade contribuiu para o desenvolvimento deste trabalho.

Quero agradecer à minha namorada Catarina Secca e Cruz, pelo carinho, pelo incentivo e pela incansável dedicação. O acompanhamento construtivo a este trabalho foi indispensável ao seu desenvolvimento.

Por fim, gostaria de agradecer à minha família, que esteve sempre presente, acreditou em mim desde o princípio e que me deu a força e os meios para o conseguir.

Resumo

O presente trabalho procura mostrar as potencialidades das técnicas de monitorização estrutural, de detecção, localização e caracterização de dano através de alterações dos parâmetros modais.

As características dinâmicas de vibração de uma estrutura são alteradas pela presença de dano, o que, representando uma perda física, conduz à consequente perda de rigidez. Este fenómeno é responsável por modificações nas frequências naturais, deslocamentos modais e respectivas derivadas. Este trabalho apresenta a curvatura das configurações modais como possível parâmetro de controlo na identificação de dano estrutural.

Neste documento encontram-se vários estudos de sensibilidade realizados em três modelos distintos (viga bi-apoiada, viga em consola e pórtico encastrado) em que se pretende avaliar a fiabilidade do método das curvaturas. Foram também desenvolvidos estudos respeitantes às alterações das frequências naturais e às alterações de dois indicadores estatísticos, o *MAC* e o *COMAC*.

Com base nos resultados do método das curvaturas foi desenvolvida uma *Java Applet*, que constitui uma ferramenta de cálculo que permite explorar as potencialidades do método das curvaturas modais na identificação de danos em várias estruturas. Esta aplicação foi desenvolvida no compilador *NetBeans IDE*.

Palavras chave:

Dano, curvatura, frequência natural, identificação modal, *MAC*, *COMAC*, *Java Applet*.

Abstract

The main purpose of this work is to show the potential of the structural monitoring techniques, concerning the detection, location and characterization of structural damage through changes in modal parameters.

The dynamic characteristics of a vibrating structure are modified by the presence of damage which, representing a physical loss, leads to a loss of stiffness. This phenomenon is responsible for changes in natural frequency, modal displacements and the corresponding derivatives. This paper presents the curvature of the elements as a possible control parameter in identifying structural damage.

In this document there are several sensitivity studies conducted in three different models (bi-supported beam, cantilever beam and frame) which purpose is to evaluate the reliability of the curvatures method. Were also developed studies relating to changes in natural frequencies and changes of two statistical indicators, MAC and COMAC.

Based on the results of the curvatures method a Java Applet was developed, which is a calculation tool that allows to explore the potentialities of the method in identifying damage in several structures. This application was developed in NetBeans IDE compiler.

Keywords:

Damage, curvature mode shapes, natural frequencies, modal identification, MAC, COMAC, Java Applet

Índice de Matérias

Índice de Figuras

Índice de Tabelas

Lista de abreviaturas e símbolos

Abreviaturas

MAC	*Modal Assurance Criterion*
COMAC	*Coordinate Modal Assurance Criterion*
IDE	*Integrated Development Environment*
TND	Técnicas de Ensaio Não Destrutivas
TD	Técnicas de Ensaio Destrutivas
MEF	Modelo de Elementos Finitos
DFC	Operador de Diferenças Finitas Centradas
GDL	Grau De Liberdade
JVM	*Java Virtual Machine* - Máquina Virtual Java
API	*Application Programming Interface* - Interface de Programação de Aplicativos

Símbolos

ε	Deformação axial
y	Distância da linha neutra a uma fibra da viga
R	Raio de curvatura
M	Momento flector
θ	Ângulo de flexão
k	Curvatura
σ	Tensão axial
E	Módulo de elasticidade

I	Segundo momento de inércia
dA	Variável de integração A preponderante
ϕ''	Curvatura numa secção
q	Nó
ϕ''_q	Valor da curvatura no ponto *q*
ϕ_{q-1}	Valor do deslocamento no ponto *q - 1*
ϕ_q	Valor do deslocamento no ponto *q*
ϕ_{q+1}	Valor do deslocamento no ponto *q + 1*
h	Comprimento de um elemento finito
EI	Rigidez de flexão
$\frac{d^2\phi}{dx^2}$	Segunda derivada da curvatura
$\frac{d\phi}{dh}$	Primeira derivada da curvatura
f_i	Frequência natural do modelo intacto
f_d	Frequência natural do modelo danificado
$\{\phi_{int}\}$	Vector dos deslocamentos modais do modelo intacto
$\{\phi_{int}\}^T$	Vector transposto dos deslocamentos modais do modelo intacto
$\{\phi_{dan}\}$	Vector dos deslocamentos modais do modelo danificado
$\{\phi_{dan}\}^T$	Vector transposto dos deslocamentos modais do modelo danificado
ϕ^i_{int}	Deslocamento modal do modelo intacto no modo *n* e GDL *i*
ϕ^i_{dan}	Deslocamento modal do modelo danificado no modo *n* e GDL *i*
$K_{localex}$	Matriz de rigidez local para um elemento exterior
$K_{localin}$	Matriz de rigidez local para um elemento interior
EA	Rigidez axial
K^A	Forma da matriz para um elemento tipo A
K^B	Forma da matriz para um elemento tipo B
K^C	Forma da matriz para um elemento tipo C
K^D	Forma da matriz para um elemento tipo D
K^{GLOBAL}	Forma da matriz global para um modelo constituído pelos elementos A, B, C e D
GDL_{ex}	Matriz GDL para um elemento exterior

GDL_{in} Matriz GDL para um elemento interior

GDL Matriz GDL

$\hat{\mathbf{K}}$ Matriz de rigidez condensada

K_{tt} Termos da matriz de rigidez global correspondentes aos GDL de translação

K_{rt}^{T} Matriz transposta dos termos da matriz de rigidez global correspondentes à relação entre os GDL de translação e de rotação

K_{rr}^{-1} Matriz inversa dos termos da matriz de rigidez global correspondentes aos GDL de rotação

K_{rt} Termos da matriz de rigidez global correspondentes à relação entre os GDL de translação e rotação

D Matriz dinâmica

K Matriz de rigidez condensada

M Matriz de massa

n Curvatura associada ao vector próprio nominal

d Curvatura associada ao vector próprio danificado

i Número de elementos

MV Modo de vibração

fn_n Frequência nominal

fn_d Frequência danificada

Capítulo 1

Introdução

1.1 Considerações Gerais

Os avanços da engenharia nas últimas décadas foram, inevitavelmente, acompanhados pela crescente necessidade de desenvolver sistemas de monitorização de desempenho, que assegurem o bom funcionamento das estruturas durante o seu período de vida útil. Por norma, os sistemas estruturais vigentes são concebidos para lidarem com diversos tipos de solicitações estáticas e dinâmicas, permanentes ou variáveis, que, a seu tempo, podem danificar elementos da estrutura. A existência de falhas ou danos localizados leva à redução da rigidez da estrutura o que, consequentemente, se traduz em modificações nos seus parâmetros modais, como as frequências próprias, modos de vibração e amortecimento.

O conhecimento de que as alterações físicas da estrutura conduzem a alterações das suas características de vibração foi o elemento percursor no desenvolvimento de várias técnicas que avaliam estas alterações. Idealmente um bom método de detecção de dano é o que é capaz de identificar rapidamente a sua ocorrência, localização, estimar a sua gravidade e prever a restante vida útil da estrutura. Este método deve ser também facilmente ajustável a processos de automatização e possuir um sistema de funcionamento não susceptível à subjetividade do utilizador. Outra das características importantes dos métodos de identificação de dano é a sua capacidade de fazer uso da informação proveniente de modelos de elementos finitos da estrutura, previamente definidos, para identificar alterações nas características da estrutura real [17].

Estes métodos, que assumem um comportamento linear, funcionam essencialmente através da análise das alterações das características das estruturas, nomeadamente as frequências naturais, parâmetros modais, curvaturas ou flexibilidade. Apesar das suas potencialidades estes apresentam algumas limitações, como a grande dependência de um ambiente controlado na obtenção de bons resultados, a sua falta de sensibilidade na avaliação de certos níveis de dano, o uso de modelos de referência e a filtragem de dados. A admissão de um comportamento linear também pode representar um problema no uso destes métodos para níveis de dano mais significativos, uma vez que a estrutura pode apresentar um comportamento não-linear.

Mais recentemente, surgiram novas técnicas, como os indicadores de anormalidades, que funcionam através de análises estatísticas e reconhecimento de padrões e pretendem colmatar alguns dos problemas dos métodos já existentes. Algumas formulações híbridas, aliando, por exemplo, a capacidade dos

métodos baseados nas alterações nas curvaturas com a das redes neuronais, foram sugeridas como processos de optimização na identificação de dano [19].

No processo de identificação de anomalias, a escolha do método a usar dependerá da tipologia da estrutura, da facilidade de acesso aos elementos, da gama de frequências esperada e do tipo de equipamento disponível. A obtenção da resposta da estrutura é feita com recurso a transdutores de vibração, que são equipamentos que permitem converter deslocamentos, velocidades ou acelerações em sinais eléctricos.

Neste documento serão analisadas algumas destas técnicas de análise modal, como a comparação de frequências, comparação dos modos de vibração e respectivas derivadas e análise de alterações dos parâmetros modais. No que refere às derivadas dos modos de vibração, serão analisadas em maior pormenor as modificações nas curvaturas. Nas técnicas baseadas nos parâmetros modais serão abordados dois indicadores estatísticos: o *Modal Assurance Criterion (MAC)*, que indica a correlação global entre dois modos de vibração e o *Coordinate Modal Assurance Criterion (COMAC)*, que indica a correlação local entre dois modos de vibração. Estes métodos serão utilizados na avaliação de modelos tipo definidos no programa de elementos finitos *SAP2000*, em que os casos de estudo são uma viga bi-apoiada, uma viga em consola e um pórtico bi-encastrado.

Será ainda desenvolvida neste trabalho uma plataforma *Java* que permitirá aos usuários da página da faculdade conhecer e utilizar o método de identificação de dano baseado na alteração das curvaturas.

1.2 Motivações

A realização deste trabalho é motivada pela importância que existe no conhecimento e aplicação de procedimentos não destrutivos na identificação de danos e manutenção da integridade estrutural. A sua relevância reflete-se no facto das técnicas aqui abordadas cada vez mais se revelarem uma solução eficaz do ponto de vista económico e operacional, pela facilidade de aplicação e baixos custos associados. Paralelamente ao estudo dos métodos de análise modal, o desenvolvimento de uma aplicação *web* que ilustra as qualidades do método das curvaturas também constitui um fator de motivação.

1.3 Contributo Inovador

O principal contributo inovador deste trabalho é feito através do desenvolvimento de uma *Java Applet*, aplicação no formato *bytecode Java*, que constitui uma ferramenta de cálculo que permite de uma forma simples explorar as potencialidades do método das curvaturas modais na identificação de danos em várias estruturas. Esta aplicação estará disponível para utilização no *website* da faculdade.

1.4 Organização da Dissertação

Este documento está organizado em 6 capítulos distintos. Os assuntos abordados em cada um deles encontram-se seguidamente sumarizados.

No presente capítulo é feito um enquadramento geral do trabalho, onde são expostos os seus objetivos principais, as razões pelas quais foi desenvolvido e o seu contributo inovador.

No segundo capítulo é feito um levantamento bibliográfico na área da monitorização e identificação de dano estrutural. São abordados os principais métodos de identificação de dano e sucintamente descritas as suas características.

No terceiro capítulo, primeira secção, é apresentado em maior pormenor o método das curvaturas, onde é feita uma descrição detalhada das suas metodologias e características. Na segunda secção são apresentados os casos de estudo onde este foi aplicado. Na secção seguinte são feitos estudos de sensibilidade ao método das curvaturas e a métodos que fazem uso de frequências naturais e indicadores estatísticos no processo de identificação.

No quarto capítulo são apresentadas as conclusões relativas aos resultados obtidos da aplicação dos métodos do capítulo anterior. É também feita, com base nos resultados, a sugestão de desenvolvimento de uma aplicação em *Java*.

O quinto capítulo apresenta os conceitos e metodologias inerentes à concepção do aplicativo *Web* de cálculo das curvaturas e está dividido em dois subcapítulos. O primeiro constitui um manual teórico, onde é explicada a forma como foi desenvolvida a aplicação e o segundo desempenha o papel de manual com instruções de utilização.

O sexto e último capítulo deste documento apresenta as conclusões retiradas ao longo da realização do trabalho, analisado o desempenho, fiabilidade, vantagens e desvantagens do método das curvaturas e do programa de cálculo. São ainda debatidas propostas para desenvolvimentos futuros.

Capítulo 2

Estado de Arte

O desenvolvimento de técnicas de caracterização e avaliação do desempenho e fiabilidade de estruturas metálicas e de betão armado durante o seu ciclo de vida em serviço constitui, do ponto de vista da engenharia, um papel fundamental na manutenção de recursos humanos e económicos. Nos últimos anos foram feitos vários avanços no desenvolvimento de técnicas de ensaio, tanto de carácter destrutivo como não destrutivo, essenciais no diagnóstico estrutural.

As técnicas de ensaio não destrutivas (TND) são particularmente adequadas para aplicação in situ durante a inspeção das estruturas, mas, na maioria dos casos, a aplicação complementar de técnicas de ensaio destrutivas (TD), a realizar em laboratório em amostras extraídas das estruturas, é imprescindível na identificação e quantificação dos mecanismos de deteorização estrutural [40]. A avaliação do desempenho estrutural por parte das técnicas não destrutivas requer a satisfação de requisitos como a detecção de defeitos/danos ou variação de propriedades nos elementos estruturais e estabelecer uma hierarquia e quantificação dos defeitos ou características analisadas para, à posteriori, serem comparadas com valores limites ou de referência.

Neste capítulo, é sumariamente apresentada uma revisão bibliográfica dos métodos de identificação de dano e monitorização estrutural baseados em alterações das propriedades dinâmicas dos elementos, referindo as suas características e aplicabilidade. A informação relativa à aplicação dos vários métodos a diferentes problemas de engenharia encontra-se disposta em subcapítulos de acordo com o tipo de técnica de diagnóstico utilizada.

A existência de dano estrutural num sistema de engenharia leva à modificação dos seus modos de vibração. Estas modificações são manifestadas através de alterações dos parâmetros modais (frequências naturais, modos de vibração e amortecimento) que podem ser obtidas através de testes dinâmicos de vibração. Importa referir que as alterações nos parâmetros modais podem não ser as mesmas para cada modo de vibração, uma vez que estas mudanças dependem da natureza, localização e severidade do dano [45].

Os parâmetros modais podem ser facilmente obtidos pela medição das respostas de vibração. As respostas são obtidas através de um tipo de transdutor que monitoriza a resposta estrutural a solicitações artificialmente introduzidas ou a solicitações ambientais existentes em serviço. A indução de níveis baixos de energia é suficiente para produzir respostas mensuráveis desde que esta energia seja dinamicamente amplificada. Há casos em que a necessidade de uma excitação artificial se pode traduzir em custos relativamente elevados. A realização destes testes de diagnóstico, quando feita coincidir

com outras inspeções programadas, oferece a possibilidade de monitorizar eventuais alterações das condições estruturais ao longo do tempo. Uma outra vantagem na medição das respostas de vibração é a possibilidade de estudar as curvaturas através das derivadas dos deslocamentos modais.

Em 1993, Rytter propôs um sistema de classificação de métodos de identificação de dano que vem definido em quatro níveis diferentes de identificação [17]:

- Nível 1: Determinação da existência de dano estrutural;
- Nível 2: Nível 1 + determinação da localização do dano;
- Nível 3: Nível 1 + quantificação do dano;
- Nível 4: Previsão da vida residual da estrutura.

Neste documento serão abordados predominantemente os primeiros três níveis de classificação uma vez estão directamente relacionados com a problemática da dinâmica estrutural.

2.1 Métodos de Deteção e Localização de Dano

2.1.1 Métodos Baseados nas Alterações das Frequências Naturais

A existência de alterações nas propriedades físicas de uma estrutura pode conduzir a alterações na sua frequência natural uma vez que esta varia proporcionalmente com a raiz quadrada do quociente da rigidez pela massa. É então evidente que são necessárias variações de rigidez significativas para haver alterações significativas na frequência. Os primeiros métodos conduzidos nesta linha da monitorização estrutural foram os que fazem uso das alterações na frequência de vibração uma vez que as medições das frequências podem ser conseguidas rapidamente e com alguma fiabilidade. O dano estrutural traduz-se numa perda de rigidez da estrutura que faz com que haja uma diminuição das frequências naturais medidas. Caso o contrário se verifique, ou seja, se a frequência de vibração for maior do que o esperado, pode ser devido à existência de um sistema de apoio mais rígido do que o expectável. Segundo Creed S.G. é necessária uma alteração de no mínimo 5% na frequência natural para que o dano possa ser detectado por este método [15]. No entanto, alterações significativas nas frequências só por si não implicam necessariamente a existência de dano uma vez que estas sofrem oscilações da mesma magnitude devidas a mudanças nas condições ambientais.

Estes métodos têm limitações práticas na identificação das alterações de frequência. O facto de terem necessariamente de existir danos grandes para provocarem uma variação expressiva na frequência faz com que este método necessite de fazer medições muito precisas para garantir resultados. Esta exigência de precisão aliada às variações induzidas pelas mudanças climatéricas condiciona muito a utilização do método. Esta dificuldade foi constatada por exemplo por Coppolino quando percebeu que as alterações de frequência induzidas em plataformas *offshore* eram difíceis de distinguir das alterações proporcionadas pelas oscilações do mar [13]. Muitos dos testes conduzidos nesta área provaram que a utilização destes métodos de detecção de dano é mais eficiente quando feitos em ambientes controlados, onde o controlo de qualidade na concepção dos elementos é maior e por conseguinte as medições são mais precisas.

Outra limitação da análise comparativa de frequências advém do facto da frequência natural ser uma propriedade global da estrutura, significando isto que o método permite inferir sobre a existência de dano mas não fornece nenhum tipo de informação espacial da estrutura, ou seja, não permite conhecer a sua localização. Ainda assim, os modos de vibração mais altos podem constituir uma excepção a esta limitação uma vez que estão associados a respostas locais.

Nos pontos onde o deslocamento modal é zero os esforços são mínimos para o modo de vibração em questão. A existência de uma alteração da frequência menor que a esperada, pode significar que o dano se encontra perto de um dos nós modais, o que, pelas dificuldades inerentes à análise, pode ser uma razão para as medições de frequência não serem fiáveis quando o dano está localizado em zonas de baixas tensões. Testes realizados em elementos de betão, permitiram concluir que o grau de redução da frequência natural era dependente da posição do defeito relativamente ao modo de vibração em questão. Danos onde as tensões são elevadas podem constituir reduções de cerca de 15% das frequências de vibração. A redução da frequência torna-se mais importante quando o dano se encontra em zonas de grande curvatura para o modo de vibração em causa, uma das questões abordadas no capítulo 2.1.3.

A existência de uma fenda na secção de uma viga é equivalente a uma redução do momento de inércia, proporcional à severidade da fenda, o que leva a uma redução na rigidez de flexão local nessa mesma secção. Com base neste conceito vários modelos foram definidos para representar a viga modificada, como o de Chondras e Dimarogonas [9] onde o dano é representado por uma mola de torção que faz a ligação com as duas vigas adjacentes, ou o modelo *fracture hinge*, desenvolvido por Ju e Mimovich [22] que faz uso do conceito de dobradiça.

Com o objectivo de contornar as limitações da análise estrutural através das alterações nas frequências naturais, que podem por em causa a sua fiabilidade, têm sido apresentados vários métodos que pretendem minimizar estas limitações. Seguidamente encontram-se descritos por traços gerais alguns destes métodos.

Em 1979, foram desenvolvidos os primeiros trabalhos por Cawley e Adams [6] que assentavam na ideia de que, para localizar o defeito, teriam de ser calculadas as variações das frequências naturais devidas a danos em posições escolhidas da estrutura e comparadas com os valores da estrutura não danificada. Este método, à semelhança da maioria dos métodos baseados na sensibilidade, não permite identificar o local do dano e exige um grande esforço computacional. A aplicação deste método para identificar zonas danificadas no betão armado e vigas de aço revelou que pelo menos nove modos de vibração deveriam estar envolvidos nos cálculos para o dano ser identificado com precisão razoável. A precisão dos métodos baseados em análise de sensibilidade é dependente da qualidade do modelo de elementos finitos usado para o cálculo da sensibilidade e, como referido anteriormente, são mais úteis em estruturas onde os danos afetam uma parcela importante da rigidez da estrutura.

Em 1990, Law S. S. [30] sugeriu uma técnica que faz uso de um modelo de elementos finitos da estrutura e uma matriz de massa assimétrica que possibilite uma identificação única. A solução do problema envolve o recurso a um método não-linear que permite, através destas matrizes e de vectores que delimitam valores limites, a identificação de alterações nos parâmetros modais. A correta utilização deste método requer o conhecimento prévio da zona da estrutura onde se encontram os danos.

Em 1992, Zhang [54] propôs um método de reconhecimento de padrões de dano em estruturas reticuladas. A mesma abordagem foi também aplicada para diagnosticar defeitos em pilares de fundação. O método baseia-se no facto de que a variação relativa na frequência natural entre quaisquer dois modos

é igual à relação dos quadrados dos correspondentes valores modais de deformação na falha. A precisão do método é altamente dependente da escolha dos parâmetros de controlo e só é aplicável a falhas que possam ser representadas como fendas.

Em 1993, Uzgider [52] propôs um método de localização de danos que usa medições de frequências para identificar parâmetros da rigidez. Os modos de vibração com maior influência são selecionados primeiro e são usadas as frequências naturais dos modos selecionados para identificar as alterações nos parâmetros da rigidez. As magnitudes relativas das diferenças entre as primeiras estimativas e os parâmetros identificados são o indicador da presença e localização de dano estrutural. O bom funcionamento do método é condicionado pela necessidade de um modelo matemático sofisticado e pela necessidade de definir corretamente tanto os parâmetros da rigidez como os limites superiores e inferiores da frequência.

Cerri e Vestroni [7], em 1999, sugeriram dois procedimentos diferentes para a identificação de dano, um deles o Método do Erro na Equação Característica e o outro através de comparações entre valores analíticos e experimentais de frequências. O dano era definido por três parâmetros (posição, grau e extensão) os quais seriam identificados no problema inverso, tipicamente nos níveis 2 e 3 de identificação.

Outros resultados de testes dinâmicos, tanto em modelos reduzidos como em modelos reais, indicam que as mudanças nas frequências naturais podem ser devidas a deficiências em zonas de apoio, a propagação de fendas, falhas nas tensões de corte ou sobrecargas que podem causar danos internos.

2.1.2 Métodos Baseados nas Alterações dos Modos de Vibração

Em 1984, West apresentou um dos primeiros meios de localização de dano através da comparação direta dos modos de vibração sem necessitar de um modelo de elementos finitos. Este método faz uso de um indicador estatístico chamado *MAC (Modal Assurance Criterion*), proposto por Allemang e Brown em 1982, que testa a ortogonalidade entre dois modos de vibração e avalia a sua relação a nível global. Os vectores dos deslocamentos modais, analíticos e experimentais, são escalados para que os termos diagonais da matriz, que relacionam diretamente cada modo de vibração, sejam unitários. Portanto, este índice vai variar entre 0 e 1, onde 0 significa que não há relação entre os modos e 1 representa uma correlação perfeita. Com esta forma de escala, espera-se que os valores fora da diagonal da matriz de massa modal sejam inferiores a 0,1. Este método, apesar do contributo inovador na validação de modelos experimentais e detecção de dano, também tem várias limitações[53, 2]. Acontece que muitas formulações alternativas que fazem uso do *MAC* foram desenvolvidas para contornar as limitações da formulação original e, grande parte das vezes, os resultados não são os mais satisfatórios devido à má utilização deste critério. Os problemas do uso deste método podem ser devidos ao facto de a massa e a rigidez não serem introduzidos na formulação, ou ainda, quando feitas medições *in situ*, por norma não são suficientes para estudar corretamente alguns modos de vibração.

Lieven e Ewins [31], em 1988, definem um novo critério que segue os princípios do *MAC*, chamado *COMAC (Coordinate Modal Assurance Criterion)*. Este critério possui um carácter local e permite estabelecer a concordância pontual entre vários modos de vibração das estruturas com e sem dano. Por outras palavras, ao passo que o *MAC* permitia uma comparação global entre os valores da estrutura com dano e os valores da estrutura sem dano, o *COMAC* permite comparar os valores modais em cada ponto da estrutura. Tal como no *MAC*, este critério usa um índice que varia entre 0 e 1, onde 0 indica que não há relação entre os valores e 1 que há concordância total.

Vários estudos conduzidos na tentativa de aliar as potencialidades destes dois métodos. Kim et al [26] investigou o uso conjunto do *Partial MAC* e do *COMAC* para isolar a área danificada da estrutura. Ko et al [29] apresentou um método que usa uma combinação do *MAC, COMAC* e análise de sensibilidade para detectar danos em estruturas moldadas em aço. A sensibilidade das derivadas dos modos de vibração a danos localizados foi usada para determinar quais os graus de liberdade mais relevantes e posteriormente usado o índice *COMAC* como indicador de dano. Salawu e Williams [38] mostraram que os valores do *MAC* podiam ser usados como indicador dos modos de vibração mais afetados pelo dano.

2.1.3 Métodos Baseados em Alterações nas Curvaturas

Uma outra maneira de obter informação sobre o local do dano através alterações da vibração é usando as derivadas dos modos de vibração, como a curvatura. Para estruturas como vigas, cascas e placas há uma relação direta de proporção entre curvatura e momento flector.

Em 1991, Pandey *et al* [36] demonstrou que as diferenças absolutas entre as curvaturas de estruturas modeladas num programa de elementos finitos podiam ser um bom indicador da localização de dano. Estas curvaturas podem ser obtidas com recurso à derivada dos deslocamentos modais obtidos do modelo de elementos finitos (MEF) através de um operador de diferenças finitas centradas. O valor absoluto da diferença das curvaturas das estruturas com e sem dano deve ser máximo, para cada modo de vibração, na zona danificada. O local onde esta diferença é máxima corresponde assim à zona danificada. Correspondendo este dano a uma perda de rigidez, quanto maior for a perda de rigidez maior será o nível de dano e, portanto, maior a variação na curvatura. Foi demonstrado que há alguma dificuldade na medição dos graus de liberdade da estrutura, principalmente os de rotação, devido a limitações dos equipamentos, o que se pode traduzir em dificuldades na aplicação do método.

Stubbs *et al* [41, 46], em 1992, apresentou um método de carácter semelhante ao de Pandey, mas este era baseado na comparação da energia de deformação em cada grau de liberdade. Uma diminuição de energia assinalável corresponderia a um dano neste local. Em 1995, num trabalho conjunto com Topole [48], examinou a viabilidade do uso de um conjunto limitado de parâmetros modais para a detecção de danos estruturais. Em 1996, juntamente com Kim [49], examinou a fiabilidade deste método mas desta vez sem ter por base os parâmetros modais.

Chance *et al* [8], em 1994, descobriu que o cálculo numérico a partir das curvaturas podia conduzir a erros graves por as medições feitas diretamente serem as deformações e não as curvaturas, o que se traduzia em resultados aparentemente satisfatórios mas que podiam conter erros.

Salawu e Williams [39], também em 1994. utilizaram medições de modos de curvatura com recurso às diferenças finitas centradas. Eles compararam o desempenho deste método de diferenças relativas com um método de diferenças relativas nos modos de vibração. Demonstraram com isto que, tipicamente, usando dados experimentais, o método de identificação de dano por comparação das diferenças relativas das curvaturas não conduz a bons resultados, apontando ainda que a escolha dos modos a usar era preponderante no uso destes métodos.

2.1.4 Métodos Baseados na Atualização de Matrizes

Estes métodos que desenvolvem a sua linha de acção com recurso à matriz de flexibilidade também constituem um meio de avaliar alterações no comportamento estático estrutural. Uma vez que a matriz de flexibilidade é o inverso da matriz de rigidez, a matriz de flexibilidade relaciona também ela a força com o deslocamento resultante. Em consequência da relação entre rigidez e flexibilidade, sabe-se que se a rigidez diminui na presença de dano, então significa que a flexibilidade terá de aumentar.

Proposto em 1990 por Lim [32], o *Unity Check Method* foi desenvolvido para localizar erros na modelação e baseia-se na relação inversa entre a matriz de flexibilidade e a matriz de rigidez, onde é definida uma matriz de erro que mede o grau de correlação entre as medições.

Doebling *et al* [16], em 1995, apresentou um método que recorre ao conceito do efeito da flexibilidade residual, onde é calculada uma matriz que compreende a contribuição de todos os modos de vibração, fora da gama de medições habituais, para que a matriz de flexibilidade exata possa ser definida. Os autores demonstraram que a inclusão das medições de flexibilidade residuais na matriz de flexibilidade permite estimativas de erros mais precisas.

A maioria dos métodos até agora citados fazia uso das frequências naturais, variações de rigidez e alterações nas propriedades dos modos de vibração para identificar anomalias. Paralelamente a estes, foram desenvolvidos outros métodos baseados na modificação da estrutura dos modelos matriciais da massa, rigidez e amortecimento de forma a reproduzir da maneira mais fidedigna possível os dados das medições estáticas ou dinâmicas de campo. Estes métodos procuram um modelo de optimização que garanta a atualização de matrizes através da equação do movimento, do modelo nominal e dos deslocamentos medidos. As variações nas matrizes atualizadas constituem uma ferramenta que permite detectar e localizar dano. Estes métodos usam um conjunto básico de equações, onde as variações nos algoritmos das mesmas dependem da função objectivo a ser minimizada, das restrições existentes no problema e do esquema numérico usado para implementar a optimização.

No âmbito das três variantes das equações referidas foram vários os autores a desenvolverem trabalhos na área. Refira-se como exemplo os trabalhos de Kabe em 1985 [23], Zimmerman e Kaouk em 1994 [25, 24], Kim e Bartkowicz em 1994 [26] e Schulz *et al* em 1996 [42].

2.1.5 Métodos Não-lineares

A avaliação das consequências de danos estruturais pode ter carácter linear ou não-linear. Por norma é feita uma análise linear ao comportamento das estruturas, onde apenas são contemplados os fenómenos que ocorrem no domínio elástico linear. No decorrer de anomalias mais significativas, como fissuras por fadiga devidas a ciclos de carga/descarga ou plastificação dos elementos, a estrutura pode apresentar um comportamento não-linear.

Lin e Ewins, em 1990, propuseram um processo para localizar não-linearidades estruturais usando testes modais. A base deste processo é uma técnica de actualização do modelo com dados modais, onde medições são feitas em diferentes níveis de resposta de forma a identificar a não-linearidade [33].

Klein *et al*, em 1994, observou experimentalmente, através da medição da função de resposta de frequência numa viga em consola fendilhada por fadiga, que as frequências naturais aumentavam consideravelmente à medida que a localização do acelerómetro se aproxima da fenda [27].

Vários esforços foram desenvolvidos no âmbito dos métodos não-lineares, sendo exemplos disso os trabalhos de Manson *et al* (1993) [34], Feldman and Braun (1995) [18] e Prime and Shevitz (1996) [37].

2.1.6 Métodos Baseados em Indicadores de Anormalidades

Mais recentemente os indicadores de anormalidades têm permitido uma análise estrutural diferente da usualmente praticada pelos métodos que fazem uso das características dos modos de vibração e dos parâmetros modais. Estes últimos provaram algumas dificuldades como, por exemplo, a falta de sensibilidade na avaliação de determinados níveis de dano, o inevitável uso de um modelo de referência, a influência das condições ambientais e a assunção de um comportamento mecânico linear. A somar a isto, o processo de identificação modal é um processo de filtragem de informação que pode condicionar a correta avaliação do estado da estrutura.

Os indicadores de anormalidades representam novas abordagens assentes em análises estatísticas e reconhecimento de padrões. Uma destas abordagens é baseada no conceito de fusão de dados, que foi desenvolvido nos anos 50 para ser utilizado como ferramenta ao serviço do exército norte-americano, e consiste em cruzar informação de forma a reduzir incertezas. Esta metodologia foi abordada por Guo [20], em 2006, e Minor *et al* [35], em 2007, na detecção de danos estruturais, numa tentativa de reunir e cruzar a informação proveniente de vários acelerómetros. Chun *et al* [10], em 2005, e Su *et al* [47], em 2009, também abordaram este método propondo alternativas na gestão de dados e admitindo as dificuldades inerentes ao facto do método lidar com muita informação.

Outra abordagem possível são os métodos baseados no reconhecimento de padrões. Estes métodos permitem identificar variações do comportamento estrutural previsto e em função disto localizar danos. As redes neuronais artificiais, por exemplo, através de modelos matemáticos conseguem treinar uma rede de classificação de padrões que permite identificar anomalias nas estruturas.

Capítulo 3

Método das Curvaturas

Neste capítulo é feita uma descrição do método usado para detecção e localização de dano, apresentados os modelos analíticos que servem à aplicação do método e posteriormente apresentados vários estudos de sensibilidade que pretendem aferir a validade deste.

3.1 Descrição do Método

As alterações na frequência permitem detectar facilmente a presença de dano mas determinar a sua localização através destas alterações é um problema completamente diferente. A existência de danos em duas localizações distintas podem conduzir exatamente à mesma variação na frequência, sendo preciso portanto determinar outros parâmetros que permitam identificar a zona que se encontra danificada[36].

Um dos métodos que faz uso das características de vibração das estruturas para obtenção de informação sobre a presença e localização de dano e que ainda constitui uma ferramenta útil na sua quantificação é o método das curvaturas. Este método faz uso das derivadas dos modos de vibração, nomeadamente a curvatura. A sua metodologia recorre ao conhecimento da relação de proporção direta que existe entre a curvatura e o momento flector para estruturas como vigas, cascas e placas. A diferença entre as curvaturas do modelo intacto e do modelo danificado é utilizada para detectar a localização do dano estrutural, sendo esperado que as diferenças mais significativas se encontrem na região onde há danos.

A definição de curvatura dos elementos é fundamental para compreender os conceitos subjacentes a este método. A figura 3.1 representa uma secção de um elemento tipo viga e mostra que para a hipótese dos pequenos deslocamentos, a deformação axial, ε, de uma secção de comprimento unitário pode ser expressa pela relação y/R. De forma equivalente, a geometria do problema mostra que o ângulo decorrente da flexão do elemento é obtido através da largura unitária e do raio de curvatura, R, através da relação 1/R, o que constitui a definição de curvatura. É então possível estabelecer as relações seguidamente apresentadas.

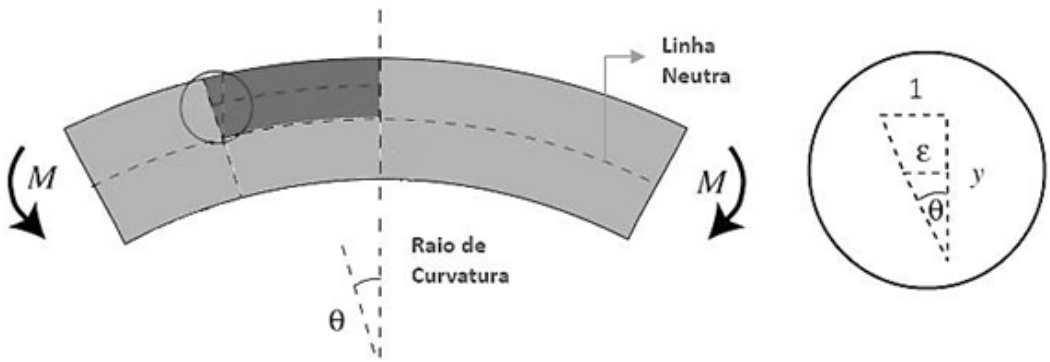


Figura 3.1: Relações decorrentes de um elemento tipo em flexão.

$$\theta = \frac{1}{R} = \frac{\varepsilon}{y} \tag{3.1}$$

$$\varepsilon = \frac{y}{R} = k \times y \tag{3.2}$$

Em que,

ε é a deformação axial;
R é o raio de curvatura;
θ é o ângulo de flexão;
y é a distância da linha neutra a uma fibra da viga;
k é a curvatura.

Estas considerações só são possíveis se respeitada a Hipótese de Bernoulli da teoria de vigas à flexão. Nestas condições, os segmentos inicialmente lineares e perpendiculares ao eixo da viga permanecem lineares e perpendiculares ao eixo quando esta é flectida, o comportamento da viga é elástico linear, a secção transversal tem de ser simétrica, o ângulo de rotação é muito pequeno e o seu material constituinte é homogéneo.

Considerando ainda as mesmas hipóteses, a tensão axial, σ, a uma distância y da linha neutra da viga é função das propriedades do material que a constitui e portanto é dada por:

$$\sigma = E \times \varepsilon = E \times k \times y \tag{3.3}$$

Em que,

E é o módulo de elasticidade do elemento.

O momento flector, M, que representa o esforço desenvolvido na viga quando esta é flectida, pode ser expresso pela relação:

$$M = \int y \times (E \times k \times y) dA = kE \int y^2 dA \tag{3.4}$$

Se considerada a definição de momento de inércia:

$$I = \int y^2 dA \tag{3.5}$$

Simplificando a expressão do momento flector, esta pode ser escrita como:

$$M = k \times E \times I \tag{3.6}$$

Finalmente é possível definir a expressão da curvatura expressa em função do momento flector e da rigidez de flexão da secção. Esta prova então a relação direta entre a curvatura e a rigidez:

$$k = \phi_x^{''} = -\frac{M_x}{EI} \tag{3.7}$$

Em que,

$\phi^{''}$ representa a curvatura da secção;
M representa o momento flector na secção;
E representa o módulo de elasticidade;
I representa o segundo momento de inércia da área da secção.

No capítulo anterior, na secção 2.1.3., foram apresentados trabalhos desenvolvidos por vários autores no âmbito dos métodos que recorrem às derivadas dos modos de vibração. Nesta dissertação será destacado o trabalho desenvolvido por Pandey respeitante às curvaturas dos elementos. Como foi referido, Pandey demonstrou que as diferenças absolutas entre as curvaturas de estruturas modeladas num programa de elementos finitos podiam ser um bom indicador da localização de dano. Estas curvaturas podiam então ser obtidas com recurso à segunda derivada dos deslocamentos modais obtidos no modelo de elementos finitos *(MEF)* através de um operador de diferença finitas centradas *(DFC)*. Este operador é definido como:

$$\phi_q^{''} = \frac{\phi_{q-1} - 2\phi_q + \phi_{q+1}}{h^2} \tag{3.8}$$

Em que,

$\phi_{q,i}^{''}$ é o valor da curvatura no ponto *q*;
$\phi_{q-1,i}$ é o valor do deslocamento no ponto *q - 1*;
$\phi_{q,i}$ é o valor do deslocamento no ponto q;
$\phi_{q+1,i}$ é o valor do deslocamento no ponto *q + 1*;
h é o comprimento de cada elemento.

Para cada um dos modos de vibração, o valor absoluto da diferença das curvaturas das estruturas com e sem dano deve ser máximo na zona danificada. Correspondendo este dano a uma perda de rigidez (EI), quanto maior for esta perda maior será o nível de dano e, portanto, maior a variação na curvatura. Foi

demonstrado que há alguma dificuldade na medição dos graus de liberdade da estrutura, principalmente os de rotação, devido a limitações dos equipamentos, o que se pode traduzir em dificuldades na aplicação do método.

As estruturas compostas de vigas ou placas têm como esforço principal o momento flector e são consideradas as hipóteses de Bernoulli para elementos finos. Portanto, este método requer alguns cuidados quando utilizado em estruturas como pórticos ou treliças. Este problema será abordado em maior pormenor no capitulo seguinte quando apresentado o caso de estudo do pórtico.

3.2 Casos de Estudo

A aplicação do método que pretende demonstrar esta característica da curvatura dos elementos foi feita em 3 modelos analíticos distintos, sendo estes uma viga bi-apoiada, uma viga em consola e um pórtico bi-encastrado. Estes modelos foram definidos no programa de cálculo automático *SAP2000* através de elementos *Frame*, também designados de elementos barra, foram admitidos como sendo de betão e possuindo uma secção transversal retangular e uniforme ao longo de todo o seu desenvolvimento. As figuras 3.2, 3.3 e 3.4 representam os modelos utilizados.

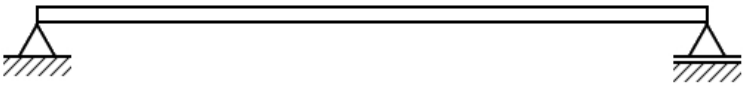


Figura 3.2: Modelo da viga bi-apoiada.

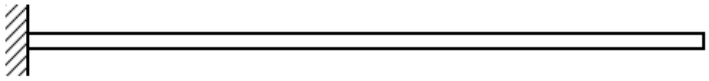


Figura 3.3: Modelo da viga em consola.

Figura 3.4: Modelo do pórtico.

Para definir estes modelos é necessário primeiro definir uma grelha e desenhar um elemento barra com o comprimento pretendido. Nesse estudo os modelos foram dimensionados com um comprimento de 6

metros de viga e, no caso do pórtico 2 metros de altura cada pilar. As secções transversais dos elementos nos 3 modelos são de 0,5 x 0,5 m.

Nesta fase é importante relembrar que são as variações de rigidez que provocam as modificações das curvaturas, e, portanto, a simulação de danos nos modelos analíticos será feita através da admissão de secções com módulos de elasticidade mais baixos que o valor característico do betão usado. Foi admitido um betão C30/37 com E=33 GPa. Uma vez que vai ser testada a sensibilidade do método, são definidos diferentes módulos de elasticidade correspondentes a vários graus de dano. A tabela seguinte apresenta os valores assumidos para cada material e as respectivas designações:

Tabela 3.1: Módulos de elasticidade introduzidos no *Define materials*.

Material	Peso Volúmico [kN/m*3*]	Módulo de Elasticidade	GPa
$Concrete1$	24	E	33
$Concrete2$	24	$0,9E$	29,7
$Concrete3$	24	$0,7E$	23,1
$Concrete4$	24	$0,5E$	16,5
$Concrete5$	24	$0,3E$	9,9
$Concrete6$	24	$0,1E$	3,3

Paralelamente ao procedimento anterior, a definição das propriedades da secção é feita com o mesmo critério definindo uma secção diferente para cada um dos materiais definidos.

As condições de apoio são definidas consoante o modelo em questão. Se o modelo em análise for a viga bi-apoiada basta bloquear as translações em ambas as extremidades, se for a viga em consola ou o pórtico são bloqueadas as translações e rotações nas zonas de apoio.

Uma vez que este método faz uso exclusivo dos deslocamentos transversais dos elementos, no separador *Analyze Options* são definidos os graus de liberdade pretendidos. Para os modelos da viga e consola são desbloqueados os deslocamentos no eixo Z e rotações em Y e para o pórtico os deslocamentos nos eixos X e Z e as rotações em Y. A figura 3.5 ilustra o referencial usado:

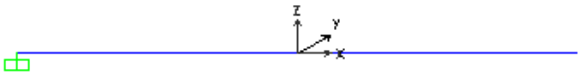


Figura 3.5: Referencial comum aos 3 modelos de elementos finitos.

Com vista a avaliar a sensibilidade do método são estudados casos com diferentes níveis de refinamento dos elementos. Importa sublinhar que a referência a um caso de estudo, implica necessariamente o uso de 2 modelos idênticos em que um está intacto e outro apresenta elementos danificados (com menor rigidez). Para o caso da viga e da consola, foram estudados 3 casos distintos:

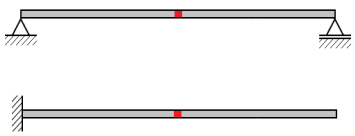
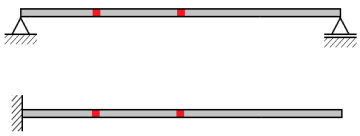
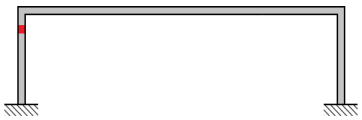
- $Caso1$: 10 elementos *Frame* (11 nós)
- $Caso2$: 20 elementos *Frame* (21 nós)
- $Caso3$: 50 elementos *Frame* (51 nós)

Feita a análise modal, as 5 primeiras frequências de vibração e os deslocamentos modais correspondentes são exportados para a ferramenta *Excel*.

De seguida são exportados os deslocamentos modais dos modelos de elementos finitos para a ferramenta de cálculo *Excel* onde serão obtidas e comparadas as curvaturas dos modelos intactos e danificados.

A tabela 3.2 sumariza os locais onde foram simulados os danos em cada modelo:

Tabela 3.2: Localização dos danos nos diferentes casos.

Viga e Consola		Pórtico
1 dano - 10, 20 e 50 elementos	2 danos - 20 e 50 elementos	1 dano - 60 elementos
		

3.3 Estudos de Sensibilidade

3.3.1 Método das Curvaturas

No capítulo 3.1 foram descritos os procedimentos necessários à obtenção das curvaturas dos modelos. Uma vez obtidos os deslocamentos modais no programa de elementos finitos para cada um dos casos de estudo, é possível, com recurso a um operador de diferenças finitas centradas, obter as curvaturas dos mesmos. Este operador faz uso do deslocamento modal de um nó, *q*, e dos dois nós que lhe são adjacentes, *q-1* e *q+1*, no processo de obtenção da curvatura do nó *q* em causa.

Na aplicação do método das diferenças finitas, as condições de fronteira permitem diminuir o número de variáveis no sistema de equações. Para aplicar um operador deste tipo aos nós de fronteira existem normas que permitem recorrer a pontos fora da viga/consola (nós artificiais da malha de diferenças finitas) e a pontos no seu interior para obter a curvatura nesse ponto.

No que se refere ao modelo da viga bi-apoiada, no nó em cima do apoio o deslocamento é nulo mas existe rotação. Existindo rotação o momento terá o seu valor igual a zero, o que resulta na equação 3.9.

$$\phi_{q+1} = -\phi_{q-1} \tag{3.9}$$

A figura 3.6 representa a aplicação do operador de diferenças finitas centradas, DFC, modelo da viga bi-apoiada na zona de fronteira.

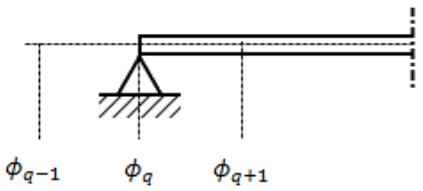


Figura 3.6: Condição de fronteira na viga bi-apoiada.

No modelo da viga em consola, tanto o deslocamento como a rotação são nulos na zona de encastramento. Aplicando estas condições de fronteira na primeira derivada, correspondente à rotação, obtém-se a equação 3.12.

$$\frac{d\phi}{dh} = \frac{\phi_{q+1} - \phi_{q-1}}{2h} \tag{3.10}$$

Quando,

$$\frac{d\phi}{dh} = 0 \tag{3.11}$$

Resulta,

$$\phi_{q+1} = \phi_{q-1} \tag{3.12}$$

A figura 3.7 representa a aplicação do operador de DFC modelo de viga em consola na zona de fronteira.

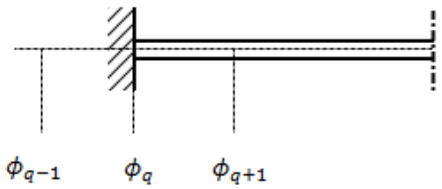


Figura 3.7: Condição de fronteira na viga em consola.

Ainda no caso da viga em consola, a sua extremidade livre permite que exista deslocamento vertical mas não existe momento flector. Neste nó o deslocamento será dado pela equação 3.14.

Quando,

$$\frac{d^2\phi}{dx^2} = 0 \tag{3.13}$$

Resulta,

$$\phi_{q+1} = 2\phi_q - \phi_{q-1} \tag{3.14}$$

A figura 3.7 representa a aplicação do operador de DFC na extremidade livre do modelo de viga em consola.

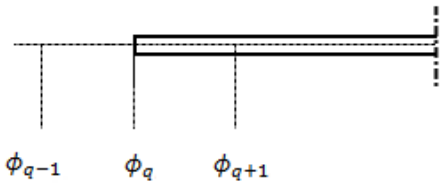


Figura 3.8: Condição de fronteira na extremidade livre.

A avaliação do método das curvaturas será feita com recurso a diferentes testes de análise comparativa ao nível dos diferentes modos de vibração, análise comparativa da influência de diferentes graus de dano e estudo da relação da localização do dano com o modo de vibração. Em seguida serão apresentados alguns exemplos que ilustram a capacidade deste método e permitem avaliar as características referidas

anteriormente. Com o intuito de clarificar a análise são apresentados nas tabelas 3.3, 3.4 e 3.5 os primeiros 5 modos de vibração de cada modelo.

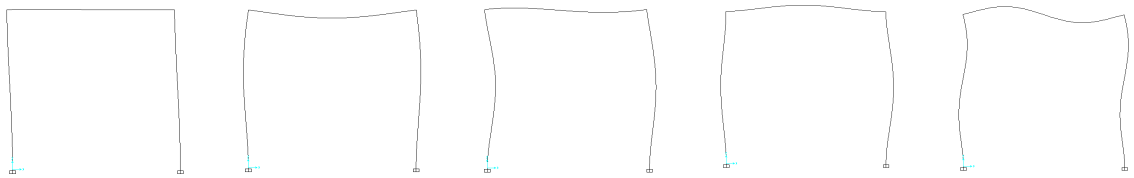
Tabela 3.3: Primeiros 5 modos de vibração da viga bi-apoiada.

Viga bi-apoiada				
1º modo	2º modo	3º modo	4º modo	5º modo
				

Tabela 3.4: Primeiros 5 modos de vibração da viga em consola.

Viga em consola				
1º modo	2º modo	3º modo	4º modo	5º modo
				

Tabela 3.5: Primeiros 5 modos de vibração do pórtico.

Pórtico				
1º modo	2º modo	3º modo	4º modo	5º modo
				

Primeiramente optou-se por fazer uma comparação que permita clarificar a diferença entre o uso de modelos com 10, 20 e 50 elementos *Frame*. Tomou-se como exemplo a curvatura do 1º modo do modelo da viga bi-apoiada com um dano de 50% a meio vão. Cada um dos gráficos 3.9, 3.10 e 3.11 apresenta a curvatura da viga intacta, a curvatura da viga danificada e a diferença em módulo entre estas.

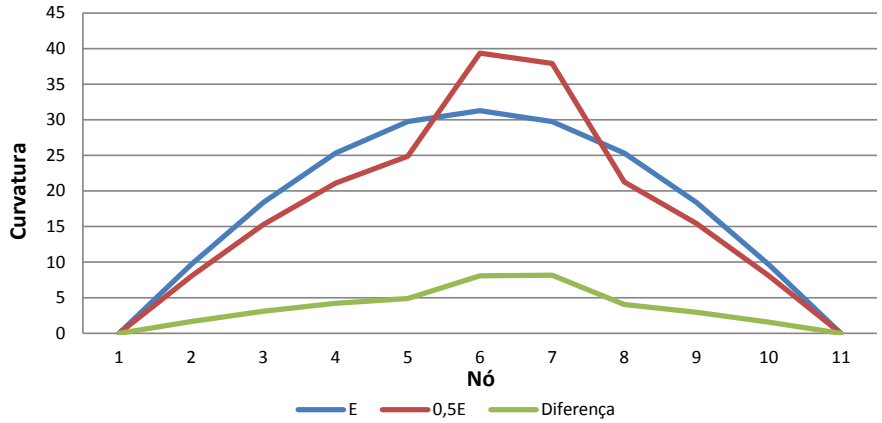


Figura 3.9: Viga com 10 elementos - Curvaturas do 1º modo com 50% de dano.

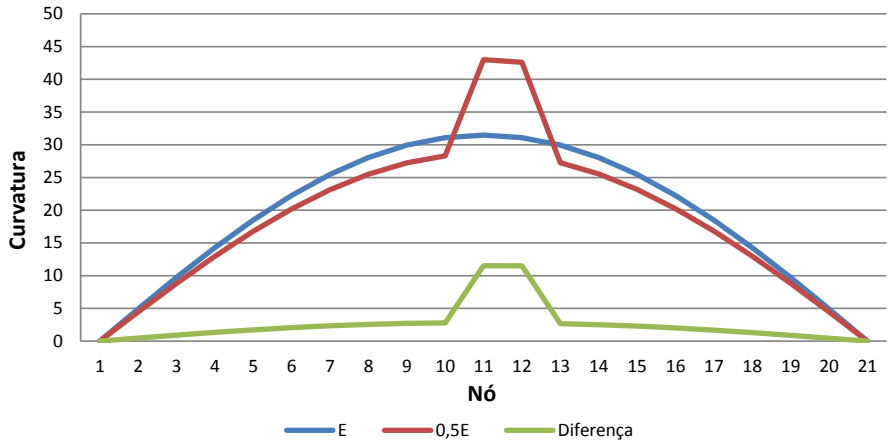


Figura 3.10: Viga com 20 elementos - Curvaturas do 1º modo com 50% de dano.

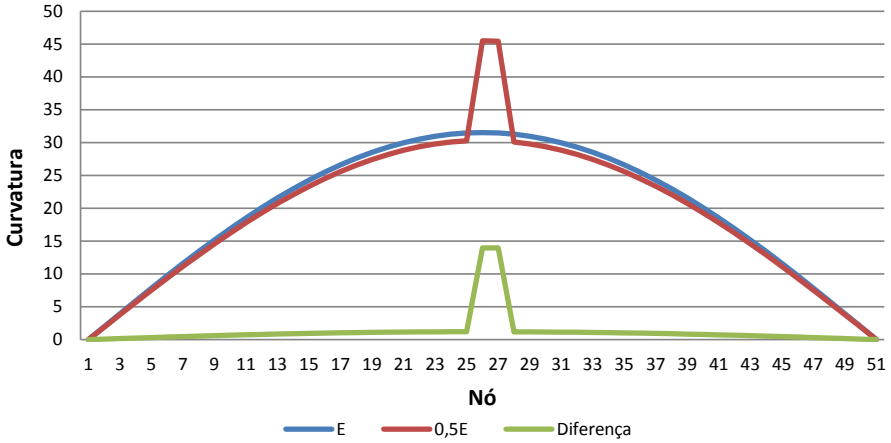


Figura 3.11: Viga com 50 elementos - Curvaturas do 1º modo com 50% de dano.

É possível constatar que o rigor aumenta com o aumento do refinamento da malha de elementos. No modelo com 10 elementos, apesar de poder identificar-se a zona onde há maiores diferenças entre as curvaturas, existem diferenças que ainda são significativas em zonas não danificadas. No modelo com 50 elementos é possível identificar o dano com evidência na zona de meio vão, sendo que as diferenças nas restantes zonas são residuais.

A identificação destes danos nos modelos da viga em consola e do pórtico foi conseguida da mesma maneira. Os gráficos 3.12 a 3.16 ilustram esta identificação, onde foi induzido um dano de 50% a meio vão no modelo da consola e um dano de 50% na zona superior do 1º pilar do pórtico. Para o modelo da consola são apresentados o 3º e 4º modos de vibração e para o pórtico apenas o 1º.

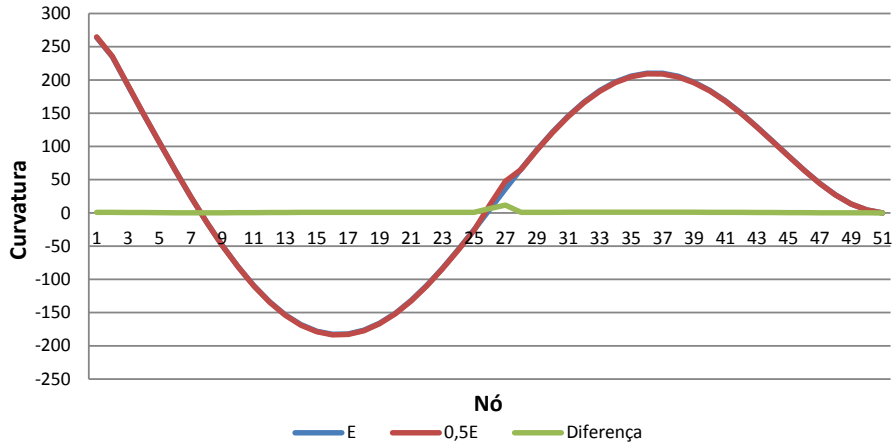


Figura 3.12: Consola com 50 elementos - Curvaturas do 3º modo com 50% de dano.

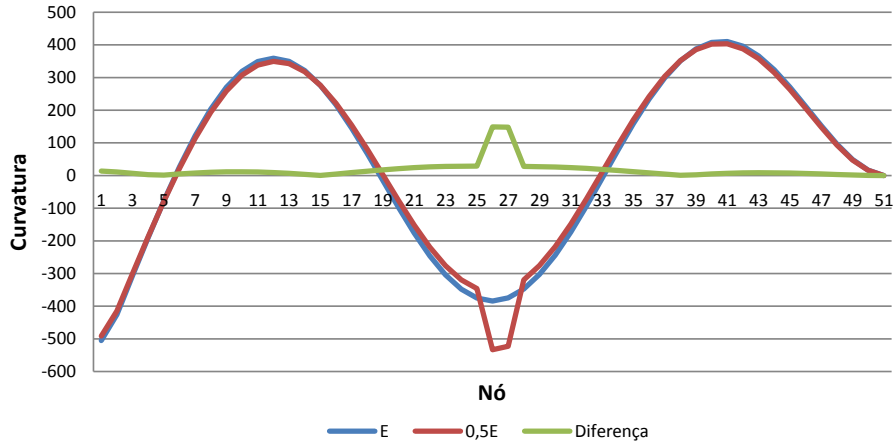


Figura 3.13: Consola com 50 elementos - Curvaturas do 4º modo com 50% de dano.

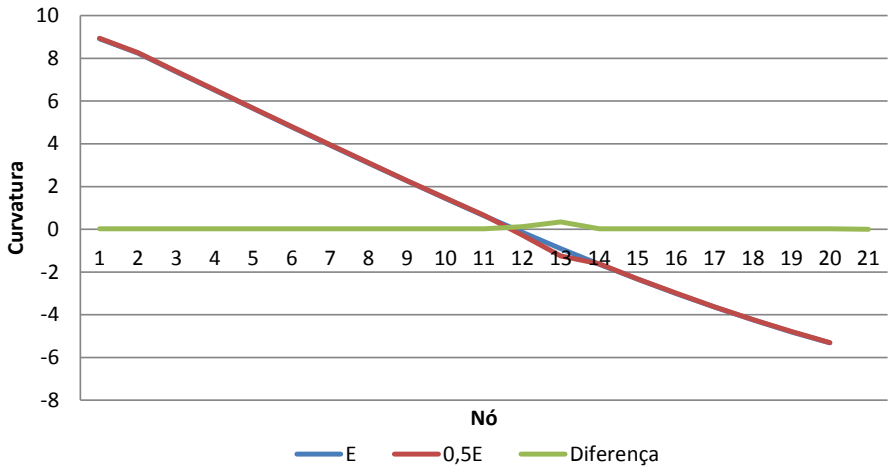


Figura 3.14: 1º Pilar do Pórtico - Curvaturas do 1º modo com 50% de dano.

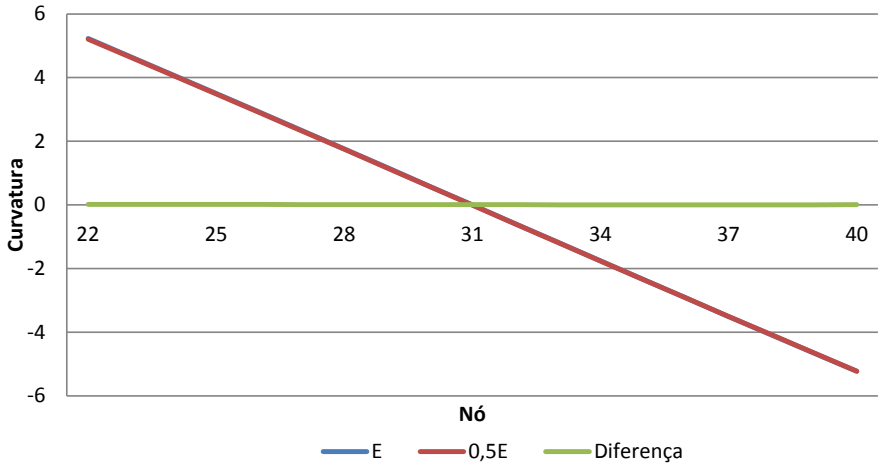


Figura 3.15: Viga do Pórtico - Curvaturas do 1º modo com 50% de dano.

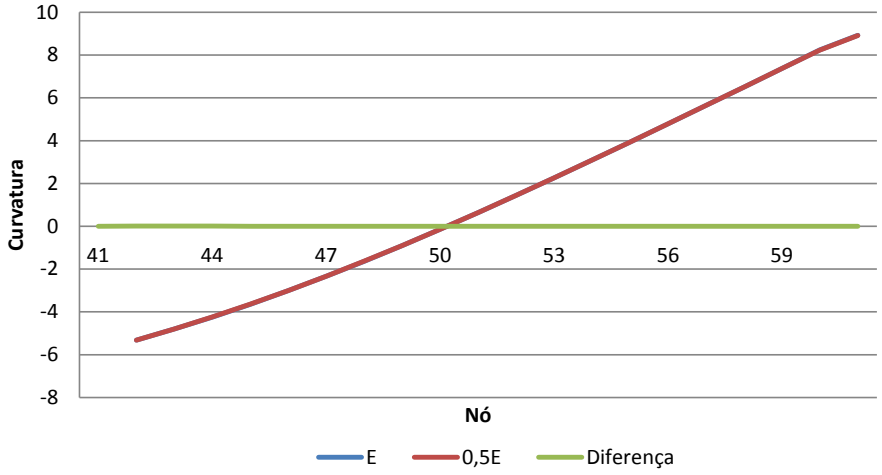


Figura 3.16: 2º Pilar do Pórtico - Curvaturas do 1º modo com 50% de dano.

A análise das curvaturas do 3º e 4º modos de vibração da viga em consola permite, como esperado, identificar a zona danificada através da diferença mais expressiva entre as curvaturas. No entanto, quando comparadas as diferenças entre estes dois modos, constata-se que o 4º modo de vibração apresenta diferenças substancialmente maiores que o 3º. Isto acontece porque a zona danificada se situa num local onde os valores dos deslocamentos modais são praticamente nulos para este modo, o que, por conseguinte, faz com que as curvaturas nesta zona sejam também pequenas. Já para o 4º modo, o dano está numa zona onde há grandes deslocamentos modais e portanto as diferenças entre as suas derivadas vão apresentar erros maiores.

No modelo do pórtico a perda de rigidez também é identificável no 12º elemento através de uma pequena variação do diferencial entre as curvaturas. O facto de se tratar de uma estrutura bidimensional onde existe interação entre vários elementos pode condicionar os resultados.

A figura 3.17 pretende mostrar que o método funciona também na identificação de mais do que um dano na mesma estrutura. Tomou-se como exemplo as curvaturas decorrentes do 4º modo de vibração do modelo da viga bi-apoiada com 2 danos, nomeadamente nos elementos 9 e 26.

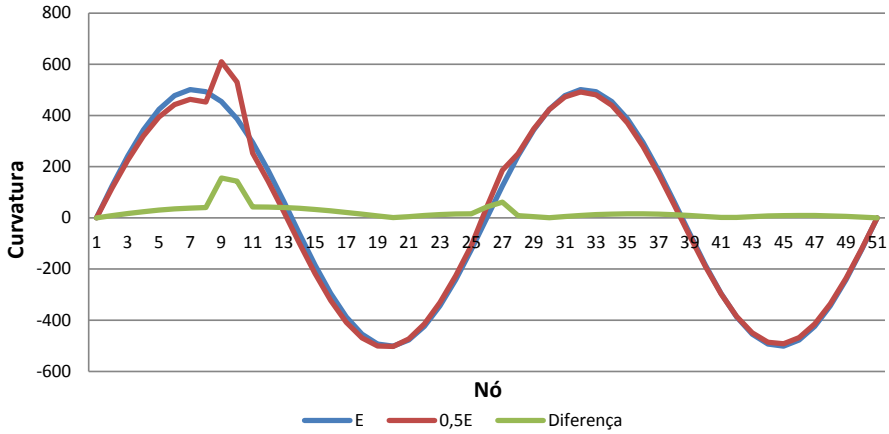


Figura 3.17: Viga com 50 elementos - Curvaturas do 4º modo com 2 danos de 50% de dano.

Neste exemplo para além de ser possível identificar os 2 danos nos elementos referidos, é também perceptível o efeito decorrente da localização do dano referido anteriormente para os casos do 3º e 4º modos de vibração da viga em consola, sendo tanto mais expressivo quanto maiores forem os valores das curvaturas. Danos onde a curvatura é aproximadamente zero são mais difíceis de identificar.

Nas figuras 3.18, 3,19 e 3.20 encontram-se representadas as diferenças entre as curvaturas dos 5 modos de vibração de cada modelo, com dano de 50%.

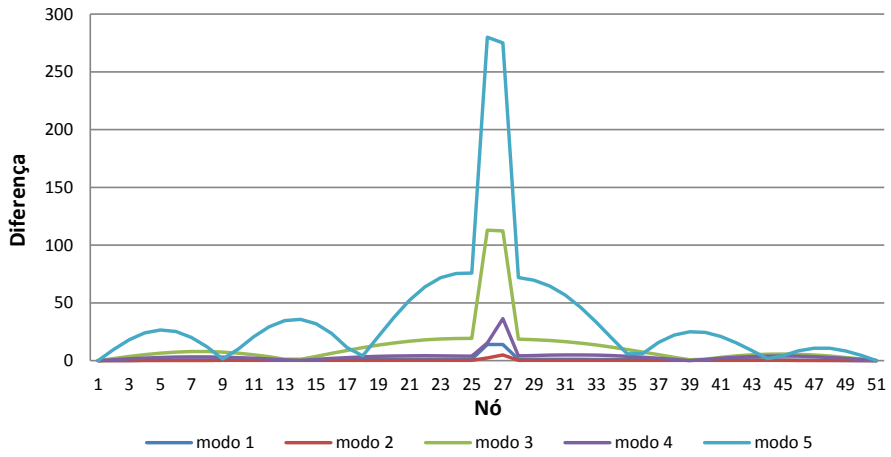


Figura 3.18: Diferenças absolutas entre curvaturas da viga intacta e danificada para os primeiros 5 modos de vibração.

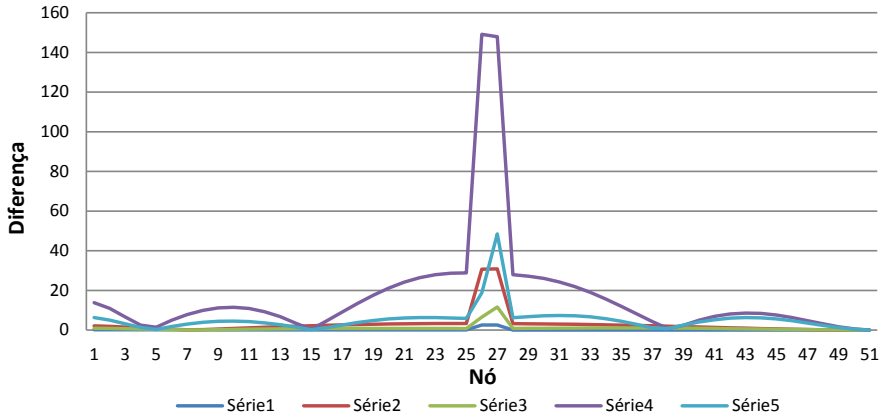


Figura 3.19: Diferenças absolutas entre curvaturas da consola intacta e danificada para os primeiros 5 modos de vibração.

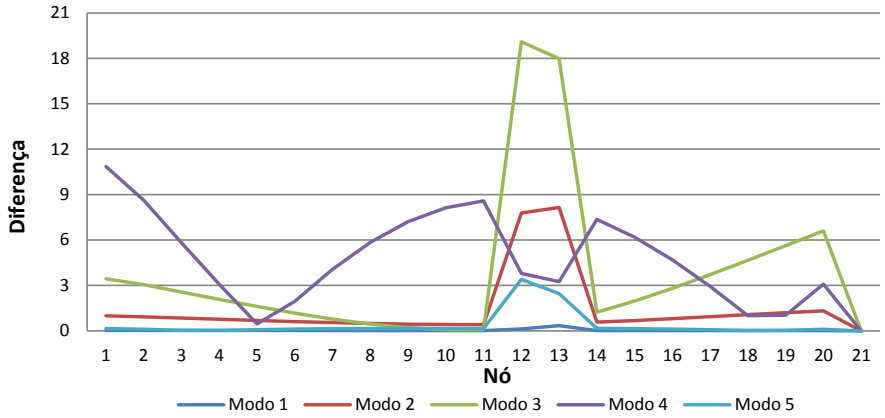


Figura 3.20: Diferenças absolutas entre curvaturas do 1º pilar do pórtico intacto e danificado para os primeiros 5 modos de vibração.

Uma análise global aos 3 modelos permite constatar que, embora não seja regra, existe uma tendência para as curvaturas dos modos mais altos acumularem diferenças maiores. A análise conjunta dos resultados dos 5 modos em cada modelo permite dizer com alguma segurança que o método permite identificar o dano com evidência. O 4º modo de vibração não permitiu obter os resultados desejados no modelo do pórtico e apresentou diferenças mais significativas em zonas não danificadas.

Para finalizar os estudos de sensibilidade relativos ao método das curvaturas, são apresentados os resultados das diferenças absolutas entre as curvaturas para vários tipos de dano no mesmo elemento. Os gráficos 3.21 a 3.26 foram escolhidos como exemplo e os referentes às curvaturas do 1º, 3º e 5º modos de vibração dos modelos da viga bi-apoiada e da consola.

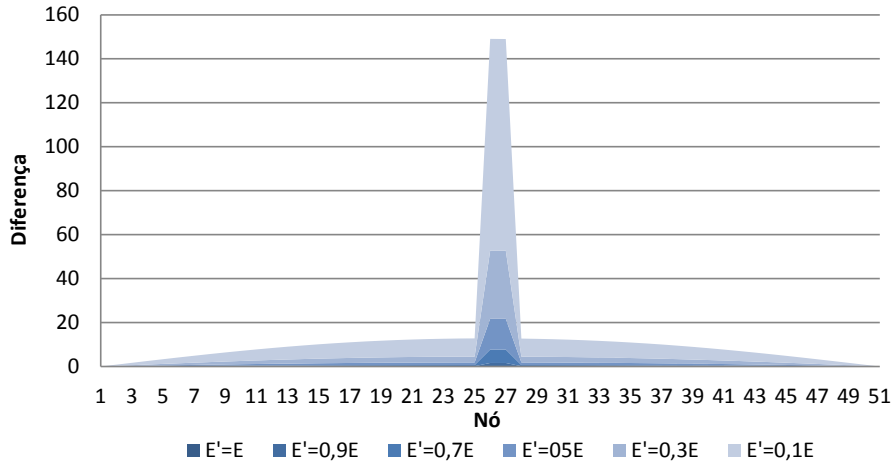


Figura 3.21: Diferença absoluta entre as curvaturas do 1º modo de vibração da viga intacta e danificada para vários graus de dano.

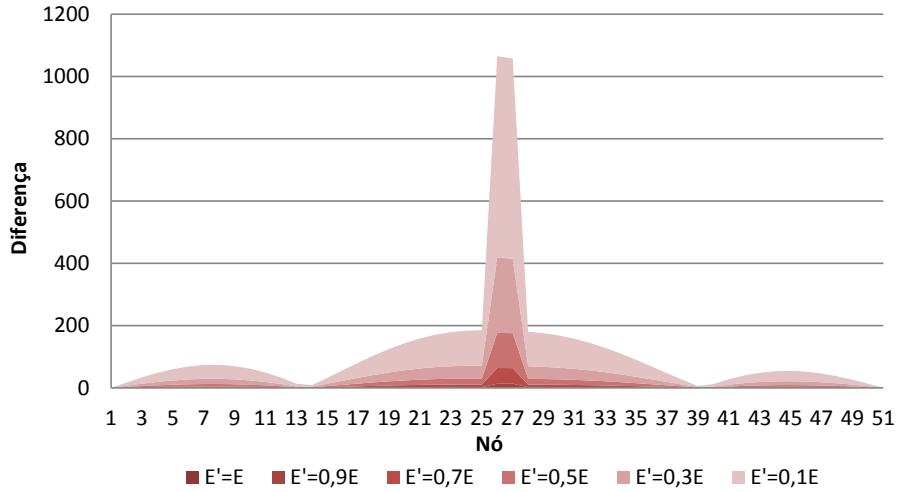


Figura 3.22: Diferença absoluta entre as curvaturas do 3º modo de vibração da viga intacta e danificada para vários graus de dano.

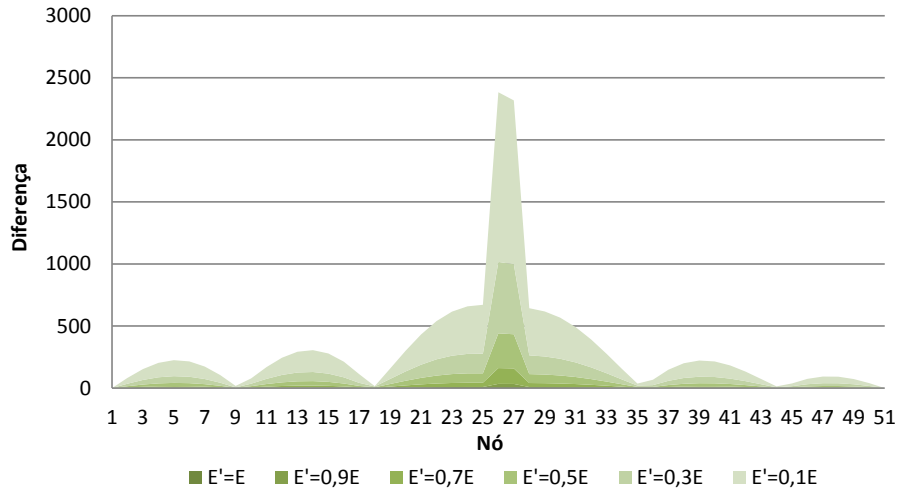


Figura 3.23: Diferença absoluta entre as curvaturas do 5º modo de vibração da viga intacta e danificada para vários graus de dano.

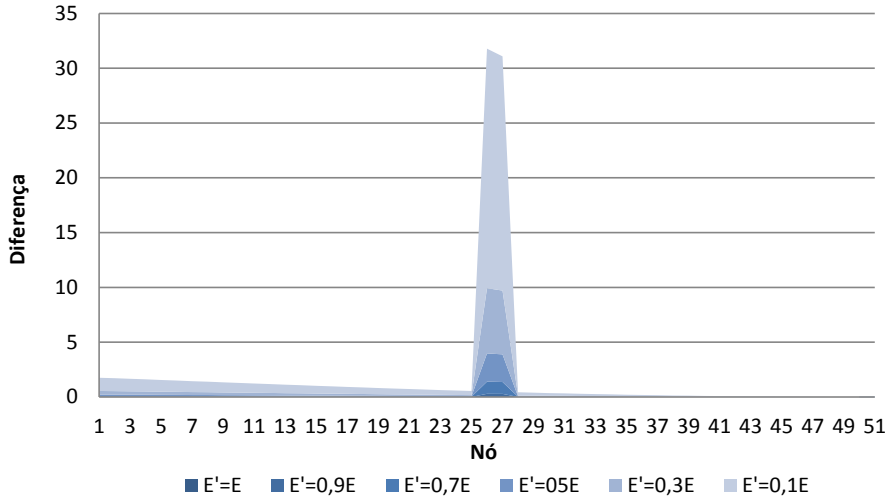


Figura 3.24: Diferença absoluta entre as curvaturas do 1º modo de vibração da consola intacta e danificada para vários graus de dano.

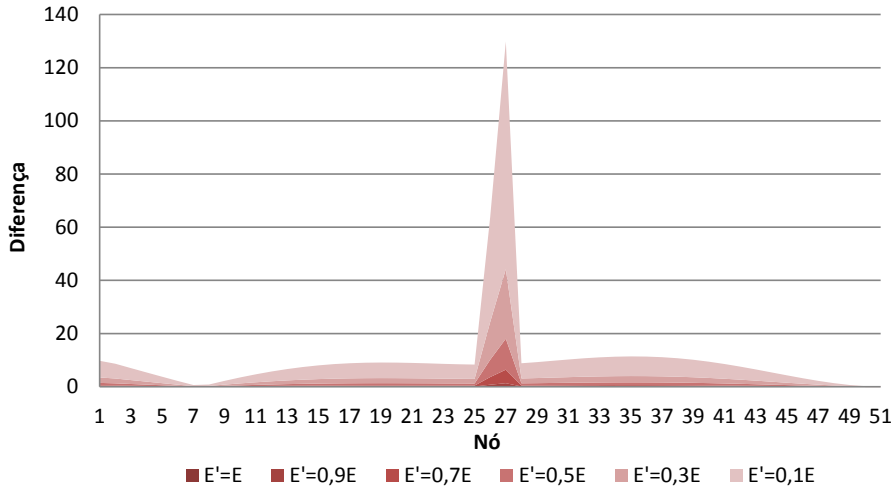


Figura 3.25: Diferença absoluta entre as curvaturas do 3º modo de vibração da consola intacta e danificada para vários graus de dano.

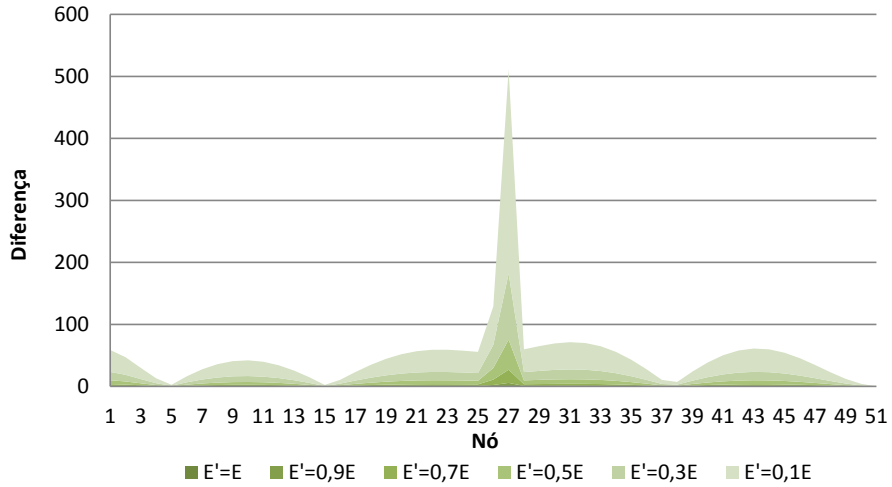


Figura 3.26: Diferença absoluta entre as curvaturas do 5º modo de vibração da consola intacta e danificada para vários graus de dano.

3.3.2 Alteração das Frequências Naturais e Parâmetros Modais

O teste do método das curvaturas foi feito em paralelo com outros métodos de identificação de dano para os mesmos modelos de estudo, sendo estes o método de alteração das frequências naturais e o método da alteração dos parâmetros modais.

As alterações da frequência natural da estrutura podem ser causadas por alterações das suas propriedades físicas, nomeadamente a rigidez e a massa, e este conhecimento constitui o argumento principal do método baseado na alteração das frequências naturais. Tal como descrito no segundo capítulo, este foi um dos primeiros métodos testados na monitorização e identificação de dano estrutural através da comparação dos valores de frequência de uma estrutura. Uma perda de rigidez, e portanto uma alteração do estado físico inicial da estrutura, traduz-se automaticamente numa diminuição da sua frequência natural.

Nas tabelas 3.6, 3.7 e 3.8 encontram-se as frequências naturais dos primeiros 5 modos de vibração dos modelos da viga bi-apoiada e da viga em consola, para um dano de 5% num elemento a meio vão.

Tabela 3.6: Frequências naturais do modelo de viga com 10, 20 e 50 elementos.

	Viga 10 Elementos			Viga 20 Elementos			Viga 50 Elementos		
Modo	f_i [Hz]	f_d [Hz]	Dif [%]	f_i [Hz]	f_d [Hz]	Dif [%]	f_i [Hz]	f_d [Hz]	Dif [%]
1	0,658	0,602	9,274	0,658	0,628	4,849	0,658	0,645	1,979
2	2,632	2,604	1,095	2,633	2,629	0,152	2,633	2,633	0,008
3	5,920	5,576	6,168	5,924	5,685	4,197	5,924	5,813	1,897
4	10,505	10,186	3,132	10,530	10,472	0,554	10,531	10,527	0,038
5	16,335	15,716	3,939	16,449	15,905	3,420	16,454	16,163	1,800

Tabela 3.7: Frequências naturais do modelo de consola com 10, 20 e 50 elementos.

	Consola 10 Elementos			Consola 20 Elementos			Consola 50 Elementos		
Modo	f_i [Hz]	f_d [Hz]	Dif [%]	f_i [Hz]	f_d [Hz]	Dif [%]	f_i [Hz]	f_d [Hz]	Dif [%]
1	0,074	0,073	1,648	0,074	0,073	0,971	0,074	0,074	0,431
2	0,457	0,418	9,358	0,463	0,441	4,948	0,464	0,455	2,021
3	1,268	1,250	1,473	1,293	1,289	0,264	1,300	1,299	0,031
4	2,459	2,326	5,705	2,526	2,429	4,014	2,546	2,499	1,869
5	4,019	3,909	2,811	4,165	4,142	0,560	4,207	4,205	0,045

Tabela 3.8: Frequências naturais do modelo de pórtico.

	Pórtico		
Modo	f_i [Hz]	f_d [Hz]	Dif [%]
1	6,750	6,7493	0,016
2	26,604	26,364	0,910
3	43,362	42,615	1,753
4	46,902	46,539	0,780
5	94,478	94,453	0,026

A análise destes valores permite concluir que existem alterações nas frequências naturais decorrentes das alterações feitas na rigidez dos elementos. Quanto maior a extensão do dano, ou seja, quanto maior for o elemento danificado, maiores são as diferenças associadas e, portanto o modelo com 10 elementos é o que apresenta maiores diferenças na frequência. Esta análise é válida para qualquer um dos modelos. Dentro de cada caso é ainda possível reparar que alguns modos de vibração têm erros maiores que outros. Atentando, por exemplo, ao modelo da viga bi-apoiada, o 1º e 3º modos de vibração apresentam erros maiores que os restantes, o que poderá estar relacionado com o facto de o dano ser numa zona onde estes modos apresentam maiores deslocamentos modais. A análise dos deslocamentos modais e dos erros das frequências no modelo da consola, nomeadamente no 2º e 4º modos, também sugere que pode existir uma relação entre estes erros e os deslocamentos modais máximos. Os modos que apresentam erros menores têm deslocamentos modais pequenos na zona danificada. No modelo do pórtico existem deslocamentos grandes nos 4 primeiros modos e no entanto não é possível estabelecer uma correlação com os valores das frequências naturais.

Nas tabelas 3.9 e 3.10 é feita a comparação entre os diferenças resultantes de vários graus de dano para os modelos com 50 elementos, intacto e danificado, da viga bi-apoiada e ada viga em consola.

Tabela 3.9: Erros na comparação de frequências entre os modelos da viga intacta e danificada para vários graus de dano.

Modo	Sem Dano	10% Dano	30% Dano	50% Dano	70% Dano	90% Dano
1	0%	0,22 %	0,85 %	1,98 %	4,56 %	16,66 %
2	0%	0,00 %	0,00 %	0,01%	0,02 %	0,08 %
3	0%	0,22 %	0,83 %	1,89 %	4,19 %	12,77 %
4	0%	0 %	0,02 %	0,04 %	0,08 %	0,27 %
5	0%	0,21 %	0,80 %	1,80 %	3,83 %	10,14 %

Tabela 3.10: Erros na comparação de frequências entre os modelos da consola intacta e danificada para vários graus de dano.

Modo	Sem Dano	10% Dano	30% Dano	50% Dano	70% Dano	90% Dano
1	0 %	0,05 %	0,19 %	0,43 %	1,01 %	3,84 %
2	0 %	0,23 %	0,87 %	2,02 %	4,61 %	16,03 %
3	0 %	0,01 %	0,02 %	0,03 %	0,06 %	0,19 %
4	0 %	0,22 %	0,82 %	1,87 %	4,09 %	12,01 %
5	0 %	0 %	0,02 %	0,05 %	0,10 %	0,31 %

Estes resultados permitem afirmar que os erros aumentam em função do aumento do dano. Estes aumentam em proporção nos mesmos modos de vibração, sendo que os modos que apresentam erros maiores são sempre os mesmos independentemente da severidade do dano. De acordo com Creed S. G. [14] são necessárias alterações de no mínimo 5% na frequência natural para que o possa ser admitida a presença de dano por este método. De acordo com o autor e com os resultados disponibilizados pelo programa de cálculo automático, é possível concluir que são precisos danos localizados muito grandes, acima de 70%, para que a identificação do dano seja fidedigna. Este método, embora permita detectar a existência de dano em ambiente controlado de laboratório, não o permite numa análise em contexto real já que as estruturas estão sempre sujeitas a pequenas variações dos seus parâmetros modais, da mesma ordem de grandeza dos erros detectados nos modelos de estudo. O método das frequências não permite então identificar com segurança nem a severidade nem a localização do dano.

Na mesma linha de raciocínio deste método das frequências naturais, a alteração da rigidez estrutural altera inevitavelmente os deslocamentos da estrutura. Os exemplos das figuras 3.27 a 3.30 pretendem ilustrar as diferenças entre os deslocamentos modais de modelos intactos e danificados e inferir sobre a hipótese deste método de análise constituir uma ferramenta válida para a identificação de dano. O elemento danificado encontra-se a meio vão com uma redução de 50% da rigidez.

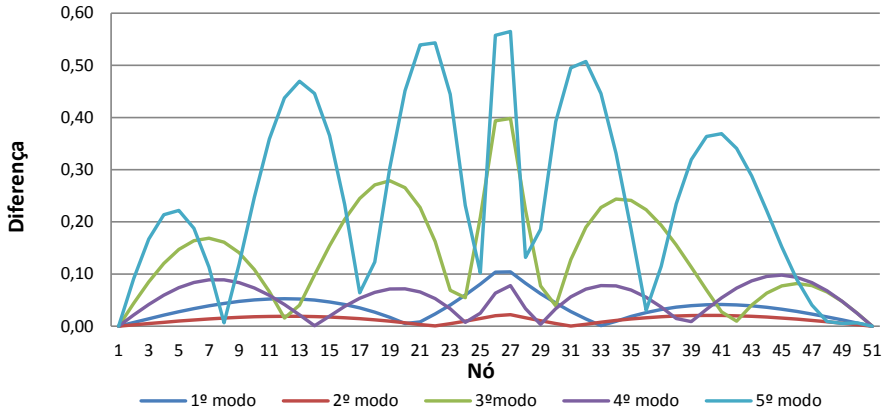


Figura 3.27: Diferenças absolutas entre os deslocamentos modais dos primeiros 5 modos de vibração dos modelos intacto e danificado da viga bi-apoiada.

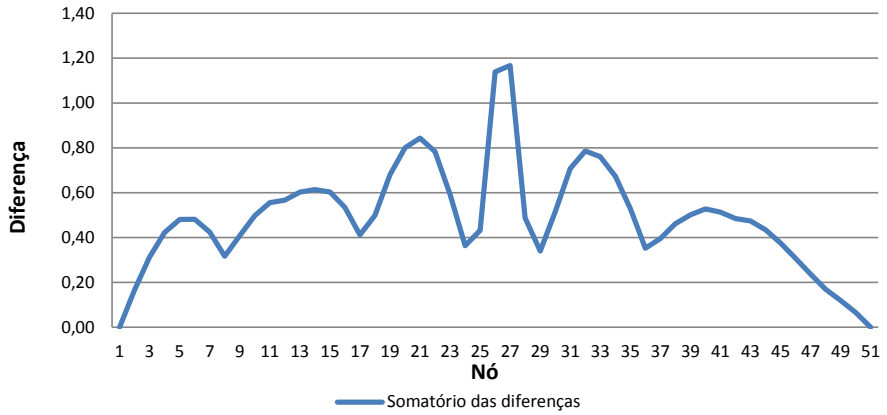


Figura 3.28: Somatório das diferenças absolutas entre os deslocamentos modais dos primeiros 5 modos de vibração dos modelos intacto e danificado da viga bi-apoiada.

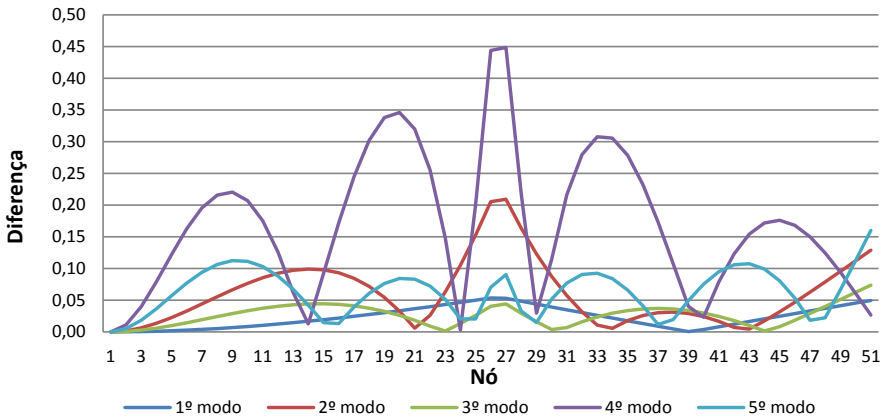


Figura 3.29: Diferenças absolutas entre os deslocamentos modais dos primeiros 5 modos de vibração dos modelos intacto e danificado da viga em consola.

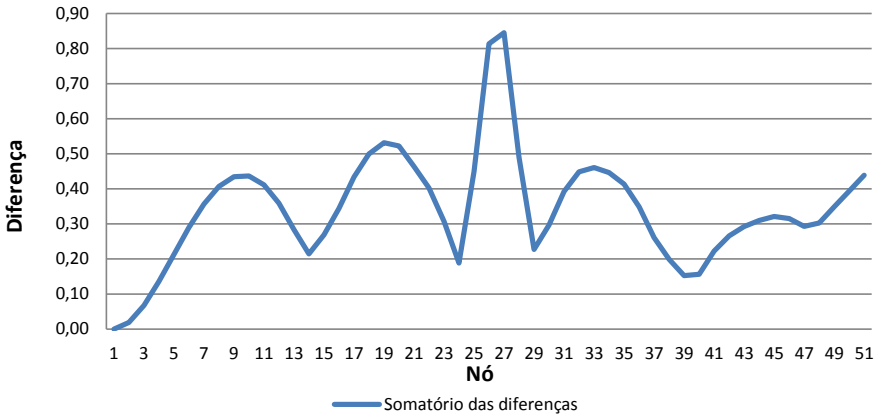


Figura 3.30: Somatório das diferenças absolutas entre os deslocamentos modais dos primeiros 5 modos de vibração dos modelos intacto e danificado da viga em consola.

A análise dos resultados para ambos os modelos permite identificar diferenças ligeiramente maiores nas zonas danificadas em alguns dos modos de vibração. No modelo da viga bi-apoiada isto não é verificável para o 2º e 4º modos de vibração, os quais apresentam erros maiores em zonas não danificadas. O mesmo se passa com os deslocamentos modais do 1º, 3º e 5º modos do modelo da viga em consola. A análise do somatório das diferenças de ambos os modelos sugere que esta análise, embora com erros grandes, possibilita a localização dos danos. Uma vez que estes somatório é fortemente influenciado pelos valores do 5º modo no caso da viga e pelos valores do 4º modo no caso da consola, poderia ser interessante estudar a relação com os respectivos factores de participação de massa. Este método dá garantias de segurança apenas na identificação de dano.

Da mesma maneira que a alteração das características físicas das estruturas podem alterar as frequências naturais, também os parâmetros modais podem ser alterados. No segundo capítulo foi feita referência a dois dos indicadores estatísticos que podem ser utilizados na monitorização estrutural, o *Modal Assurance Criterion (MAC)* e o *Coordinate Modal Assurance Criterion (COMAC)*.

O indicador *MAC*, sugerido por Allemang e Brown [2], avalia a relação existente entre dois modos de vibração através de um índice de correlação global. Os vectores dos deslocamentos modais, do modelo intacto e do modelo danificado, são escalados para que os termos diagonais da matriz, que relacionam diretamente cada modo de vibração, sejam unitários. Portanto, este índice vai variar entre 0 e 1, onde 0 significa que não há correlação entre os modos e 1 representa uma correlação perfeita. Uma vez que a presença de dano na estrutura altera os modos de vibração e consequentemente os termos da matriz, este indicador poderá produzir resultados na identificação de danos. Este índice é dado pela expressão 3.15.

$$MAC = \frac{|\{\phi_{int}\}^T\{\phi_{dan}\}|^2}{\{\phi_{int}\}^T\{\phi_{int}\}\{\phi_{dan}\}^T\{\phi_{dan}\}} \tag{3.15}$$

Em que,

$\{\phi_{int}\}$ é o vector dos deslocamentos modais do modelo intacto;
$\{\phi_{int}\}^T$ é o vector transposto dos deslocamentos modais do modelo intacto;
$\{\phi_{dan}\}$ é o vector dos deslocamentos modais do modelo danificado;
$\{\phi_{dan}\}^T$ é o vector transposto dos deslocamentos modais do modelo danificado.

Os termos da matriz serão constituídos por todas as combinações possíveis entre os vectores dos deslocamentos modais do modelo intacto e o danificado, onde cada um dos 5 termos da diagonal representa a relação entre os 2 vectores do mesmo modo. As matrizes dos valores *MAC* para os 5 primeiros modos de vibração dos 3 modelos de estudo são apresentadas nas figuras 3.31 a 3.36, para um dano de 50% nos elementos referidos nos restantes testes. São ainda apresentados os respectivos gráficos com os valores *MAC* das diagonais principais.

Figura 3.31: Valores *MAC* dos deslocamentos modais dos modelos intactos e danificados da viga bi-apoiada.

		Viga Intacta				
	M	1	2	3	4	5
Viga Dan.	1	1,000	0,000	0,000	0,000	0,000
	2	0,000	1,000	0,000	0,000	0,000
	3	0,000	0,000	1,000	0,000	0,000
	4	0,000	0,000	0,000	1,000	0,000
	5	0,000	0,000	0,000	0,000	0,999

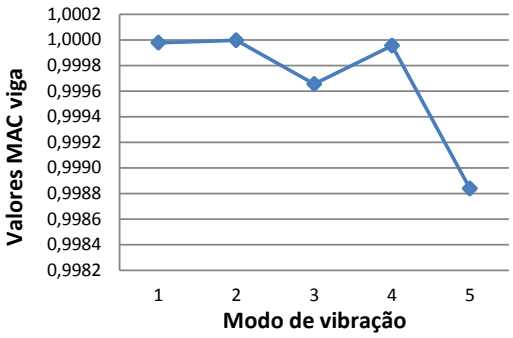


Figura 3.32: Valores *MAC* da diagonal principal dos modelos da viga bi-apoiada.

Figura 3.33: Valores *MAC* dos deslocamentos modais dos modelos intactos e danificados da viga em consola.

		Consola Intacta				
	M	1	2	3	4	5
Consola Dan.	1	1,000	0,002	0,001	0,001	0,001
	2	0,001	1,000	0,002	0,001	0,001
	3	0,001	0,001	1,000	0,001	0,001
	4	0,002	0,002	0,002	1,000	0,001
	5	0,001	0,001	0,001	0,001	1,000

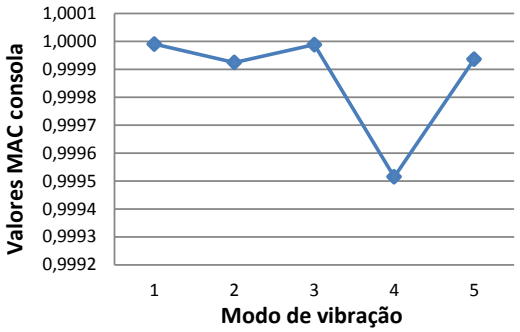


Figura 3.34: Valores *MAC* da diagonal principal dos modelos da viga em consola.

Figura 3.35: Valores *MAC* dos deslocamentos modais dos modelos intactos e danificados do pórtico.

		Pórtico Intacto				
	M	1	2	3	4	5
Pórtico Dan.	1	1,000	0,000	0,253	0,000	0,027
	2	0,000	0,999	0,000	0,000	0,000
	3	0,246	0,022	0,976	0,001	0,001
	4	0,004	0,000	0,013	0,983	0,003
	5	0,027	0,000	0,001	0,000	1,000

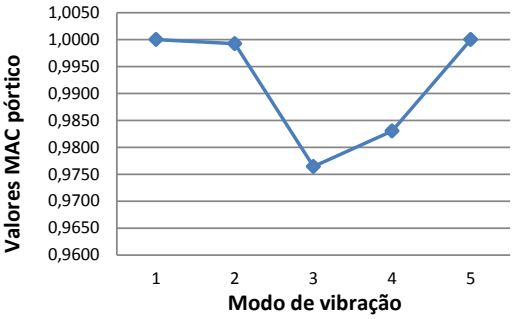


Figura 3.36: Valores *MAC* da diagonal principal dos modelos da viga bi-apoiada e da viga em consola.

Os valores da diagonal principal são aproximadamente todos unitários nos 3 modelos. No modelo do pórtico existem erros mais expressivos fora da diagonal que, no entanto, nada indicam relativamente ao local danificado. Foram calculados os índices *MAC*, tanto para outros graus de dano como para modelos com menos elementos, e estes apenas apresentaram pequenas variações na casa das centésimas, nada acrescentando às possibilidades de análise aqui expostas. Para estes modelos o índice *MAC* não foi capaz de detectar o estado de dano.

O indicador *COMAC*, proposto por Lieven e Ewins [31], à semelhança do *MAC*, também permite avaliar a correlação entre modos de vibração, com a diferença de este método possuir um carácter local que permite estabelecer a concordância pontual entre os modos de vibração da estrutura intacta e danificada. A comparação entre os valores será então feita ao nível de cada um dos graus de liberdade do modelo. O *COMAC* faz uso de um índice que varia entre 0 e 1, onde 0 indica que não há relação entre os valores e 1 que há concordância total. Este índice é calculado pela expressão 3.16.

$$COMAC = \frac{\sum_{r=1}^{n} |\phi_{int}^{i} \phi_{dan}^{i}|^2}{\sum_{r=1}^{n} \{\phi_{int}^{i}\}^2 \sum_{r=1}^{n} \{\phi_{dan}^{i}\}^2} \tag{3.16}$$

Em que,

ϕ_{int}^{i} é o deslocamento modal do modelo intacto no modo n e GDL i;
ϕ_{dan}^{i} é o deslocamento modal do modelo danificado no modo n e GDL i.

Nas tabelas 3.11, 3.12 e 3.13 encontram-se os valores *COMAC* que comparam os modelos intactos e danificados dos modelos de 50 elementos da viga bi-apoiada e viga em consola e para o modelo do pórtico. Os danos são os mesmos já referidos nos restantes métodos, com uma percentagem de 50% na redução de rigidez.

Tabela 3.11: Valores *COMAC* dos deslocamentos modais do modelos intacto e danificado da viga bi-apoiada.

Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC
1	1,00	11	1,00	21	1,00	31	1,00	41	1,00
2	1,00	12	1,00	22	1,00	32	1,00	42	1,00
3	1,00	13	1,00	23	1,00	33	1,00	43	1,00
4	1,00	14	1,00	24	1,00	34	1,00	44	1,00
5	1,00	15	1,00	25	1,00	35	1,00	45	1,00
6	1,00	16	1,00	26	1,00	36	1,00	46	1,00
7	1,00	17	1,00	27	1,00	37	1,00	47	1,00
8	1,00	18	1,00	28	1,00	38	1,00	48	1,00
9	1,00	19	1,00	29	1,00	39	1,00	49	1,00
10	1,00	20	1,00	30	1,00	40	1,00	50	1,00
								51	1.00

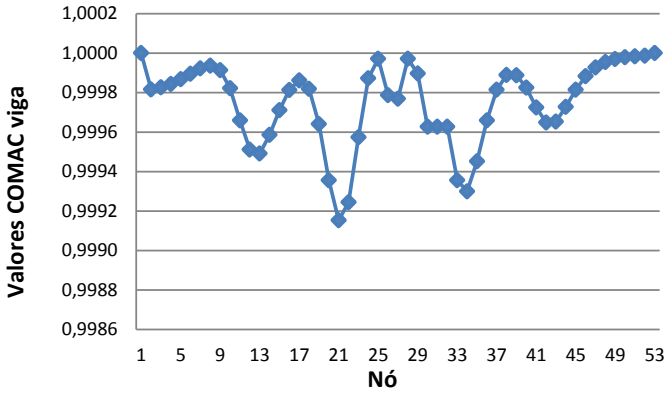


Figura 3.37: Valores *COMAC* dos modelos da viga bi-apoiada.

Tabela 3.12: Valores *COMAC* dos deslocamentos modais do modelos intacto e danificado da viga em consola.

Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC
1	1,00	11	1,00	21	1,00	31	1,00	41	1,00
2	1,00	12	1,00	22	1,00	32	1,00	42	1,00
3	1,00	13	1,00	23	1,00	33	1,00	43	1,00
4	1,00	14	1,00	24	1,00	34	1,00	44	1,00
5	1,00	15	1,00	25	1,00	35	1,00	45	1,00
6	1,00	16	1,00	26	1,00	36	1,00	46	1,00
7	1,00	17	1,00	27	1,00	37	1,00	47	1,00
8	1,00	18	1,00	28	1,00	38	1,00	48	1,00
9	1,00	19	1,00	29	1,00	39	1,00	49	1,00
10	1,00	20	1,00	30	1,00	40	1,00	50	1,00
								51	1.00

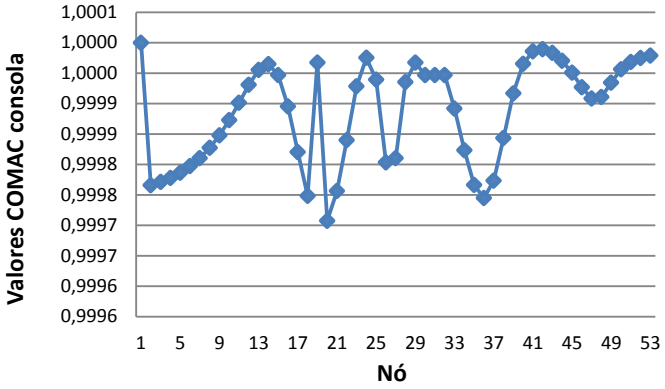


Figura 3.38: Valores *COMAC* do modelo da viga em consola.

Tabela 3.13: Valores *COMAC* dos deslocamentos modais do modelos intacto e danificado do pórtico.

Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC	Ponto	COMAC
1	1,00	11	0,99	21	1,00	31	0,99	41	1,00	51	0,99
2	0,99	12	0,99	22	1,00	32	0,99	42	1,00	52	0,99
3	0,99	13	0,99	23	1,00	33	0,99	43	1,00	53	0,99
4	0,99	14	0,99	24	1,00	34	0,99	44	1,00	54	0,99
5	0,99	15	0,99	25	1,00	35	1,00	45	1,00	55	0,99
6	0,99	16	0,99	26	0,99	36	1,00	46	1,00	56	0,99
7	0,99	17	1,00	27	0,99	37	1,00	47	1,00	57	0,99
8	0,99	18	1,00	28	0,99	38	1,00	48	1,00	58	0,99
9	0,99	19	1,00	29	0,99	39	1,00	49	1,00	59	0,99
10	0,99	20	1,00	30	1,00	40	1,00	50	1,00	60	0,99
										61	1.00

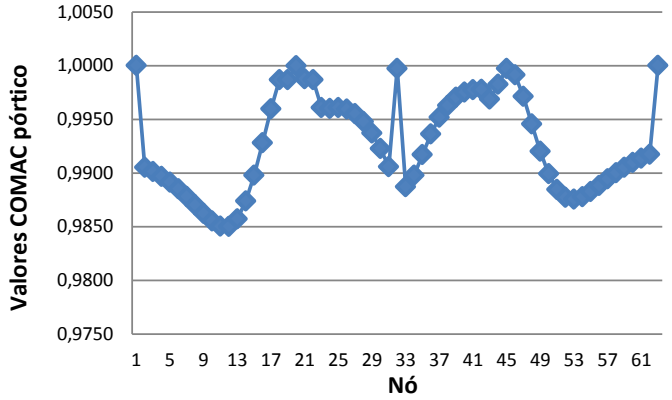


Figura 3.39: Valores *COMAC* do modelo do pórtico.

O uso do *Coordinate Modal Assurance Criterion* permitiu obter os índices que expressam a relação entre os deslocamentos modais ao nível de grau de liberdade dos modelos em estudo. Em todos os casos, com algumas oscilações praticamente imperceptíveis, os índices foram unitários em todos os pontos. Tal como no estudo do *MAC*, foram testados modelos com graus de dano e malhas diferentes que não constituíram uma mais valia na utilização do método. Portanto, este método não permite inferir sobre o estado de dano em estruturas.

3.4 Conclusões do Método

Neste capítulo são apresentadas as conclusões referentes à aplicação dos métodos de detecção de dano estudados e abordada a hipótese de desenvolvimento de uma plataforma Java que permita testar o uso destes métodos.

No que diz respeito ao método das curvaturas, foi constatado que a qualidade dos seus resultados é função da dimensão da malha de pontos escolhida, sendo que quanto maior o refinamento melhores são os resultados. As diferenças entre as curvaturas diminuem com o aumento do número de elementos.

Foi também possível concluir que é mais fácil identificar os danos em zonas onde os deslocamentos modais, e consequentemente a curvatura dos elementos, sejam maiores. Para um dado modo de vibração, o dano numa zona com deslocamentos modais residuais conduz a diferenças menos expressivas entre as curvaturas do modelo intacto e danificado.

Os resultados obtidos no modelo do pórtico permitiram concluir que este modelo, por ser uma estrutura bidimensional onde existe dependência entre os graus de liberdade horizontais e verticais, apresenta um comportamento mais instável que os restantes, traduzindo-se em erros maiores.

Este método permite ainda, em função do número de elementos do modelo, identificar mais do que um dano e tem sensibilidade suficiente para detectar a severidade deste.

A análise dos resultados em todos os modelos permitiu também identificar uma tendência para a existência de diferenças maiores entre as curvaturas dos modos mais altos. Ainda assim, por norma, os

modos mais altos possuem um fator de participação de massa menor, o que significa que a sua influência no comportamento global da estrutura é menor.

Em termos gerais, os resultados obtidos pelo método das curvaturas na identificação de danos permitem afirmar que é um método fidedigno e permite identificar, localizar e conhecer o estado de degradação estrutural.

O método da comparação de frequências naturais permitiu identificar alterações nas frequências quando alterada a rigidez dos elementos nos modelos. Os erros identificados foram tanto maiores quanto maior a extensão do dano, dado expresso por intermédio da diferença do número de elementos do modelo.

Foi ainda possível constatar que os erros aumentam em função do aumento do dano. O seu aumento é proporcional nos mesmos modos de vibração, sendo que os modos que apresentam maiores erros continuam a ser os mesmos independentemente da severidade do dano. A necessidade de existência de alterações mínimas de 5% nas frequências naturais para identificar o dano com segurança implica que os danos na estrutura sejam elevados, o que pode comprometer e negligenciar a segurança da estrutura.

Conclui-se, por fim, que o método das frequências naturais não permite identificar a existência de dano numa estrutura real nem dar indicações quanto à sua localização e severidade.

Relativamente ao método da diferença entre deslocamentos, foi possível constatar que, à semelhança do método das frequências naturais, permite identificar a presença de dano sem no entanto dar provas consistentes quanto à sua localização.

Os métodos que recorrem a indicadores estatísticos, *Modal Assurance Criterion* e *Coordinate Modal Assurance Criterion*, foram capazes de identificar pequenas alterações na rigidez dos modelos analíticos. No entanto, nenhum deles foi capaz de localizar e caracterizar o estado de degradação estrutural.

Dos 5 métodos abordados apenas o método das curvaturas provou ter a capacidade de conseguir uma análise fidedigna do estado de dano nestas estruturas. Neste âmbito foi desenvolvido um programa no formato *Java* que permite ao usuário testar as capacidades do método nos modelos aqui abordados.

Capítulo 4

Programa de Cálculo

Neste capítulo são apresentados os conceitos e processos utilizados na concepção do aplicativo *Web* de cálculo das curvaturas. Este foi desenvolvido através de uma plataforma *NetBeans IDE*, que constitui um ambiente de desenvolvimento integrado de código aberto onde é possível criar o *sofware* pretendido em linguagem *Java*.

4.1 Manual Teórico

4.1.1 Introdução ao Java e Definição dos Modelos

O *software IDE NetBeans* auxilia programadores a escrever, compilar, depurar e instalar aplicações. Foi projectado na forma de uma estrutura que visa simplificar o desenvolvimento e aumentar a produtividade, uma vez que reúne numa única aplicação todas estas funcionalidades. O *NetBeans* é escrito na linguagem *Java* e portanto funcional em qualquer sistema operacional que suporte a máquina virtual *Java* (JVM), programa que carrega e executa os aplicativos Java. Este *software* foi escolhido por possuir os recursos necessários ao desenvolvimento da plataforma pretendida, que são, essencialmente, um editor de código fonte com recursos para aplicações *Web* e aplicações visuais com *Swing*, que é uma API (Interface de Programação de Aplicativos) *Java* para interfaces. Por outras palavras, permite desenvolver um aplicativo que fica disponível numa página de internet onde o utilizador pode explorar as suas funcionalidades por intermédio de uma interface gráfica. Existe ainda a vantagem do visualizador de classes estar integrado ao de interfaces, que gera automaticamente o código dos componentes de forma bem organizada e torna o seu uso mais simples e intuitivo [44, 12].

O primeiro passo foi construir a interface gráfica através da qual o utilizador pode estabelecer um contacto intuitivo com o programa de cálculo. Foi criada uma classe, *Metodo_Curvaturas.java*, onde foram implementados os cálculos referentes ao método das curvaturas e uma classe, *Interface.java*, onde foi definida a interface. Na secção *design* é possível aceder às componentes disponibilizadas pelo *NetBeans*, que estão organizadas numa *palette* onde constam vários grupos *Swing* com as ferramentas visuais existentes que podem ser usados das mais variadas maneiras. A interface em causa fará uso maioritário de painéis, caixas de texto, *comboboxes* e grupos de botões. A figura 4.1 mostra os elementos disponíveis na palete do *NetBeans* e as propriedades disponibilizadas quando selecionado um componente *JButton*:

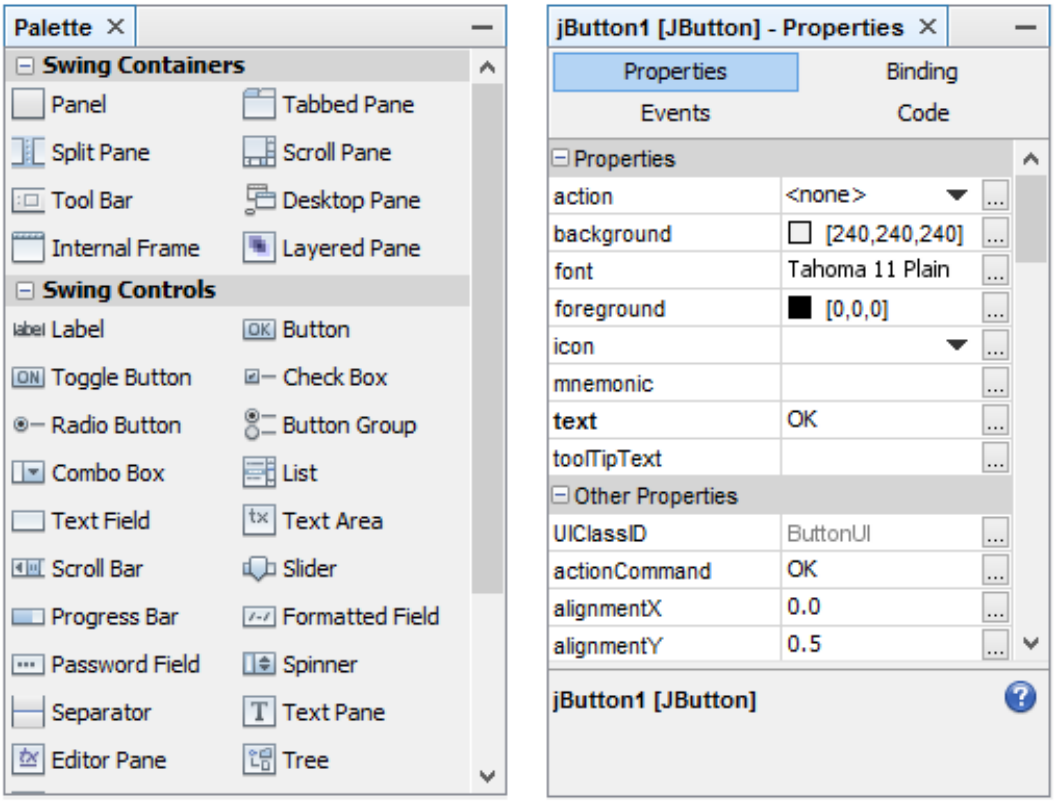


Figura 4.1: Ferramentas de *design NetBeans - palette* e propriedades de um botão.

Foram então criados 6 modelos distintos onde vai ser possível usar o método das curvaturas. Os modelos diferem no número de elementos, pretendendo-se com esta diferença avaliar a qualidade de aplicação do método em função das características do grau de refinamento nos modelos da viga bi-apoiada e da viga em consola. Os modelos disponíveis são os seguintes:

- Viga bi-apoiada (10 elementos)
- Viga bi-apoiada (20 elementos)
- Viga bi-apoiada (50 elementos)
- Consola (10 elementos)
- Consola (20 elementos)
- Consola (50 elementos)

Cada um destes modelos é constituído por um número de botões (correspondente ao número de elementos) e por uma imagem do tipo .png que pretende representar as condições de apoio. A figura 4.2 pretende ilustrar como foram definidos estes modelos na secção de *design*:

Figura 4.2: Modelos com 10, 20 e 50 elementos da viga bi-apoiada e consola.

O acesso a cada um destes modelos é feito através de uma *combobox*, que é uma caixa que apresenta uma lista dos casos disponíveis. Quando a aplicação estiver a correr, a escolha de um modelo na *combobox* implica que o programa faça os cálculos e apresente os resultados unicamente para esse modelo. Para calcular o modelo da viga bi-apoiada com 10 elementos por exemplo, é definida na linha de comandos da *combobox* uma condição que permita ao utilizador aceder apenas ao painel que contém os botões correspondentes a este modelo. Este raciocínio foi aplicado para todos os modelos.

Em cada caso de estudo, viga bi-apoiada e viga em consola, vão ser estudados os primeiros 5 modos de vibração, à semelhança do estudo feito no capítulo 3. A escolha de cada modo de vibração será feita também através de uma *combobox* disponível na interface da aplicação, sendo que, depois de escolhido, será obtido o *output* gráfico das curvaturas correspondentes a esse modo. Nesta *combobox* está também disponível uma outra opção que disponibiliza a diferença absoluta entre o somatório das curvaturas do modelo nominal e o somatório das curvaturas do modelo danificado, de acordo com os casos de estudo apresentados no capitulo 3. Na figura 4.3 estão representadas a *combobox* que permite escolher o modelo e a *combobox* que permite escolher o modo de vibração e somatório das diferenças:

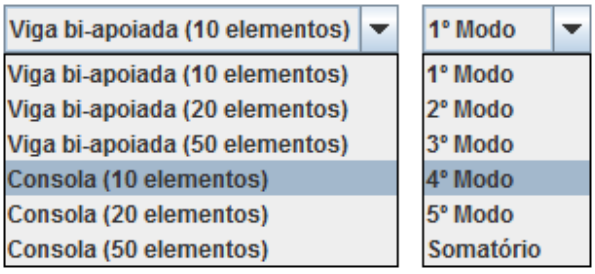


Figura 4.3: *Combobox* dos modelos e *combobox* dos modos de vibração.

Como já foi referido, os botões de cada modelo representam o número de elementos nos quais foi dividida a viga ou a consola. A cada um destes botões será possível atribuir um dano específico dentro da

gama disponível. Tomando como exemplo o primeiro modelo, viga bi-apoiada de 10 elementos, quando selecionado um botão a meio vão, vai aparecer uma janela onde é possível escolher o grau de dano pretendido para o elemento respectivo. Um dano de 10% corresponde a uma rigidez de 0,9E, um dano de 30% a uma rigidez de 0,7E e assim sucessivamente. A figura 4.4 ilustra esta situação:

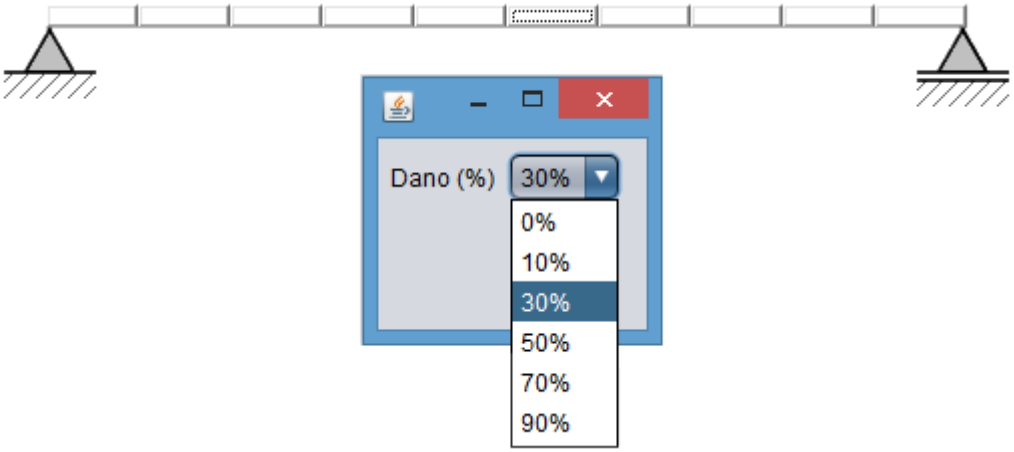


Figura 4.4: Janela de selecção de dano.

Uma vez selecionado o dano pretendido, este elemento vai assumir uma cor correspondente a uma escala pré-definida que pretende simular a gravidade do dano. Esta atribuição de cor é feita para tornar a aplicação mais intuitiva e apelativa ao utilizador. Caso fossem atribuídos 2 danos distintos, por exemplo 50% ao 3º elemento e 30% ao 6º elemento, o modelo, de acordo com a escala de dano, apresentaria o aspecto da figura 4.5.

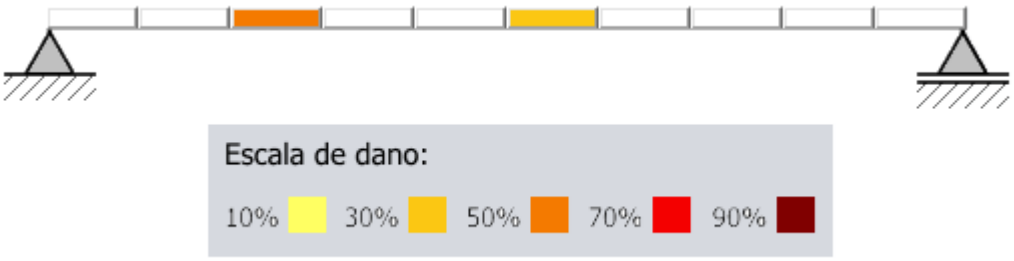


Figura 4.5: Modelo com elementos danificados a 50 e 30% respectivamente e escala de dano.

Escolhidos o modelo, modo de vibração e dano(s) nos elementos, é possível então calcular as curvaturas dos modelos nominal e danificado e as diferenças correspondentes à comparação dos mesmos. Para proceder a estes cálculos basta carregar no botão ***Calcular*** no canto inferior esquerdo, que está diretamente associado ao procedimento que permite correr o programa e plotar os gráficos. Caso se pretenda fazer o estudo de outro modelo ou modo de vibração é possível fazê-lo através do botão ***Reiniciar***, no canto inferior esquerdo, que permite que o programa retorne ao ponto inicial. Na

eventualidade de se pretender fazer uma comparação entre os resultados de 2 modos de vibração distintos basta selecionar a janela principal da aplicação, escolher os modos/danos pretendidos e ***Calcular*** novamente. Este processo pode ser repetido para qualquer modelo e qualquer caso que se pretenda.

A estrutura do programa *Java* deve conter sempre uma declaração na forma *public class* NomeDaFunção { *public static void main (String args[])* código da função}. É dentro desta classe que se encontram todos os dados, funções e procedimentos. Todas as aplicações em *Java* (embora não seja uma exigência no caso das *applets*) devem conter um procedimento chamado *main*. Os procedimentos contêm um conjunto de declarações de dados e de comandos que são executados mediante a sua solicitação. Neste caso de estudo por exemplo, existe um procedimento *main*, no ficheiro *NewJFrame* (ficheiro onde é feita a gestão dos componentes da interface) que inicia as variáveis representativas das estruturas em análise e inicia a janela de interface com o utilizador. Quando o utilizador carrega no botão ***Calcular***, são executadas as tarefas existentes dentro do procedimento *plot* e só depois é possível executar os gráficos.

Uma vez definida a interface da aplicação é preciso definir as funções que vão permitir obter as curvaturas do caso danificado e não danificado nos modelos da viga bi-apoiada e viga em consola. De seguida é feita uma pequena introdução ao método de definição das funções em linguagem java. A explicação das funções criadas será acompanhada de um exemplo simples de aplicação em cada um dos modelos tipo, uma viga bi-apoiada com 4 elementos e uma viga em consola também ela com 4 elementos.

Funções são rotinas ou sub-rotinas automatizadas e devem ser criadas sempre que pretendemos usar a mesma codificação para algo específico. No desenvolvimento desta aplicação foram criadas várias funções para evitar a escrita do código repetidamente, podendo desta maneira ser chamadas as funções sempre que necessário.

As funções programadas na linguagem Java podem ser de vários tipos. Estas podem simplesmente executar uma rotina, podem conter argumentos a ser processados pela função (variáveis de entrada) e podem ainda retornar valores de processos executados dentro delas, guardando esses valores dentro de uma variável no programa. Simplificadamente, dentro de uma classe, as funções podem ser definidas como:

- *public static void* NomeDaFunção { código da função }, quando queremos criar uma função que pode ser solicitada globalmente e não retorna nenhum valor;
- *public static int* NomeDaFunção (argumentos) { código da função }, quando queremos criar uma função com *n* argumentos que pode ser solicitada globalmente e retornar valores. A variável *int* serve para trabalhar com números inteiros e foi usada apenas como exemplo, podendo ser em vez disso um *float* caso o interesse fosse trabalhar com números racionais.

Uma variável é simplesmente um espaço vago, reservado e rotulado para armazenar dados. Todas as variáveis têm um nome e um valor, correspondentes à informação que se pretende atribuir-lhes. Por exemplo *int n*, especifica que *n* é o nome de uma variável pode armazenar um número inteiro como valor. O valor da variável pode ser consecutivamente alterado ao longo da execução do programa, por intermédio de atribuições de valor.

As variáveis usadas para definir as funções no *NetBeans*, correspondentes a grandezas físicas da estrutura, foram por simplificação tomadas como unitárias. Estas variáveis são o comprimento da estrutura, L, a rigidez de flexão dos elementos, EI, e a massa total, M. A admissão destes valores não prejudica a análise das curvaturas porque esta é feita através da comparação dos valores danificado e nominal de cada modelo e estes fazem ambos uso das mesmas variáveis unitárias. Portanto, as diferenças, em proporção, serão idênticas.

4.1.2 Rigidez e Massa no Modelo da Viga

A matriz que define a rigidez global da estrutura é definida através da assemblagem das matrizes locais dos elementos constituintes do modelo. Assim sendo, é necessário definir primeiramente as matrizes locais de cada elemento tipo. Recorrendo às tabelas técnicas que apresentam as forças de fixação para deslocamentos impostos, foi possível obter as matrizes locais, para os elementos extremos e para os elementos internos, necessárias para definir os modelos da viga e consola. Para o primeiro elemento da viga foi definida a função *Build_Klocal_Vigaex1*, para o último elemento a função *Build_Klocal_Vigaex2* e para os elementos internos a função *Build_Klocal_in*. Estas funções possuem a rigidez de flexão, EI, e o comprimento, L, como variáveis de entrada e resultam nas seguintes matrizes:

$$K_{local\_vigaex1} = \begin{bmatrix} \frac{3EI}{L^3} & \frac{3EI}{L^2} \\ \frac{3EI}{L^2} & \frac{3EI}{L} \end{bmatrix}$$

$$K_{local\_vigaex2} = \begin{bmatrix} \frac{3EI}{L^3} & \frac{3EI}{L^2} \\ \frac{3EI}{L^2} & \frac{3EI}{L} \end{bmatrix}$$

$$K_{localin} = \begin{bmatrix} \frac{12EI}{L^3} & \frac{-12EI}{L^3} & \frac{6EI}{L^2} & \frac{6EI}{L^2} \\ \frac{-12EI}{L^3} & \frac{12EI}{L^3} & \frac{-6EI}{L^2} & \frac{6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{-6EI}{L^2} & \frac{4EI}{L} & \frac{2EI}{L} \\ \frac{6EI}{L^2} & \frac{-6EI}{L^2} & \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

Uma vez definidas as matrizes de rigidez locais de cada elemento é necessário proceder ao cálculo da matriz de rigidez global da estrutura com recurso a um processo de assemblagem da rigidez das diversas barras [11, 43, 3]. Tomou-se como exemplo uma viga com cinco nós, representada na figura 4.6, onde foram considerados dois graus de liberdade, translação vertical e rotação, por cada nó. Sendo que nas zonas de apoio a translação vertical não é possível, os nós extremos consideram apenas os graus de liberdade de rotação. Todos os elementos de barra possuem as mesmas características (comprimento, área de secção transversal e módulo de elasticidade).

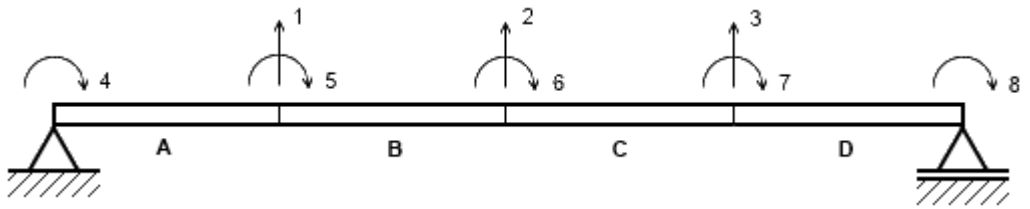


Figura 4.6: Graus de liberdade no modelo da viga.

Os elementos extremos possuem cada um 3 graus de liberdade, o que corresponde a uma matriz de rigidez local de dimensões *3x3* e os elementos interiores possuem 4 graus de liberdade cada, o que corresponde a uma matriz de rigidez local de dimensões *4x4* para cada um deles. As forças de fixação dos elementos extremos podem ser obtidas através do caso encastrado-rotulado, diminuindo assim a dimensão da matriz para *2x2* já que o grau de liberdade de rotação no apoio não é contabilizado. As matrizes destes elementos são as matrizes locais escritas anteriormente. A matriz de rigidez global, depois de feita a assemblagem, terá uma dimensão *8x8*, correspondente ao número de graus de liberdade.

A assemblagem é conseguida através da função *Build_Global_K_Viga*, que possui como parâmetros de entrada a rigidez de flexão EI e o número de elementos, onde é percorrido um ciclo *for* que atualiza a matriz em função do tipo de elemento em causa. Cada elemento extremo é responsável pela atualização de 4 valores da matriz de rigidez global (devido à sua matriz local de dimensões *2x2*) e cada elemento interno é responsável pela atualização de 16 valores (dada a sua matriz local de dimensões *4x4*). O grau de liberdade horizontal no apoio da direita não é contabilizado por ter sido admitida a rigidez axial, EA, como infinita.

O mesmo raciocínio é válido para uma estrutura subdividida num número diferente de elementos, sendo a dimensão na matriz de rigidez global equivalente ao número de graus de liberdade da estrutura.

Foram definidos por convenção os elementos barra da figura 5.6 de A a D da esquerda para a direita. Atendendo à numeração global dos graus de liberdade, as matrizes dos elementos, num contexto global, apresentam a forma que se segue,

$$K^A = \begin{bmatrix} A_{11} & 0 & 0 & A_{14} & A_{15} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_{41} & 0 & 0 & A_{44} & A_{45} & 0 & 0 & 0 \\ A_{51} & 0 & 0 & A_{54} & A_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^B = \begin{bmatrix} B_{11} & B_{12} & 0 & 0 & B_{15} & B_{16} & 0 & 0 \\ B_{21} & B_{22} & 0 & 0 & B_{25} & B_{26} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{51} & B_{52} & 0 & 0 & B_{55} & B_{56} & 0 & 0 \\ B_{61} & B_{62} & 0 & 0 & B_{65} & B_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{22} & C_{23} & 0 & 0 & C_{26} & C_{27} & 0 \\ 0 & C_{32} & C_{33} & 0 & 0 & C_{36} & C_{37} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{62} & C_{63} & 0 & 0 & C_{66} & C_{67} & 0 \\ 0 & C_{72} & C_{73} & 0 & 0 & C_{76} & C_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D_{33} & 0 & 0 & 0 & D_{37} & D_{38} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D_{73} & 0 & 0 & 0 & D_{77} & D_{78} \\ 0 & 0 & D_{83} & 0 & 0 & 0 & D_{87} & D_{88} \end{bmatrix}$$

Onde a designação de cada uma das matrizes se apresenta simplificada pelos símbolos K^A , K^B, K^C e K^D.

Uma vez definidas as matrizes locais de todos os elementos estão reunidas as condições para obter a matriz de rigidez global da estrutura, que apresenta a seguinte forma:

$$K^{GLOBAL} = K^A + K^B + K^C + K^D = \tag{4.1}$$

$$= \begin{bmatrix} A_{11}+B_{11} & B_{12} & 0 & A_{14} & A_{15}+B_{15} & B_{16} & 0 & 0 \\ B_{21} & B_{22}+C_{22} & C_{23} & 0 & B_{25} & B_{26}+C_{26} & C_{27} & 0 \\ 0 & C_{32} & C_{33}+D_{33} & 0 & 0 & C_{36} & C_{37}+D_{37} & D_{38} \\ A_{41} & 0 & 0 & A_{44} & A_{45} & 0 & 0 & 0 \\ A_{51}+B_{51} & B_{52} & 0 & A_{54} & A_{55}+B_{55} & B_{56} & 0 & 0 \\ B_{61} & B_{62}+C_{62} & C_{63} & 0 & B_{65} & B_{66}+C_{66} & C_{67} & 0 \\ 0 & C_{72} & D_{73} & 0 & 0 & C_{76} & C_{77}+D_{77} & D_{78} \\ 0 & 0 & D_{83} & 0 & 0 & 0 & D_{87} & D_{88} \end{bmatrix}$$

O armazenamento dos termos de todos os elementos na matriz de rigidez global é feito de uma forma sequencial particular, que, quando explorada da melhor maneira pode conduzir a melhorias no consumo de recursos informáticos, nomeadamente a diminuição da quantidade de memória necessária e a redução do número de operações de cálculo. O facto de a matriz ser simétrica permite evitar as operações de cálculo associadas a um dos triângulos superior ou inferior da matriz no *sofware* utilizado. Este procedimento de cálculo foi utilizado para cada um dos casos, diferenciados pelo número de elementos finitos, através da definição da função *Build_GDL* que permite definir as posições afetadas na matriz de rigidez global por cada elemento das matrizes locais. Para o exemplo dado anteriormente esta matriz tomará a forma apresentada seguidamente, onde cada linha diz respeito aos graus de liberdade afetados por cada elemento:

$$GDL = \begin{bmatrix} 0 & 1 & 4 & 5 \\ 1 & 2 & 5 & 6 \\ 2 & 3 & 6 & 7 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

Para garantir a compatibilidade da matriz GDL com a posição dos elementos na matriz de rigidez global foi necessário considerar como existentes a posição dos graus de liberdade de rotação nos apoios. Isto traduz-se, para o exemplo anterior, numa matriz de rigidez global com dimensões *8x8* com duas linhas e duas colunas de zeros, com a forma da matriz K^{GLOBAL} apresentada no exemplo anterior.

Esta é uma matriz singular (com determinante nulo), que torna impossível as operações aritméticas com outras matrizes. Para resolver este problema, foi definida uma função, *[][] RemoveZeros*, que elimina estas linhas e colunas reduzindo a dimensão da matriz de rigidez global para *[(2*elementos)-2] [(2*elementos)-2]*, o que corresponde a uma dimensão de *4x4* no caso da viga com 3 elementos.

A massa da estrutura é representada através da função *Build_Massa_Viga*, que define uma matriz diagonal com uma massa por elemento correspondente a 1/elementos, tem uma dimensão igual ao número de GDL verticais. Assim sendo, para o exemplo anterior, esta matriz terá uma dimensão *3x3*, em que cada GDL vertical possui $\frac{1}{4}$ da massa total e a restante é associada aos apoios, não tendo influência nos GDL interiores. A figura 4.7 pretende ilustrar esta situação:

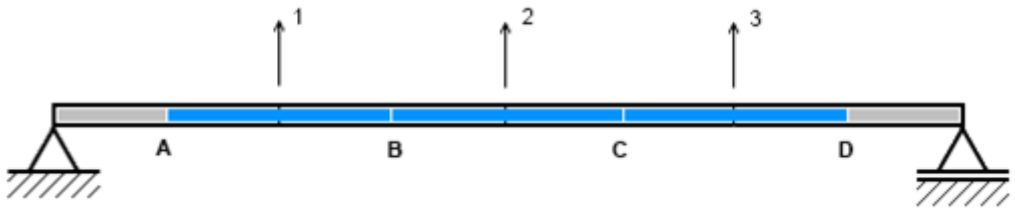


Figura 4.7: Distribuição da massa por GDL no modelo da viga.

A matriz de massa tomaria a seguinte forma:

$$M_{viga} = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}$$

A matriz de massa terá a mesma dimensão da matriz de rigidez global condensada, que será explicada adiante.

4.1.3 Rigidez e Massa no Modelo da Consola

Tal como referido no caso da viga bi-apoiada, foi necessário definir as matrizes de rigidez local do elemento junto ao encastramento e dos elementos internos através das tabelas de forças de fixação. Os elementos da consola podem ser todos definidos através da mesma matriz local, sendo que o elemento junto ao encastramento, por ter menos graus de liberdade, apenas contribui com 4 destes termos para a rigidez global da estrutura. Esta matriz, definida pela função *Build_Klocal_in*, com variáveis de entrada EI e L, é a mesma que foi definida para os elementos interiores no modelo da viga e é dada por:

$$K_{localin} = \begin{bmatrix} \frac{12EI}{L^3} & \frac{-12EI}{L^3} & \frac{6EI}{L^2} & \frac{6EI}{L^2} \\ \frac{-12EI}{L^3} & \frac{12EI}{L^3} & \frac{-6EI}{L^2} & \frac{6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{-6EI}{L^2} & \frac{4EI}{L} & \frac{2EI}{L} \\ \frac{6EI}{L^2} & \frac{-6EI}{L^2} & \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

Similarmente ao caso da viga bi-apoiada, é tomado como exemplo, para o caso da consola, uma barra com 4 nós onde existem os graus de liberdade de translação vertical e rotação, como ilustrado na figura 4.8. Neste caso o nó junto ao encastramento não possui liberdade de movimento e portanto não é necessário na definição da matriz de rigidez global. Os elementos barra possuem também as mesmas características referidos no caso da viga.

O elemento junto ao encastramento possui 2 graus de liberdade e, portanto, é caracterizado por uma matriz de rigidez local de dimensões *2x2*, enquanto que os restantes elementos têm matrizes de rigidez local de dimensões *4x4*, por terem cada um 2 graus de liberdade de translação vertical e 2 graus de liberdade de rotação. Este exemplo de uma consola de 4 elementos, com 8 graus de liberdade no total, também terá uma matriz de rigidez global de dimensões *8x8*.

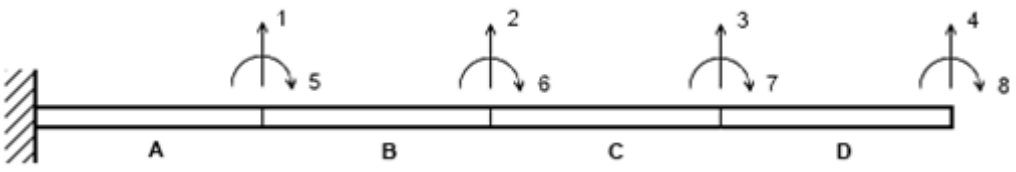


Figura 4.8: Graus de liberdade no modelo da consola.

A assemblagem é feita por um processo idêntico ao referido no caso da viga, mas desta vez através da função *Build_Global_K_Consola*, que percorre também um ciclo *for* idêntico ao da viga mas desta vez com uma condição apenas para o elemento extremo junto ao encastramento.

Da mesma forma que o exemplo anterior, foram definidos por convenção os elementos barra de A a D. Atendendo à numeração global dos graus de liberdade, as matrizes dos elementos apresentam a forma que se segue,

$$K^A = \begin{bmatrix} A_{11} & 0 & 0 & 0 & A_{15} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_{51} & 0 & 0 & 0 & A_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^B = \begin{bmatrix} B_{11} & B_{12} & 0 & 0 & B_{15} & B_{16} & 0 & 0 \\ B_{21} & B_{22} & 0 & 0 & B_{25} & B_{26} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{51} & B_{52} & 0 & 0 & B_{55} & B_{56} & 0 & 0 \\ B_{61} & B_{62} & 0 & 0 & B_{65} & B_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{22} & C_{23} & 0 & 0 & C_{26} & C_{27} & 0 \\ 0 & C_{32} & C_{33} & 0 & 0 & C_{36} & C_{37} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{62} & C_{63} & 0 & 0 & C_{66} & C_{67} & 0 \\ 0 & C_{72} & C_{73} & 0 & 0 & C_{76} & C_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K^D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D_{33} & D_{34} & 0 & 0 & D_{37} & D_{38} \\ 0 & 0 & D_{43} & D_{44} & 0 & 0 & D_{47} & D_{48} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D_{73} & D_{74} & 0 & 0 & D_{77} & D_{78} \\ 0 & 0 & D_{83} & D_{84} & 0 & 0 & D_{87} & D_{88} \end{bmatrix}$$

Onde a designação de cada uma das matrizes se apresenta simplificada pelos símbolos K^A , K^B, K^C e K^D.

A matriz de rigidez global fica então definida da seguinte forma:

$$K^{GLOBAL} = K^A + K^B + K^C + K^D = \tag{4.2}$$

$$= \begin{bmatrix} A_{11}+B_{11} & B_{12} & 0 & 0 & A_{15}+B_{15} & B_{16} & 0 & 0 \\ B_{21} & B_{22}+C_{22} & C_{23} & 0 & B_{25} & B_{26}+C_{26} & C_{27} & 0 \\ 0 & C_{32} & C_{33}+D_{33} & D_{34} & 0 & C_{36} & C_{37}+D_{37} & D_{38} \\ 0 & 0 & D_{43} & D_{44} & 0 & 0 & D_{47} & D_{48} \\ A_{51}+B_{51} & B_{52} & 0 & 0 & A_{55}+B_{55} & B_{56} & 0 & 0 \\ B_{61} & B_{62}+C_{62} & C_{63} & 0 & B_{65} & B_{66}+C_{66} & C_{67} & 0 \\ 0 & C_{72} & D_{73} & D_{74} & 0 & C_{76} & C_{77}+D_{77} & D_{78} \\ 0 & 0 & D_{83} & D_{84} & 0 & 0 & D_{87} & D_{88} \end{bmatrix}$$

O preenchimento sequencial da matriz de rigidez global é feito da mesma forma que no caso da viga, através da função *Build_GDL*, e esta vai aceder aos termos da mesma matriz local para todos os elementos (os termos da matriz local do elemento junto ao encastramento são termos da matriz local de um elemento interno). Para o exemplo anterior, esta tomará então a seguinte forma:

$$GDL = \begin{bmatrix} 0 & 1 & 4 & 5 \\ 1 & 2 & 5 & 6 \\ 2 & 3 & 6 & 7 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

À semelhança da função que define a matriz de massa da viga, a função *Build_Massa_Consola* constrói uma matriz diagonal com dimensão correspondente ao número de GDL verticais. Nesta matriz, de dimensão *4x4*, é feita uma correção no último elemento por este só possuir metade da massa dos outros elementos associada ao GDL. No elemento A, metade da sua massa fica associada ao encastramento. A figura 4.9, referente ao exemplo anterior, pretende ilustrar a distribuição da massa por cada um dos GDL:

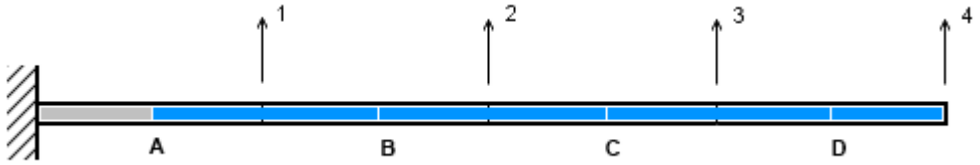


Figura 4.9: Distribuição da massa por GDL no modelo da consola.

Em que a matriz de massa seria dada por:

$$M_{consola} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix}$$

4.1.4 Matriz Dinâmica, Valores Próprios, Vectores Próprios e Curvaturas

Depois de obtida a matriz de rigidez global é feita uma condensação desta, operação que permite diminuir a dimensão de uma matriz, porque apenas interessam para a análise das curvaturas os deslocamentos verticais, transversais ao desenvolvimento do modelo. Esta matriz condensada terá então uma dimensão equivalente ao número de graus de liberdade verticais. A operação que permite obter a matriz condensada é definida pela função *Matrix Build_Cond_K*, e corresponde à seguinte fórmula:

$$[K] = K_{tt} - K_{rt}^{T} \times k_{rr}^{-1} \times k_{rt} \tag{4.3}$$

Em que,

K_{tt} são os termos da matriz de rigidez global correspondentes aos GDL de translação ;
K_{rt}^{T} é a matriz transposta dos termos da matriz de rigidez global correspondentes à relação entre os GDL de translação e de rotação;
K_{rr}^{-1} é a matriz inversa dos termos da matriz de rigidez global correspondentes aos GDL de rotação;
K_{rt} são os termos da matriz de rigidez global correspondentes à relação entre os GDL de translação e rotação.

A matriz de rigidez global comporta estes valores organizados de acordo com a seguinte ordem:

$$[K] = \begin{bmatrix} K_{tt} & K_{tr} \\ K_{rt} & K_{rr} \end{bmatrix}$$

Para fazer esta operação foi necessário importar um pacote que permitisse fazer as operações aritméticas necessárias entre as matrizes. Este pacote, *JAMA*, permite multiplicar, transpor e inverter matrizes. Para usar as operações deste pacote foi necessário definir uma função auxiliar que convertesse as variáveis *float* para variáveis *double*, por serem estas as que são compatíveis com o pacote *JAMA*. Esta função, *ConvertFF2DD*, converte matrizes de *floats* em matrizes de *doubles*. Executada esta conversão, é então possível realizar as operações entre matrizes.

É ainda importante notar que para um modelo com o mesmo número de elementos, as matrizes de rigidez condensada e de massa terão sempre uma dimensão igual, de *[elementos-1][elementos-1]* no caso da viga e de *[elementos] [elementos*] no caso da consola.

Depois de calculadas estas matrizes é então possível, através de uma operação simples, obter a matriz dinâmica que caracteriza cada um dos sistemas. É esta matriz que vai permitir, à posteriori, obter as deformadas e as curvaturas dos *n* modos de vibração no espaço modal. A matriz dinâmica é então dada por:

$$D = K^{-1} \times M \tag{4.4}$$

Em que,

D é a matriz dinâmica;
K é a matriz de rigidez condensada;
M é a matriz de massa.

Uma vez calculada a matriz dinâmica, através do pacote JAMA é possível obter as matrizes de valores e vectores próprios, através dos comandos *eig().getD()* e *eig().getV()*, respectivamente. A matriz de valores próprios é uma matriz diagonal que apresenta as frequências de vibração de cada modo, sendo a mais baixa a frequência fundamental correspondente ao 1º modo de vibração. No *NetBeans* esta matriz foi definida como *Valores Proprios* e é calculada através da seguinte fórmula:

$$ValoresProprios = \frac{1}{\sqrt{eigenvalues(D)}} \tag{4.5}$$

A matriz de vectores próprios, obtida a partir da matriz dinâmica, apresenta os deslocamentos modais de cada modo de vibração, organizados por colunas. A coluna do 1º modo corresponderá, em posição, à coluna onde se encontra a frequência fundamental na matriz dos valores próprios. A coluna do 2º modo corresponderá à coluna da frequência imediatamente a seguir à fundamental e assim sucessivamente. Para uma dada coluna, e portanto para um dado modo de vibração, de cima para baixo, cada linha corresponde ao deslocamento de um dos elementos do modelo. Esta matriz foi definida como *Vectores Proprios* e é calculada pela fórmula seguinte:

$$VectoresProprios = eigenvectors(D) \tag{4.6}$$

Será, para ambos os modelos, a partir desta matriz que vão ser obtidas as curvaturas.

Como referido anteriormente, a matriz de vectores próprios contempla os deslocamentos modais dos elementos de cada caso de estudo, em que cada coluna corresponde a um modo de vibração. O processo de obtenção das curvaturas pressupõe o uso de um operador de diferenças finitas centradas (capítulo 3.3.1) que faz uso do deslocamento nodal de um nó, *q*, e dos dois nós que lhe são adjacentes, *q-1* e *q+1*.

$$\frac{d^2\phi}{dx^2} = \frac{\phi_{q-1} - 2\phi_q + \phi_{q+1}}{h^2} \tag{4.7}$$

Uma vez que cada botão corresponde a um elemento, não é possível por este método fazer a consideração de um ponto de deslocamento nulo na zona de encastramento (entenda-se este ponto como a extremidade esquerda do botão). Foi então necessário recorrer a um artifício que permitisse considerar um deslocamento nulo num ponto fictício, *q*, e coincidente com a zona de encastramento, para tornar possível o uso da fórmula das curvaturas. A solução para este problema passa por adicionar uma linha de zeros à matriz de vectores próprios que garanta que o deslocamento é nulo num primeiro ponto em todos os modos de vibração. Foi então definida uma variável (matriz [][]C) de forma a alterar a dimensão das matrizes *Vectores Proprios* e *Curvatura* para *[elementos+1]* e *[elementos]*, para as linhas e colunas, respectivamente.

4.1.5 Gráficos das Curvaturas

Os gráficos das curvaturas e diferenças dos modelos podem ser obtidos quando conhecidas as matrizes curvatura, C, que possuem em cada coluna os valores correspondentes à curvatura de cada modo vibração. Foi então definida uma nova classe, *Graficos_Curvaturas*, onde estão definidos os comandos necessários para plotar os gráficos. Inicialmente foi necessário importar vários pacotes necessários à definição dos elementos gráficos, que foram os seguintes:

- *javax.swing.JFrame* - permite definir a janela da interface;
- *javax.swing.JPanel* - permite, depois de existir uma Frame, adicionar elementos à janela;
- *java.awt.BorderLayout* - permite organizar e redimensionar os conteúdos da janela;
- *java.awt.Color* - permite anexar cores do tipo RGB;
- *java.awt.Polygon* - permite armazenar informação numa zona fechada com vários segmentos;
- *java.lang.Math.abs* - contém métodos que permitem executar operações numéricas básicas, como exponenciais e logaritmos por exemplo.

Foi criada então a função *Gráficos_Curvatura*, que utiliza a classe *JFrame* onde estão definidas como variáveis de entrada as curvaturas, o modo de vibração e as frequências correspondentes aos modelos nominal e danificado para poderem ser definidos os gráficos.

Uma vez que o Java não permite fazer gráficos de forma automática foi necessário definir uma janela para onde serão exportados os dados dos vectores das curvaturas correspondentes a cada modo de vibração e o correspondente modo de vibração. Para disponibilizar a relação entre as curvaturas e respectivas diferenças foi primeiramente definido um referencial cartesiano bidimensional à imagem dos que foram apresentados no Capitulo 3, um eixo horizontal com a dimensão longitudinal do modelo e um eixo vertical com o valor da curvatura dos elementos. O traçado destes eixos foi feito através do comando *g.drawLine* e a legenda do gráfico através do comando *g.drawString*.

Para além das linhas e legendas foi necessário atribuir uma cor a cada um dos gráficos, do modelo intacto, danificado e das diferenças. Esta atribuição foi feita através do comando *g.setColor* e as cores foram atribuídas em correspondência com os gráficos expostos anteriormente (azul no modelo nominal, vermelho para o danificado e verde para as diferenças). A figura 4.10 ilustra a disposição destas componentes.

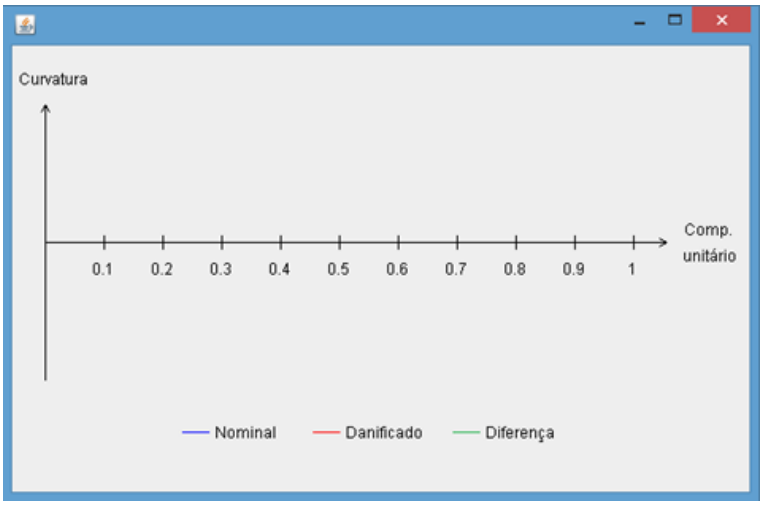


Figura 4.10: Eixos cartesianos e legenda.

As variáveis intervenientes (*n*, *d*, *i*, *MV*, *fn_n* e *fn_d*) têm o seu valor definido na classe *Método_Curvaturas* e são utilizadas nesta classe pela função *DrawSine*, que permite abrir a janela do gráfico e fazer os *plots*.

Para garantir que os *plots* dos gráficos das curvaturas são feitos de forma perceptível foi definida uma função, *Escala_Y*, que permite ajustar a escala do vector através da fixação do maior valor deste vector como máximo da escala do gráfico. Desta forma, as curvaturas de cada modo de vibração estarão sempre à escala dos eixos do gráfico. Este procedimento foi efetuado para ambos os modelos, nominal e danificado. Da mesma maneira foi necessário ajustar da curvatura de cada modo de vibração à escala do eixo horizontal.

Uma vez que também serão plotadas informações relativas às frequências naturais, foi ainda nesta classe definida uma função, *round*, que permite arredondar o seu valor, que correspondem aos valores da diagonal principal da matriz de valores próprios, para 3 casas decimais.

4.2 Manual do Utilizador

O utilizador pode aceder a esta aplicação através do *website* da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa, onde pode fazer uso do programa de cálculo e consultar o seu código fonte.

A interface desta aplicação foi desenvolvida por forma a ser o mais simples e prática possível. Como já foi explicado anteriormente no Manual Teórico foram desenvolvidos 6 modelos onde será possível estudar o método das curvaturas associado aos casos da viga bi-apoiada e viga em consola. Cada um dos casos tem 3 hipóteses de estudo distintas que se distinguem entre si pelo número de elementos constituintes do modelo.

Na figura 4.11 encontra-se representada a aparência da interface do programa quando é aberto pelo utilizador.

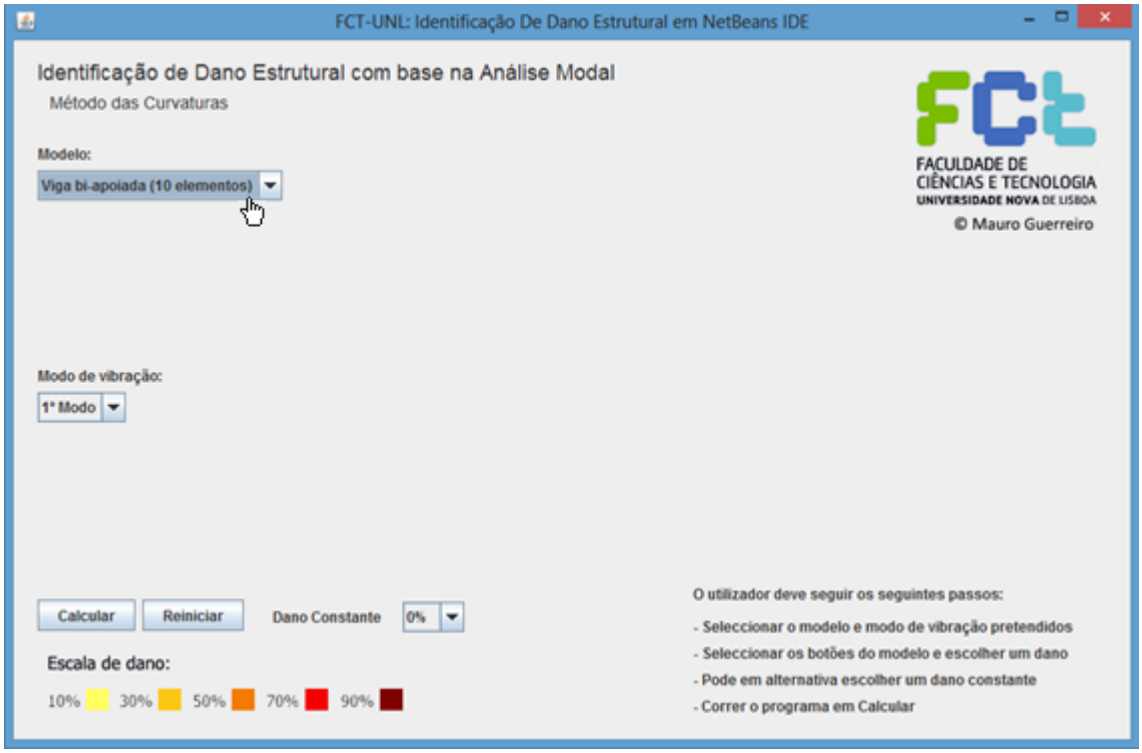


Figura 4.11: Interface inicial do programa da *Java Applet*.

No canto inferior direito existe uma lista com os passos básicos de funcionamento do programa que pretendem direcionar o utilizador no primeiro contacto. Em cima, à esquerda, existe uma *combobox* onde estão reunidos os 6 modelos de estudo, sendo aqui que o utilizador começa por escolher o que pretende estudar. Para facilitar esta explicação será dado um exemplo de utilização num modelo escolhido de forma aleatória.

Admitindo que o utilizador escolhe o modelo da Viga bi-apoiada (20 elementos), surge no centro da interface um modelo representativo, constituído por 20 elementos, com as condições de apoio da viga bi-apoiada e com uma escala que pretende facilitar o processo de comparação entre a informação atribuída aos botões e a correspondente zona nos gráficos das curvaturas. Na figura 4.12 está representado o processo de escolha do modelo.

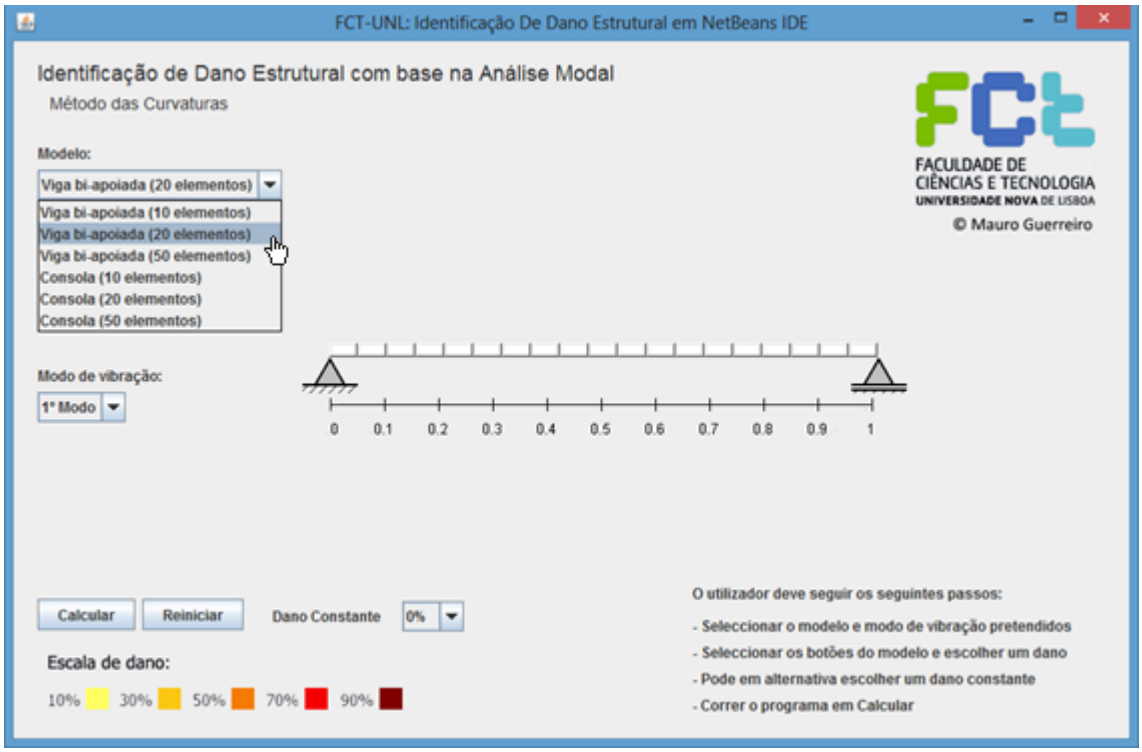


Figura 4.12: Escolha do modelo da viga bi-apoiada com 20 elementos.

De seguida, na *combobox* que se situa a baixo da que permite escolher o modelo, está outra *combobox* que permite escolher o modo de vibração do qual se pretende obter as curvaturas. Dando seguimento ao exemplo anterior, admite-se que o utilizador pretende estudar o 2^{o} modo de vibração para o modelo da viga escolhido. A figura 4.13 ilustra a escolha desta variável.

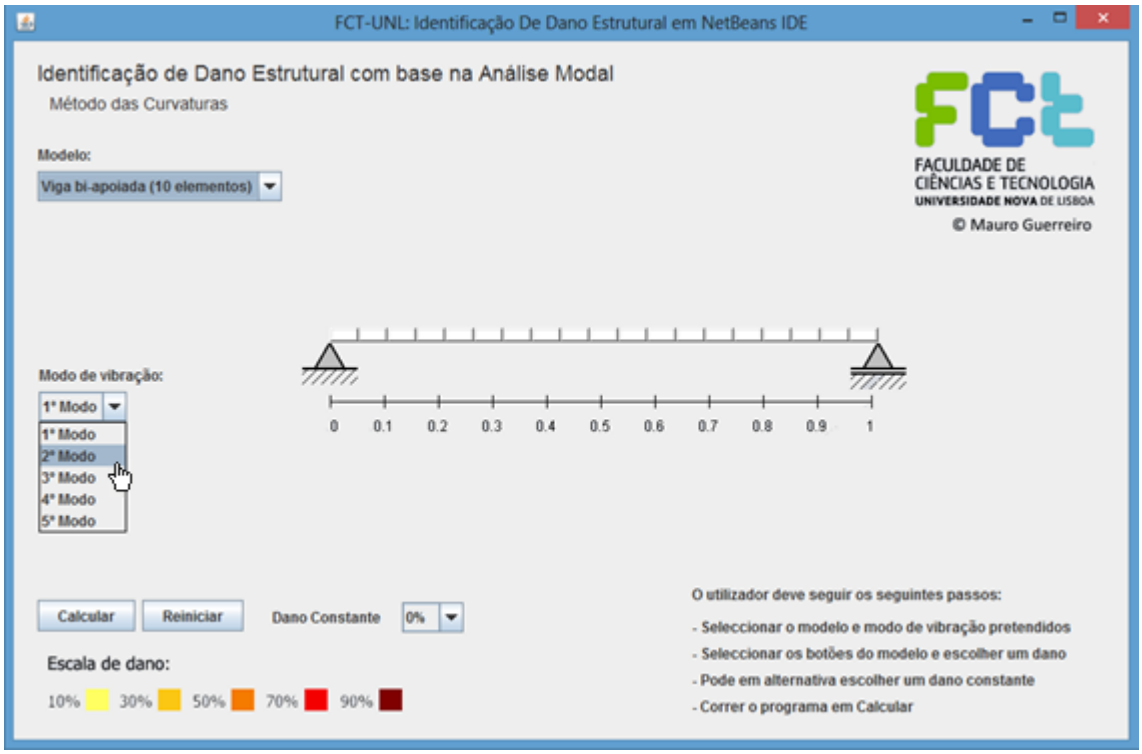


Figura 4.13: Escolha do modo de vibração.

Depois de escolhido o modelo e modo de vibração pretendidos procede-se à escolha do(s) elemento(s) que se pretende danificar. Foi dito no *Manual Teórico* que estes elementos são representados por um conjunto de botões aos quais o utilizador pode atribuir um grau de dano que pretenda. Ao carregar no botão do modelo aparece uma nova janela onde o usuário pode escolher a percentagem de dano que pretende atribuir ao elemento. Quanto maior a percentagem de dano escolhida menor será a rigidez de flexão desse elemento. Para tornar a visualização mais apelativa foi definida uma escala de cor que é atribuída ao elemento em função do dano escolhido. Continuando o exemplo anterior, admita-se que o utilizador pretende danificar um elemento a 0,2m com um dano de 50%. Depois de escolhido o dano (como foi ilustrado no *Manual Teórico*) o modelo terá o aspecto da figura 4.14.

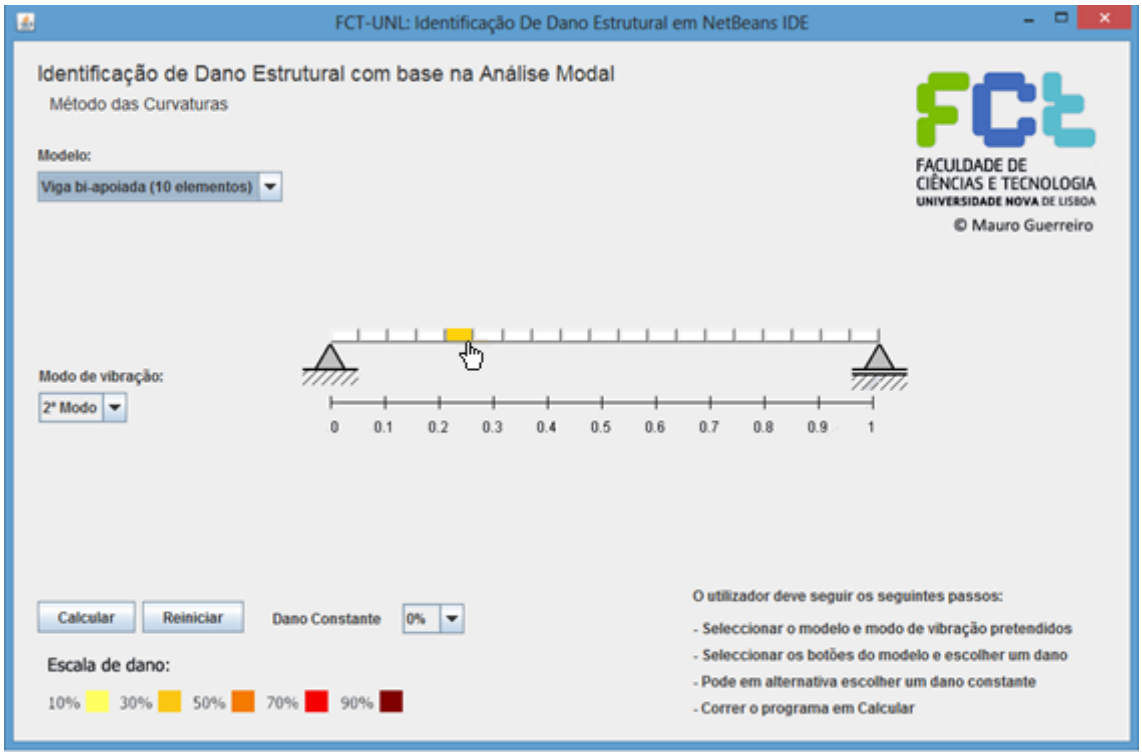


Figura 4.14: Escolha do grau de dano.

Depois de atribuído o dano à estrutura é possível correr o programa e obter os gráficos das curvaturas pretendidos. Basta então ao utilizador carregar no botão *Calcular* para que apareça uma nova janela (apresentada no capitulo anterior na secção dos gráficos) com os resultados. Na figura 4.15 encontra-se o resultado deste caso de estudo.

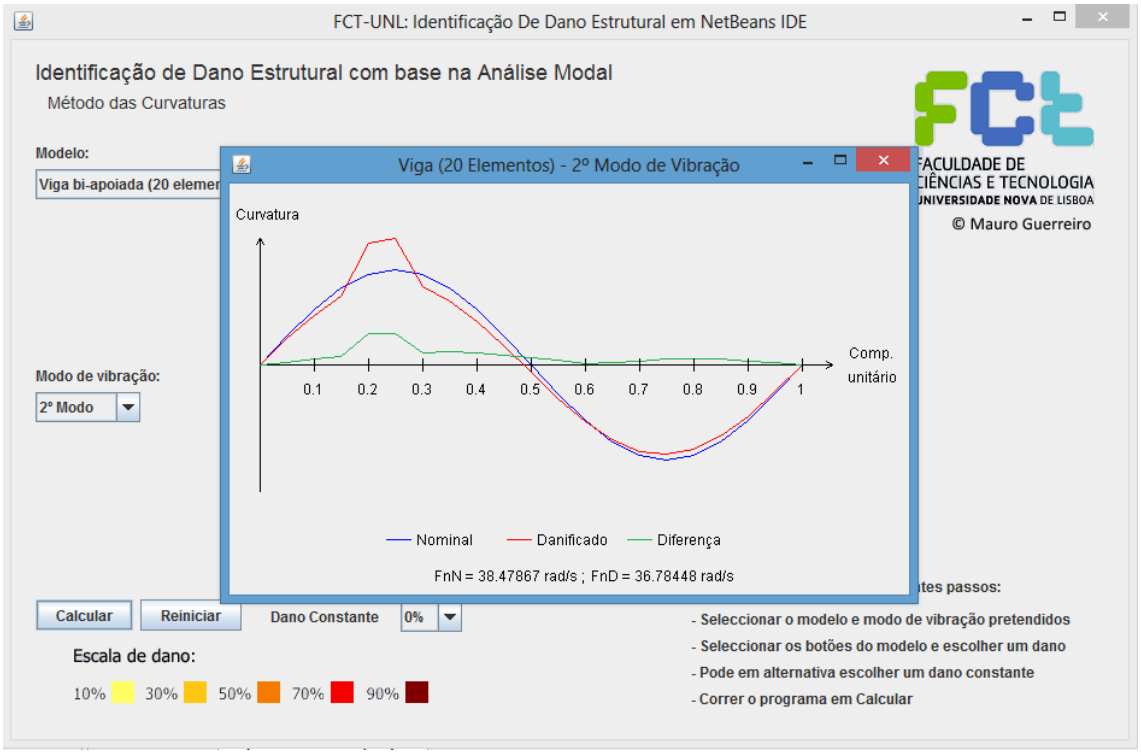


Figura 4.15: Gráficos das curvaturas com elemento danificado a 30% a 0,2m de vão.

Se o usuário eventualmente pretender introduzir outra alteração na estrutura basta selecionar a janela anterior e escolher o elemento que quer alterar. O programa foi concebido para que quando o utilizador regressa à página principal encontre o modelo com as alterações feitas antes de consultar os gráficos das curvaturas. Caso pretenda recomeçar e introduzir uma alteração diferente no mesmo modelo basta selecionar o botão Reiniciar para o programa voltar ao ponto de partida. Dando continuidade ao raciocínio anterior, admita-se que o utilizador pretende introduzir mais um dano na estrutura para além do que já lá está. Regressando à janela principal e repetindo o processo de atribuição de dano pode introduzir mais um dano, agora de 70% a 0,7m do vão. Este processo está representado na figura 4.16.

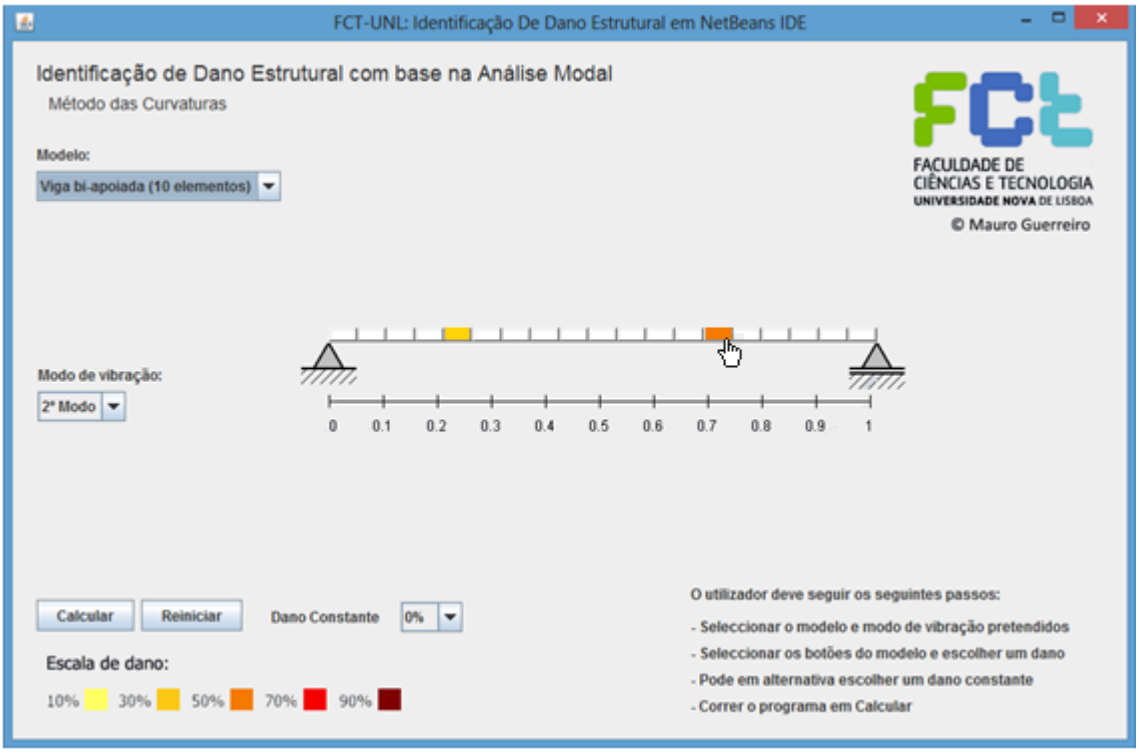


Figura 4.16: Escolha de diferentes graus de dano.

O resultado apresentado na figura 4.17 mostra a sensibilidade do método na detecção de dano. Para dois danos localizados em zonas de igual curvatura é possível distinguir distintamente qual das duas zonas está mais danificada com uma diferença de apenas 20% entre danos.

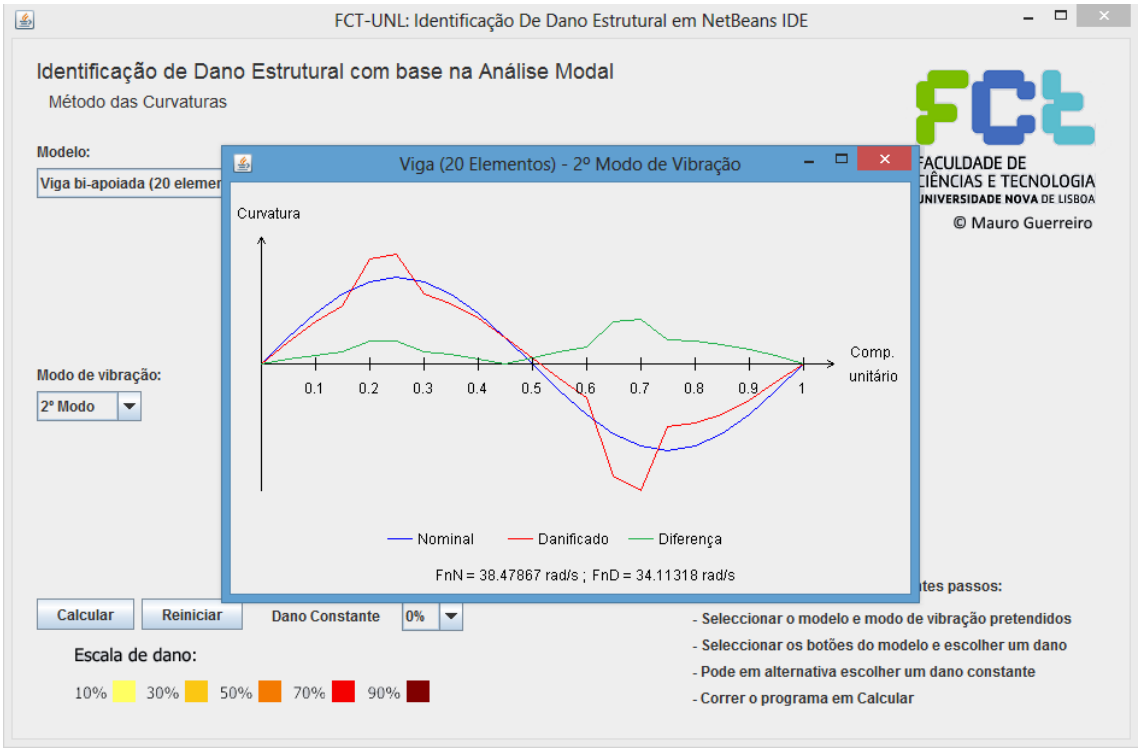


Figura 4.17: Gráficos das curvaturas com elemento danificado a 30% a 0,2m de vão e outro com 50% a 0,8m de vão.

Na aplicação é possível fazer vários cálculos e permanecer com as janelas dos resultados abertas em simultâneo. Ainda no seguimento do exemplo, se o utilizador pretende comparar este último resultado com um outro onde aumenta ainda mais o dano no elemento a 0,7 m de vão basta voltar à janela inicial, introduzir um dano de por exemplo 90% neste elemento e *Calcular* novamente. Esta ação vem ilustrada na figura 4.18, onde é possível distinguir facilmente as diferenças entre ambos os casos, comprovando mais uma vez a sensibilidade do método.

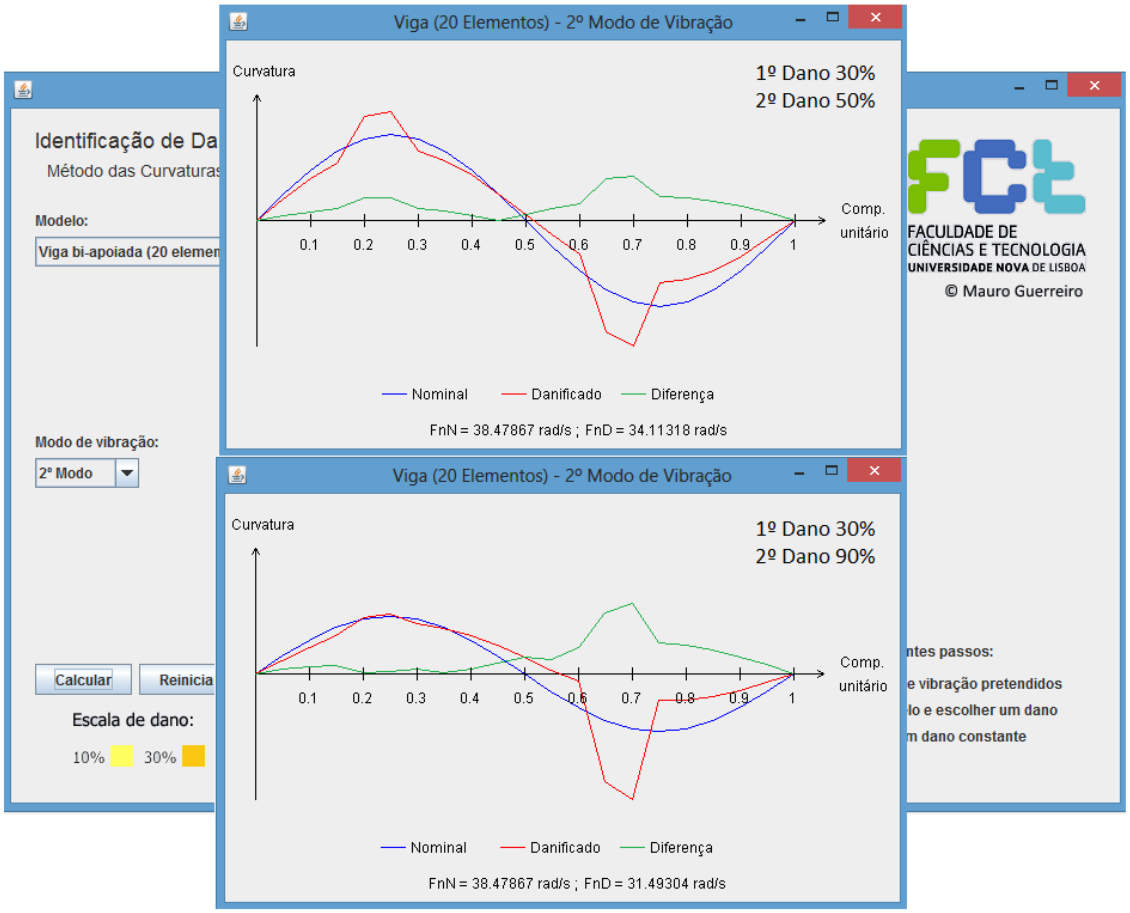


Figura 4.18: Comparação de dois resultados na *Java Applet*.

Nas figuras 4.19 e 4.20 são ainda mostrados mais 2 exemplos de uso da aplicação onde podem ser testadas as capacidades da aplicação. Como foi dito no Capitulo 3, o aumento do número de elementos do modelo aumenta o rigor da análise devido ao facto do método de obtenção das curvaturas fazer uso de um operador de diferenças finitas centradas que recorre aos pontos adjacentes ao elemento para obter a curvatura no próprio ponto. No exemplo seguinte é apresentado um teste com um dano de 50% a 0,3 m de vão de uma consola com 10 elementos no seu 3º modo de vibração, e, um teste idêntico para uma consola com 50 elementos. Na figura 4.19 é mostrada a aparência da interface quando usado o modelo da consola com 10 elementos, enquanto na figura 4.20 já são apresentados os resultados dos modelos referidos.

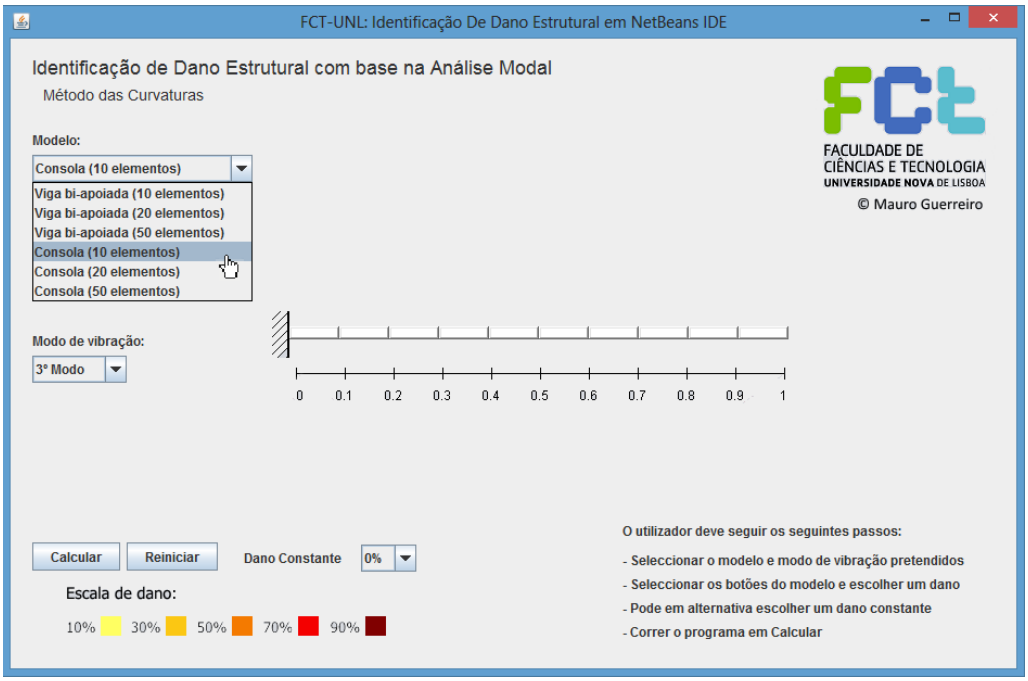


Figura 4.19: Escolha do modelo da consola com 10 elementos.

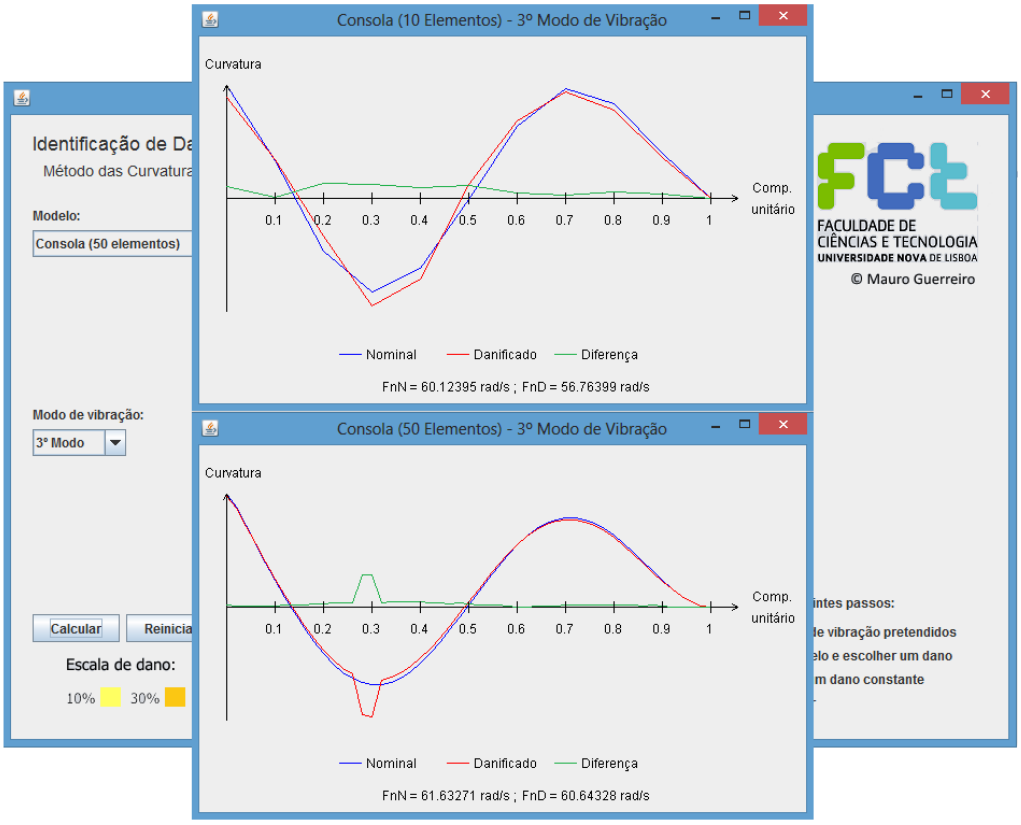


Figura 4.20: Comparação entre modelo com 10 e 50 elementos da consola.

A análise destes resultados comprova os resultados obtidos na análise feita através do *SAP2000*. É possível identificar uma zona danificada no modelo com 10 elementos mas não é possível dizer com rigor qual o sítio exato do dano, ao passo que no modelo com 50 elementos é perfeitamente identificável.

Seguidamente, na figura 4.21, encontra-se um exemplo do funcionamento da opção Somatório e para isso foi aplicado um dano de 70% em 2 dos 50 elementos de uma viga bi-apoiada. Esta diferença entre os somatórios dos modos dos modelos permite identificar estes danos situados a 0,2 e 0,7 m do vão, respectivamente.

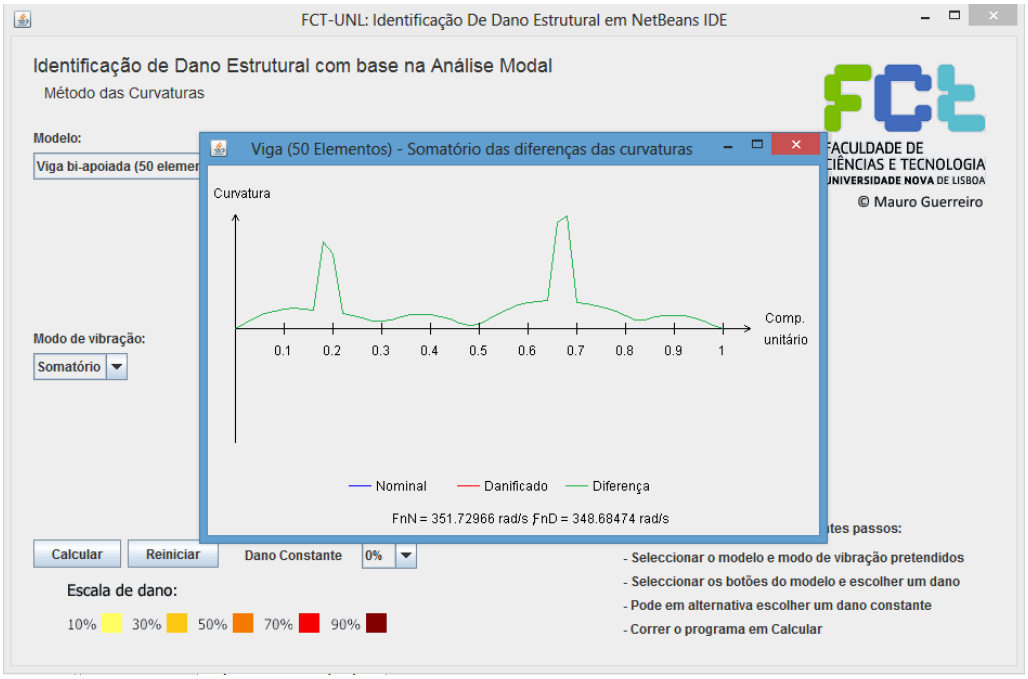


Figura 4.21: Somatório das diferenças absolutas numa viga com 2 danos de 70%.

Finalmente, o último recurso do programa é uma *combobox*, situada ao lado do botão **Reiniciar**, que permite introduzir um dano constante (com uma percentagem das possíveis na escala de dano) ao longo de todo o modelo. Esta situação vem ilustrada na figura 4.22, onde se admitiu como exemplo o 5º modo de vibração de uma viga de 50 elementos danificada a 70%.

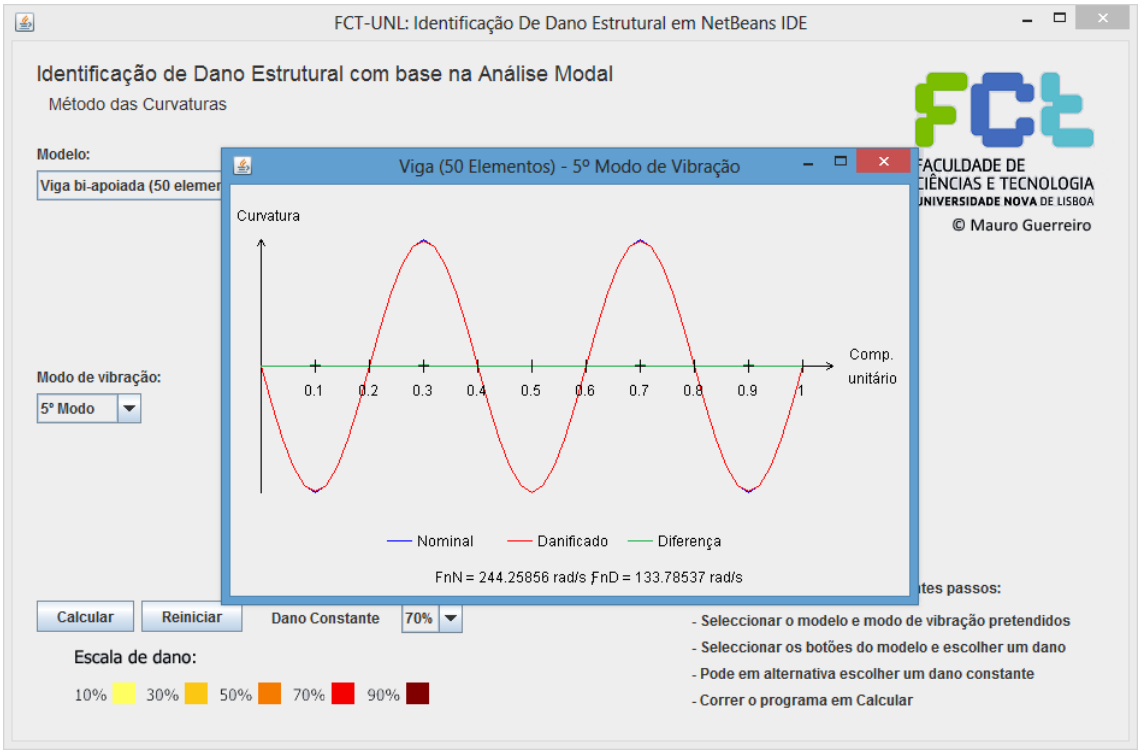
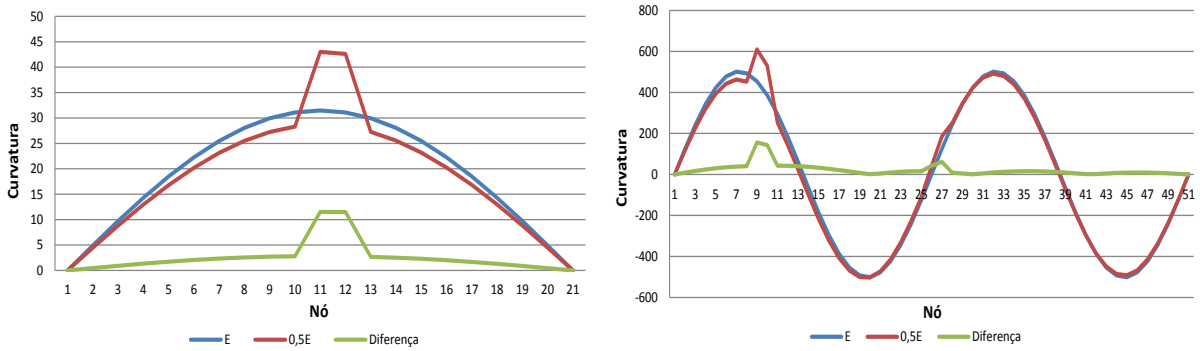


Figura 4.22: Viga de 50 elementos com dano constante de 70%.

A análise do resultado pode, à partida, levantar algumas dúvidas uma vez que não há distinção entre as curvaturas do modelo nominal e do modelo danificado, aparentando nada ter ocorrido. Na verdade, enquanto o programa processou os dados introduziu várias alterações ao modelo danificado, diminuindo a rigidez de flexão para 30% da inicial e influenciando todos os resultados até aos vectores próprios. Acontece que a rigidez de flexão diminui proporcionalmente em todos os elementos e, portanto, embora aumentem os deslocamentos modais por a estrutura ser menos rígida, aumentam com a mesma proporção. Uma vez que as curvaturas são obtidas através da segunda derivada destes vectores próprios, e esta derivada é feita através das diferenças finitas centradas, o que interessa é a relação entre estes valores e não os valores em si, dai a curvatura ser adimensional. A diferença entre estes dois modelos é apenas perceptível nas frequências dos modos de vibração, que sofrem grandes alterações, neste caso passando de 244,26 rad/s para 173,79 rad/s.

4.3 Comparação de resultados do *SAP2000* com a *Java Applet*

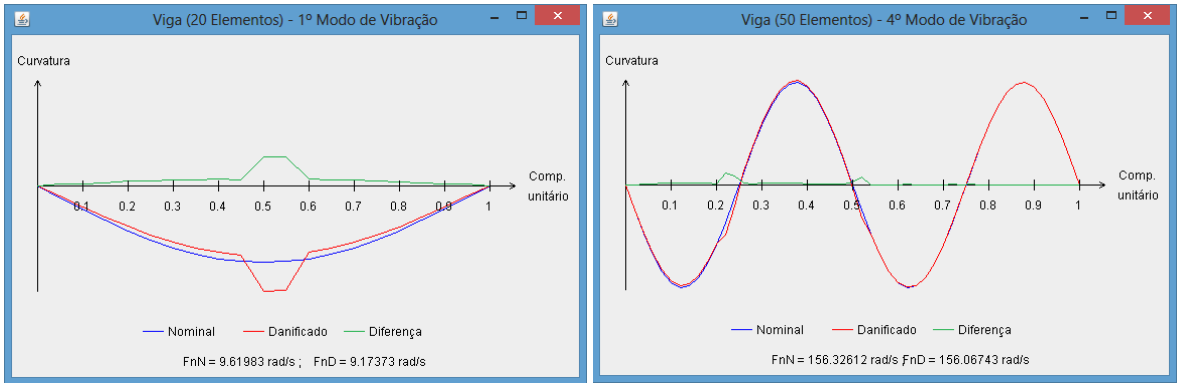
Nas duas figuras seguintes é feita uma comparação entre os resultados obtidos através do *SAP2000* e da *Java Applet*, através de um modelo da viga com 20 elementos e um dano e um com 50 elementos e dois danos:



(a) Curvaturas do 1º modo com dano de 50 % a meio vão.

(b) Curvaturas do 4º modo com 2 danos de 50%.

Figura 4.23: Resultados obtidos no *SAP2000*.



(a) Curvaturas do 1º modo com dano de 50 % a meio vão.

(b) Curvaturas do 4º modo com 2 danos de 50%.

Figura 4.24: Resultados obtidos na *Java Applet*.

Ainda que o andamento das curvaturas seja simétrico em ambos os casos isto nada significa, uma vez que a análise é feita através da comparação do valor das diferenças absolutas em módulo entre os modelos intacto e danificado. A análise destas diferenças, obtidas por um e outro processo, permite afirmar que os resultados são idênticos e portanto serve como garantia da qualidade da aplicação.

Capítulo 5

Conclusões e Desenvolvimentos Futuros

Neste capítulo, numa primeira seção, são apresentadas as conclusões gerais referentes ao trabalho desenvolvido com os métodos de detecção de dano e, numa segunda seção, analisadas as hipóteses de trabalho futuro no âmbito da identificação de dano e do software desenvolvido.

5.1 Conclusões

No capítulo 4 foram já retiradas algumas conclusões relativas aos resultados obtidos, tanto para o método das curvaturas como para os testes realizados com as frequências naturais, com os deslocamentos modais e com os indicadores estatísticos, pelo que, nesse sentido, apenas será feita uma pequena referencia às conclusões já citadas.

No que diz respeito ao método das curvaturas, os resultados melhoram com o aumento do refinamento dos modelos e que os danos são mais facilmente identificáveis em zonas onde as curvaturas são maiores (para um dado modo de vibração). Foi também possível constatar que os resultados da aplicação do método são mais satisfatórios em estruturas lineares como a viga e a consola do que modelos do tipo porticado e, que a sua aplicação acumula mais erros quando comparadas as curvaturas nos modos mais altos.

Relativamente aos restantes métodos, foi visto que tanto o método de comparação de frequências como o método de comparação de deslocamentos permitem identificar a existência de dano em modelos de elementos finitos mas não possuem capacidade de obter resultados consistentes relativos à sua localização e gravidade. Os indicadores estatísticos, *MAC* e *COMAC*, à semelhança dos anteriores, também eles foram capazes de detectar pequenas variações na rigidez estrutural sem apresentarem resultados relativamente à localização e estado de degradação estrutural.

O programa de cálculo desenvolvido como teste às capacidades do método das curvaturas foi concluído com sucesso. Os resultados obtidos são coerentes com os princípios supramencionados e os danos facilmente identificáveis. Esta aplicação constitui uma ferramenta auxiliar de uso simples para qualquer utilizador que pretenda conhecer os contornos de funcionamento deste método de identificação modal.

Como conclusão final, este é um método que, quando aplicado em ambientes controlados, é muito confiável no que diz respeito à identificação de danos. Este permite com alguma segurança caracterizar o estado de degradação estrutural e identificar/localizar anomalias numa estrutura simples.

5.2 Desenvolvimentos Futuros

Uma das questões fundamentais na problemática de detecção de dano através da análise modal é a sua dependência de modelos analíticos já existentes e, não menos importante, o pressuposto de que existem dados de testes estruturais da estrutura intacta. Para que o dano possa ser detectado e localizado, estes elementos são imprescindíveis numa análise deste tipo. Neste trabalho foram simulados testes em modelos analíticos cujo comportamento dinâmico era facilmente obtido e comparado com outros modelos com danos induzidos em zonas conhecidas. Na realidade, a esmagadora maioria das estruturas correntes não foi sujeita a este tipo de análise não dispondo de informação relativa às suas características dinâmicas enquanto intacta. Nestas circunstâncias torna-se difícil aplicar o método das curvaturas integralmente, pelo que, este seria um dos problemas a contornar no futuro. Uma das sugestões seria criar uma rotina de leituras do espectro de frequências após a concepção da estrutura, de forma a garantir um registo do seu comportamento ao longo do tempo e, desta forma, poder fazer, quando necessário, uma comparação entre os dados reunidos em cada leitura.

As alterações das condições do ambiente, como as variações de pressão e mudanças de temperatura ao longo do dia, também têm influência nas medições feitas pelos transdutores e podem comprometer os resultados dos testes. A implementação de um sistema de controlo destas variáveis será também ela uma mais valia na garantia da qualidade dos resultados.

No que diz respeito à *Java Apllet*, podem ainda ser criados novos modelos estruturais, como por exemplo o modelo do pórtico que foi desenvolvido no capitulo 3, de forma a aumentar o leque de hipóteses do utilizador.

O sistema de botões que constitui cada modelo na aplicação é restrito ao número de botões imposto pelo programador. É assim sugerido o desenvolvimento de um método que permita ao utilizador escolher o número de elementos que pretende atribuir ao modelo.

Um outro desenvolvimento futuro interessante será criar um botão na interface que permita importar ficheiros com dados de estruturas reais de forma a poderem ser comparados com os valores do modelo analítico intacto.

Para finalizar é ainda sugerida a conversão da aplicação para o formato HTML5 por ser uma linguagem processada por todos os *browsers* actuais, possibilitando que esta aplicação seja utilizada em *tablets* e *smartphones*.

Bibliografia

[1] Allemang, R.J. The modal assurance criterion, twenty years of use and abuse. *Journal of Sound and Vibration*, (2):14–23, 2003.

[2] Allemang, R.J. and L., B.D. A correlation coeficient for modal vector analysis. Proc. 1st. Int. Modal Analisis Conference, Soc. for Experimental Mech. Bethel, 1978.

[3] Azevedo, A. *Metodo dos Elementos Finitos*. Faculdade de Engenharia da Universidade do Porto, Abril de 2003.

[4] Barai, S. and Pandey, P. Vibration signature analysis using artificial neural networks. *Journal of Computing in Civil Engineering*, 9(4):259–265, 1995.

[5] Bishop, C. Neural networks and their applications. *Review of Scientific Instrumentation*, 65(6):1803–1832, 1994.

[6] Cawley, P. and Adams, R.D. 'the location of defects in structures from measurements of natural frequencies. *Journal of Strain Analysis*, vol. 4(2):49–57, 1979.

[7] Cerri, M. and Vestroni, F. 'detection of damage in beams subjected to diffused cracking. *Journal of Sound and Vibration*, vol. 234(2):28–87, 1999.

[8] Chance, J., T.G.R. and Worden, K. A simplified approach to the numerical and experimental modeling of the dynamics of a cracked beam. *Proc. of the 12th International Modal Analysis Conference*, pages 778–785, 1994.

[9] Chondras, T.G. and Dimarogonas, A.D. Identification of cracks in welded joints of complex structure'. *Journal of Sound and Vibration*, vol. 64(4):531–538, 1980.

[10] Chun, X., W.Q. and Dongmei, T. An application of data fusion technology in structural health monitoring and damage identification. Proceedings of Smart Structures and Materials 2005: Smart Sensor Technology and Measurement Systems, 2005.

[11] Cismasiu, C. *Apontamentos do Metodo dos Elementos Finitos*. Faculdade de Ciencias e Tecnologias da Universidade Nova de Lisboa, 2009. Monte de Caparica.

[12] Coelho, P. *Programacao em Java 2*. FCA - Editora Informatica, 2003.

[13] Coppolino, R. and Rubin, S. Detectability of structural failures in offshore platforms by ambient vibration monitoring. Proc. 72th Offshore Technology Conferences, 1980.

[14] Creed, S. Assessment of large engineering structures using data collected during in-service loading'. Butterworths.

[15] Creed, S. Structural assessment. the use of full and large scale testing. assessment of large engineering structures usig in-service loading. *Butterworths and Company Publishers, Limited*, vol. 19(9):52–62, 1987.

[16] Doebling, S. W. *Measurement Of Structural Flexibility Matrices For Experiments With Incomplete Reciprocity*. Ph.D. thesis, University of Colorado.

[17] Doebling, S.W., Farrar, C. R., Prime, M. B. and Shevitz, D.W. *Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in Their Vibration Characteristics: a Literature Review*. Los Alamos National Laboratory, 1996.

[18] Feldman, M. and Braun, S. Identification of nonlinear system parameters via the instantaneous frequency: Application of the hilbert transform and wigner-wille techniques. Proc. 73th International Modal Analysis Conference, 1995.

[19] Genovese, M., Bauchspiess, A., Brito, J.L.V. and Doz, G.N. *Damage Detection Using an Hibrid Formulation Between Changes in Curvature Mode Shapes and Neural Network*. Brazian Congress on Computational Mechanics, 2001.

[20] Guo, H. Structural damage detection using information fusion technique. *Mechanical Systems and Signal Processing*, 20(1):1173–1188, 2006.

[21] Iwasaki, A. and Kobayashi, H. An unsupervised statistical damage detection method for structural health monitoring. *Smart Material Structures*, 13:810–859, 2004.

[22] Ju, F.D. and Mimovich, M. 'experimental diagnosis of fracture damage in structures by the modal frequency method. ASME, 1987.

[23] Kabe, A. Stiffness matrix adjustment using mode data. *AIAA Journal*, 23(9):1431–1436, 1985.

[24] Kaouk, M. and Zimmerman, D. Assessment of damage affecting all structural properties. *Proc. of the 9th Symposium on Dynamics and Control of Large Structures*, pages 445–455, 1994.

[25] Kaouk, M. and Zimmerman, D. Structural damage assessment using a generalized minimum rank perturbation theory. *AIAA Journal*, 32(4):836–842, 1994.

[26] Kim, J.H., J.H. and Lee, C. Application of the modal assurance criteria for detecting and locating structural faults. Proc. 10th International Modal Analysis Conference, 1992.

[27] Klein, K., G.J. and Swamidas, A. Monitoring changes in modal parameters with fatigue. Proc. of the 12th International Modal Analysis Conference, 1994.

[28] Klenke, S. and Paez, T. Damage identification with probabilistic neural networks. Proc. of the 14th International Modal Analysis Conference, 1996.

[29] Ko, J. M., W.C. and Lam, H. Damage detection in steel framed structures by vibration measurement approach. Proc. of 12th lnternational Modal Analysis Conference, 1994.

[30] Law, S. S., X.L. and Ward, H.S. 'a vibration technique for structural stiffness identification. International Conference on Vibration Problems in Engineering, 1990.

[31] Lieven, N. and Ewins, D.J. Spatial correlation of mode shapes, the coordinate modal assurance criterion (comac). Proc. 6th International Modal Analysis Conference, 1988.

[32] Lim, T.W. Nondestructive damage evaluation of a structure from limited modal parameters. *A Submatrix Approach To Stiffness Using Modal Test Data*, 28(6):1123–1130, 1990.

[33] Lin, R. and Ewins, D. On the location of structural nonlinearity-from modal testing-a feasibility study. of the 8th International Modal Analysis Conference, 1990.

[34] Manson, G., W.K. and Tomlinson, G. Pseudo-fault induction in enaineering structures. ASME Adaptive Structures and Materials Systems, 1994.

[35] Minor, C. and Gottuk, D. Data fusion with a multisensor system for damage control and situational awareness. Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.

[36] Pandey, A. K., B.M. and Samman, M.M. Damage detection from changes in curvature mode shapes. *Journal of Sound and Vibration*, 145(2):321–332, 1991.

[37] Prime, M. and Shevitz, D. Linear and nonlinear methods for detecting cracks in beams. Proc. of the 74th International Modal Analysis Conference, 1996.

[38] Salawu, O. and Williams, C. Damage detection in steel framed structures by vibration measurement approach. Proc. of 12th lnternational Modal Analysis Conference, 1994.

[39] Salawu, O. and Williams, C. Damage location using vibration mode shapes. *Proc. of the 12th International Modal Analysis Conference*, pages 933–939, 1994.

[40] Salta, M. and Silva, A. *Tecnicas de Ensaio Para Inspecao de Estruturas*. LNEC, 2014.

[41] Sanders, D., K.Y. and Stubbs, R. Nondestructive evaluation of damage in composite structures using modal parameters. *Experimental Mechanics*, 32:240–251, 1992.

[42] Schulz, M.J., P.P. and Abdelnaser, A. Frequency response function assignment technique for structural damage identification. Proc. ofthe 74th International Modal Analysis Conference, 1996.

[43] Scremin, A. *Mecanica dos Solidos: um segundo curso*. Departamento de Engenharia Mecanica da UniversidadeFederal do Parana.

[44] Sierra, K. and Bates, B. *Java*. Alta Books, 2010.

[45] Solawu, O.S. Detection of Structural Damage Throught Changes in Frequency: A Review. *Engineering Structures*, vol. 19(9):718–723, 1995.

[46] Stubbs, N. and Kim, J.T. Damage localization in structures without baseline modal parameters. *AIAA Journal*, 34(8):1644–1649, 1996.

[47] Su, Z. and Chen, Z. Selection of data fusion schemes for structural damage evaluation. *Structural Health Monitoring*, 9(3):223–241, 2009.

[48] Topole, K.G. and Stubbs, N. Nondestructive damage evaluation of a structure from limited modal parameters. *Earthquake Engineering and Structural Dynamics*, 24(11):1427–1436, 1995.

[49] Topole, K.G. and Stubbs, N. Nondestructive damage evaluation of a structure from limited modal parameters. *The International Journal of Analytical and Experimental Modal Analysis*, 10(2):95–103, 1995.

[50] Trendafilova, I. and Heylen, W. Categorisation and pattern recognition methods for damage localisation from vibration measurements. *Mechanical Systems and Signal Processing*, 17(4):825–836, 2003.

[51] Tsou, P. and Shen, H. Structural damage detection and identification using neural networks. *AlAA Journal*, 32(1):176–183, 1994.

[52] Uzgider, Z., S.K. and Caglayan, D.B. Identification of railway bridges using locomotive-induced vibrations. Bridge Management 2: Conference, 1992.

[53] West, W. Illustration of the use of modal assurance criterion to detect structural changes in an orbiter test specimen. Proc. Air Force Conference on Aircraft Structural Integrity, 1984.

[54] Zhang, K. Y.. Gu, A.J. and Li, J.W. 'diagnosis of a slot fault on a frame structure. 10th International Modal Analysis Conference, 1992.

Anexo A

Gráficos das Curvaturas

O presente Anexo contempla alguns dos testes de relevância realizados no âmbito do método das curvaturas, referidos no capitulo 3. Encontram-se separados por capítulos os modelos da consola, viga, pórtico e testes comparativos entre diferentes graus de dano.

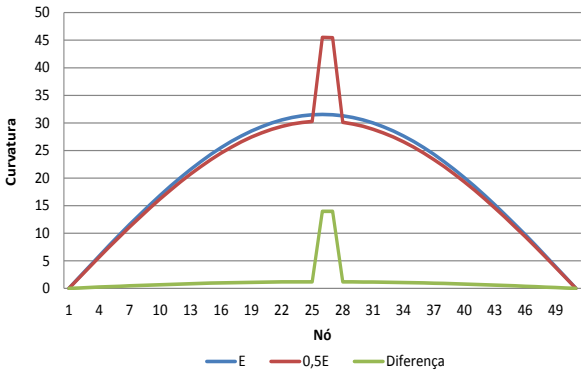
A.1 Modelo da Viga Bi-apoiada

A tabela que segue pretende ilustrar o processo de obtenção tanto dos deslocamentos como das curvaturas depois de importados os dados dos deslocamentos do *SAP2000* para o *Excel*, correspondendo a mesma ao modelo da viga com 10 elementos com um dano de 50% a meio vão.

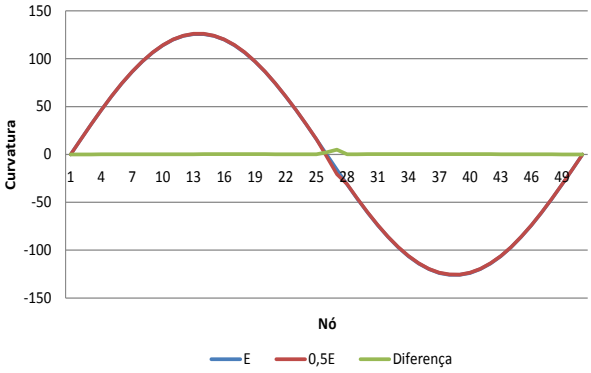
Tabela A.1: Exemplo tipo do processo de obtenção do vector das curvaturas.

Ponto	*Desloc. (E)*	*Desloc. (0,5E)*	*|dE-d0,5E|*	*Curv. (E)*	*Curv. (0,5E)*	*|cE-c0,5E|*
-	3,951	3,770	-	-	-	-
1	0,000	0,000	0,000	0,000	0,000	0,000
2	-3,951	-3,770	0,180	9,668	8,020	1,648
3	-7,515	-7,220	0,294	18,389	15,281	3,108
4	-10,343	-10,059	0,284	25,311	21,091	4,220
5	-12,159	-12,053	0,105	29,755	24,879	4,876
6	-12,785	-13,053	0,268	31,286	39,374	8,088
7	-12,159	-12,478	0,319	29,755	37,917	8,163
8	-10,343	-10,386	0,043	25,311	21,258	4,053
9	-7,515	-7,443	0,071	18,389	15,430	2,959
10	-3,951	-3,884	0,067	9,668	8,106	1,562
11	0,000	0,000	0,000	0,000	0,000	0,000
-	3,951	3,884	-	-	-	-

Os seguintes 5 gráficos constituem os resultados da comparação das curvaturas dos primeiros 5 modos de vibração do mesmo modelo de uma viga intacta e uma viga com um dano a meio vão (elemento 26), simulado através de uma diminuição da rigidez de 50% num elemento local. O último gráfico desta série apresenta o somatório da diferença entre os dois modelos.

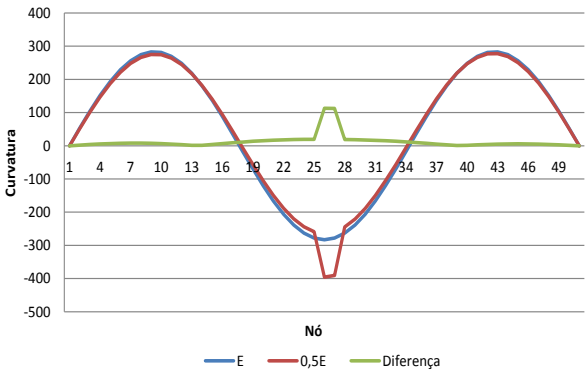


(a) Curvatura do 1º modo.

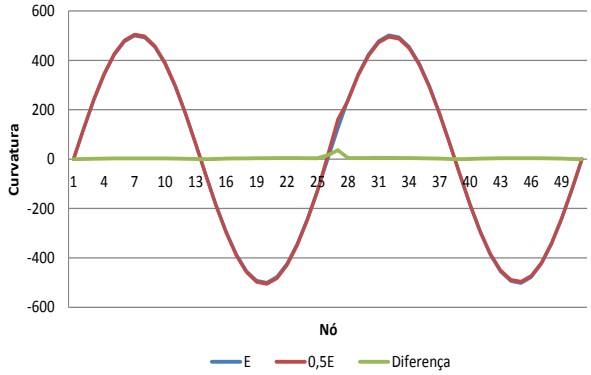


(b) Curvatura do 2º modo.

Figura A.1: Viga de 50 elementos danificada a meio vão.

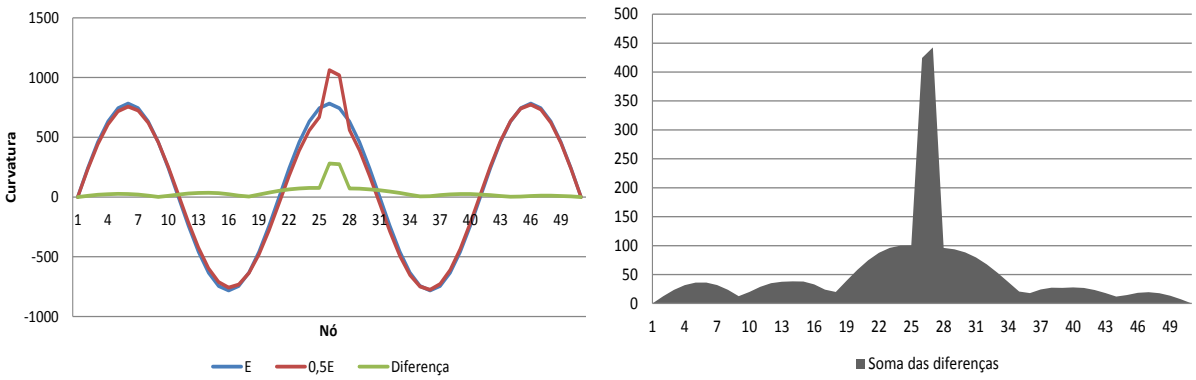


(a) Curvatura do 3º modo.



(b) Curvatura do 4º modo.

Figura A.2: Viga de 50 elementos danificada a meio vão.



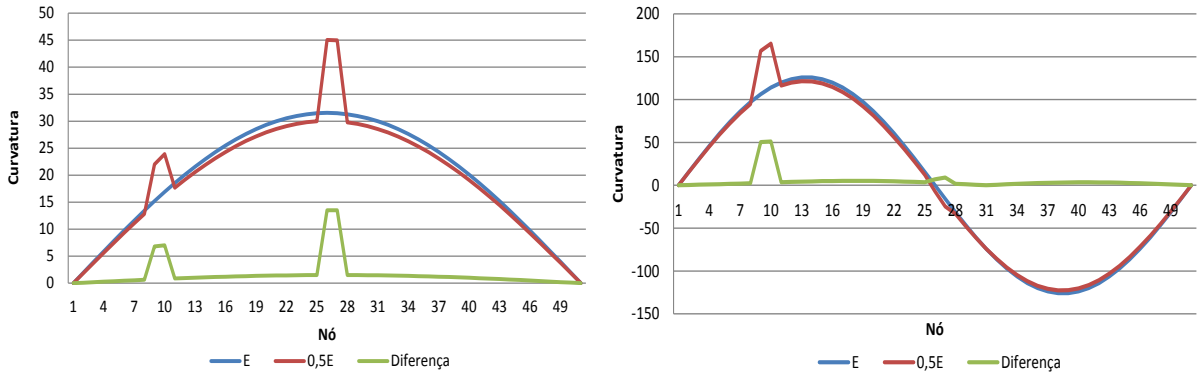
(a) Curvatura do 5º modo.

(b) Somatório das diferenças.

Figura A.3: Viga de 50 elementos danificada a meio vão

.

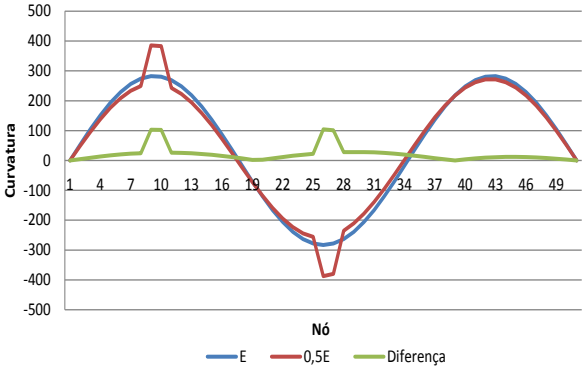
Os seguintes 5 gráficos constituem os resultados da comparação das curvaturas dos primeiros 5 modos de vibração do mesmo modelo de uma viga intacta e uma viga com 2 danos (um no elemento 9 e outro no 26), simulado através de uma diminuição da rigidez de 50% nos elementos referidos. O último gráfico desta série apresenta o somatório da diferença entre os dois modelos.



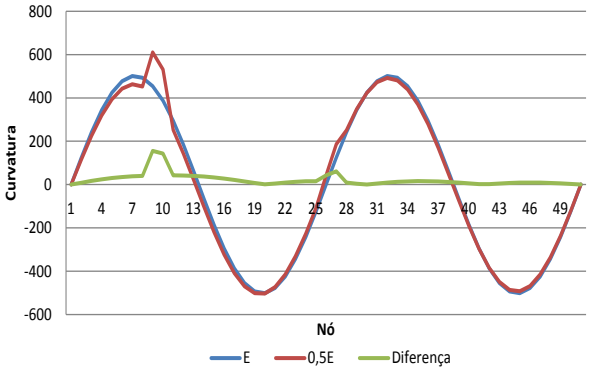
(a) Curvatura do 1º modo.

(b) Curvatura do 2º modo.

Figura A.4: Viga de 50 elementos danificada nos elementos 9 e 26.

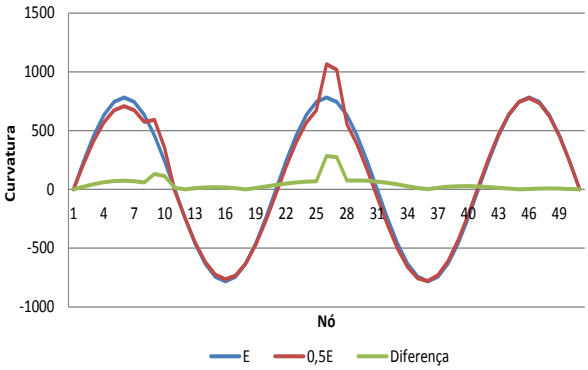


(a) Curvatura do 3º modo.

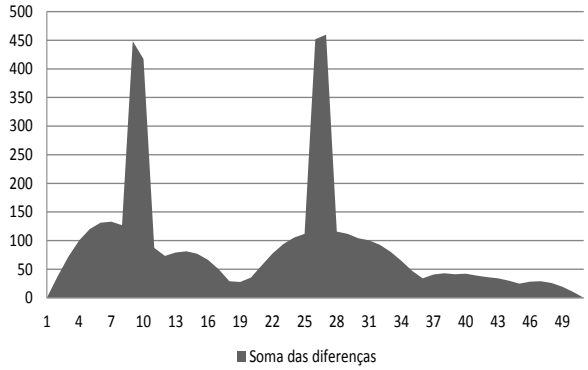


(b) Curvatura do 4º modo.

Figura A.5: Viga de 50 elementos danificada nos elementos 9 e 26.



(a) Curvatura do 5º modo.

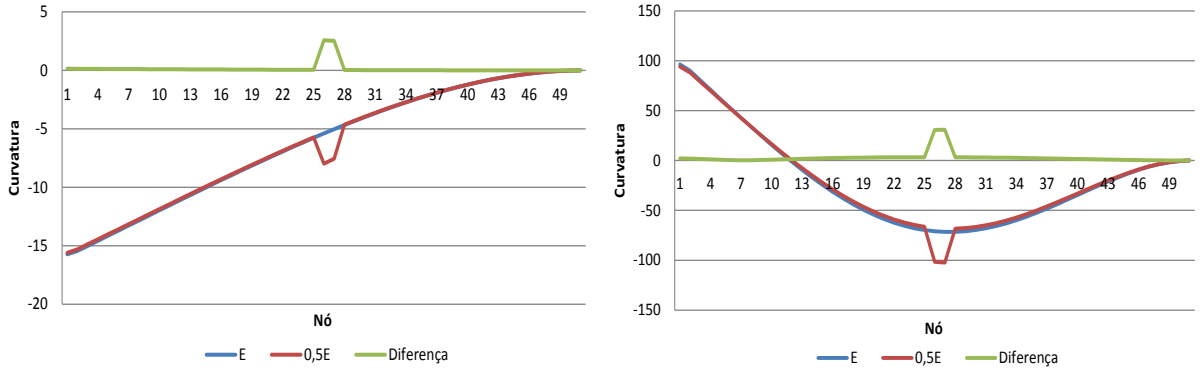


(b) Somatório das diferenças.

Figura A.6: Viga de 50 elementos danificada nos elementos 9 e 26.

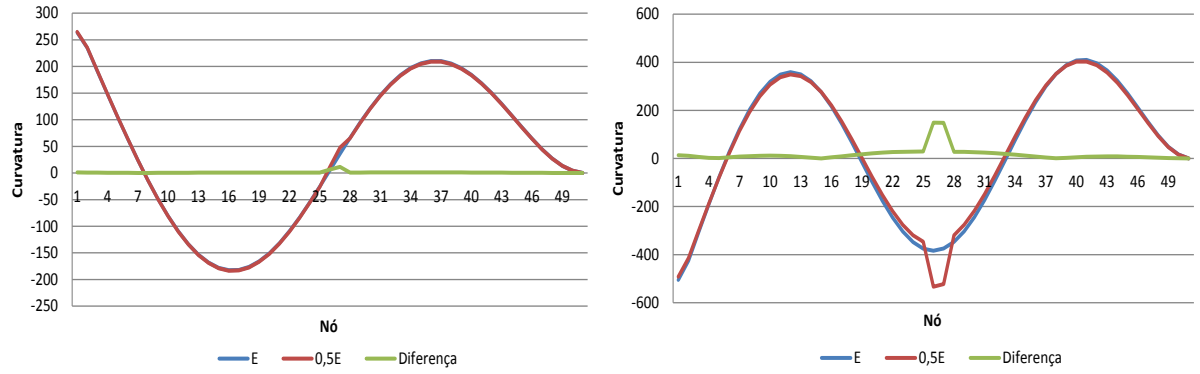
A.2 Modelo da Consola

Os seguintes 5 gráficos constituem os resultados da comparação das curvaturas dos primeiros 5 modos de vibração do mesmo modelo de uma consola intacta e uma consola com um dano a meio vão (elemento 26), simulado através de uma diminuição da rigidez de 50% num elemento local. O último gráfico desta série apresenta o somatório da diferença entre os dois modelos.



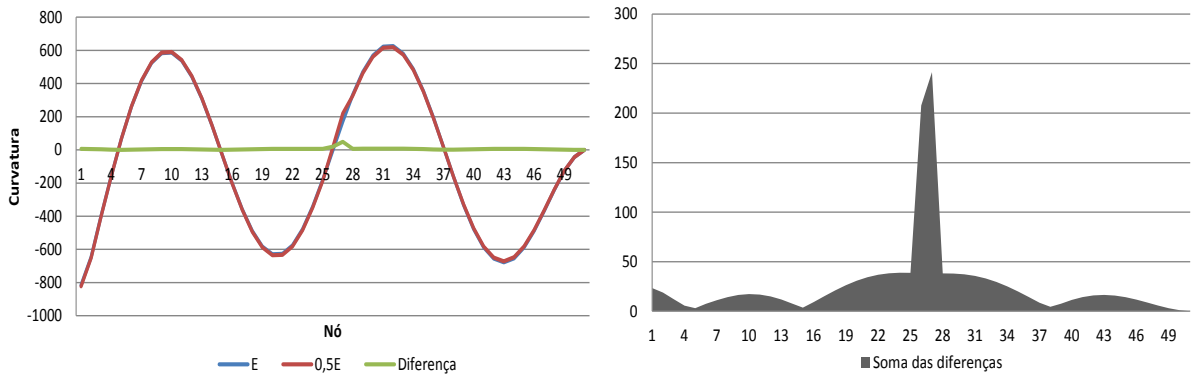
(a) Curvatura do 1º modo. (b) Curvatura do 2º modo.

Figura A.7: Consola de 50 elementos danificada a meio vão.



(a) Curvatura do 3º modo. (b) Curvatura do 4º modo.

Figura A.8: Consola de 50 elementos danificada a meio vão.



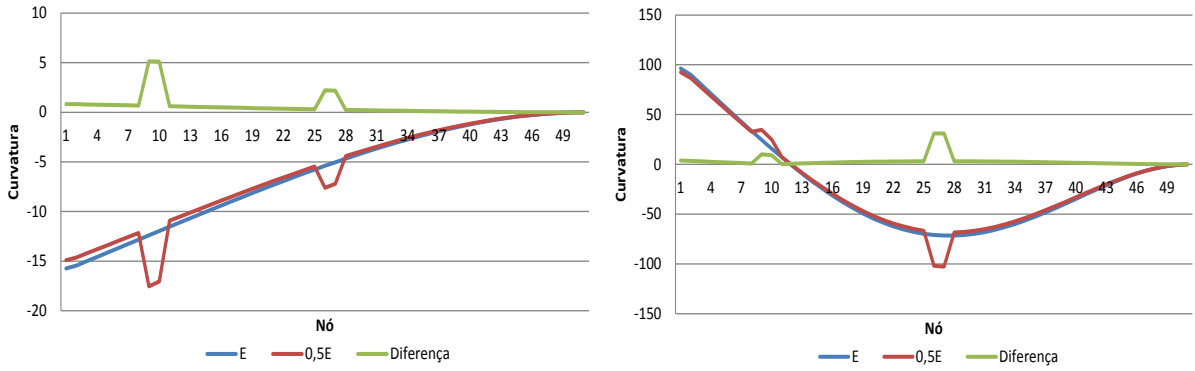
(a) Curvatura do 5º modo.

(b) Somatório das diferenças.

Figura A.9: Consola de 50 elementos danificada a meio vão

.

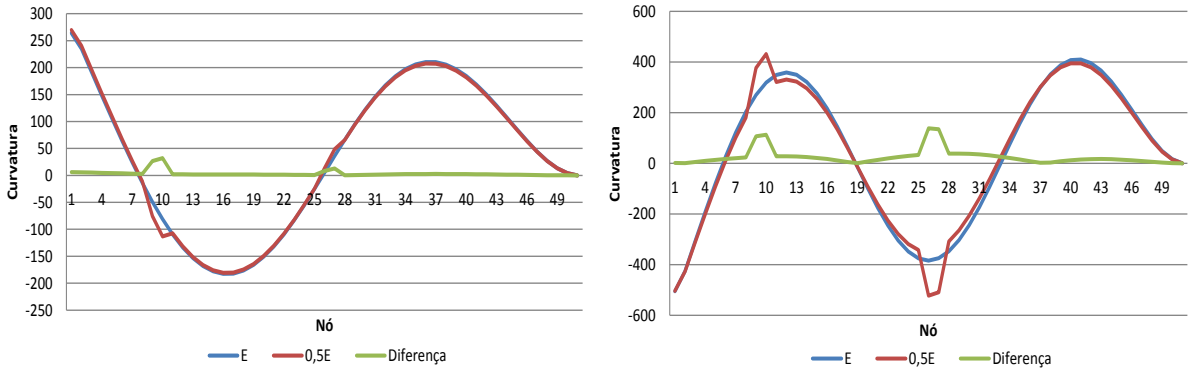
Os seguintes 5 gráficos constituem os resultados da comparação das curvaturas dos primeiros 5 modos de vibração do mesmo modelo de uma consola intacta e uma consola com com 2 danos (um no elemento 9 e outro no 26), simulado através de uma diminuição da rigidez de 50% nos elementos referidos. O último gráfico desta série apresenta o somatório da diferença entre os dois modelos.



(a) Curvatura do 1º modo.

(b) Curvatura do 2º modo.

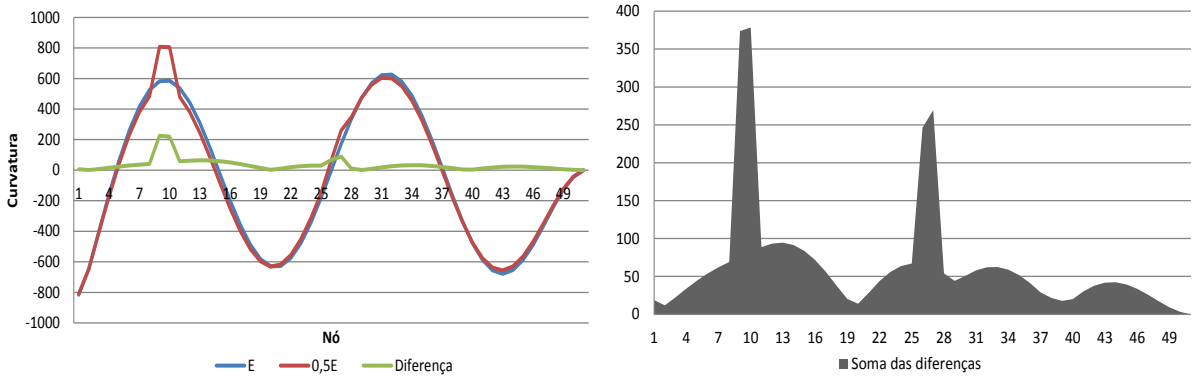
Figura A.10: Consola de 50 elementos danificada nos elementos 9 e 26.



(a) Curvatura do 3º modo.

(b) Curvatura do 4º modo.

Figura A.11: Consola de 50 elementos danificada nos elementos 9 e 26.



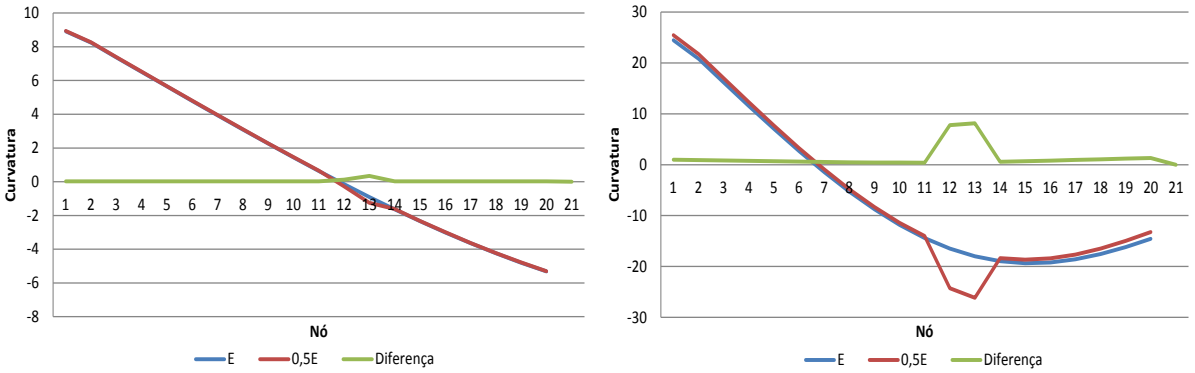
(a) Curvatura do 5º modo.

(b) Somatório das diferenças.

Figura A.12: Consola de 50 elementos danificada nos elementos 9 e 26.

A.3 Modelo do Pórtico

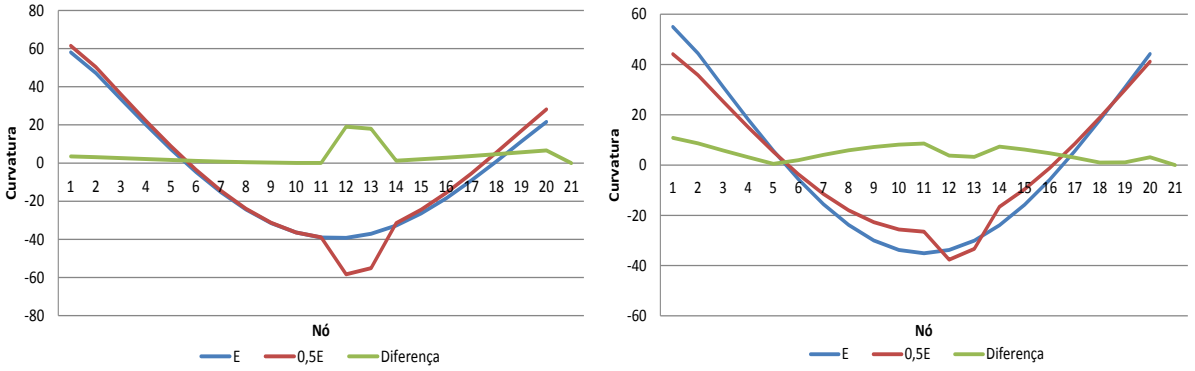
Os seguintes 5 gráficos constituem os resultados da comparação das curvaturas dos primeiros 5 modos de vibração do mesmo modelo de um pórtico intacto e um pórtico com um dano no 12º elemento do primeiro pilar, simulado através de uma diminuição da rigidez de 50% num elemento local. São apresentados os gráficos das curvaturas apenas do 1º pilar uma vez que as diferenças na zona de viga e no segundo pilar não são expressivas. O último gráfico desta série apresenta o somatório da diferença entre os dois modelos.



(a) Curvatura do 1º modo.

(b) Curvatura do 2º modo.

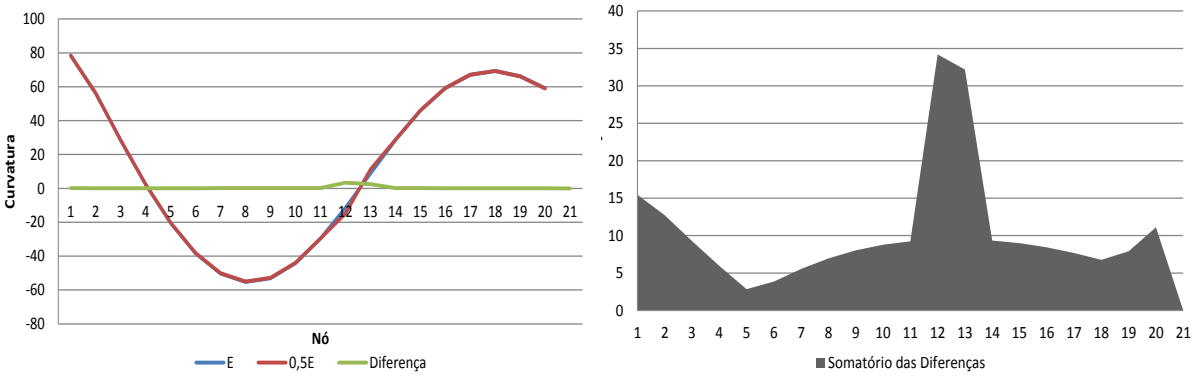
Figura A.13: Pórtico de 60 elementos danificado no 1º pilar.



(a) Curvatura do 3º modo.

(b) Curvatura do 4º modo.

Figura A.14: Pórtico de 60 elementos danificado no 1º pilar.



(a) Curvatura do 5º modo.

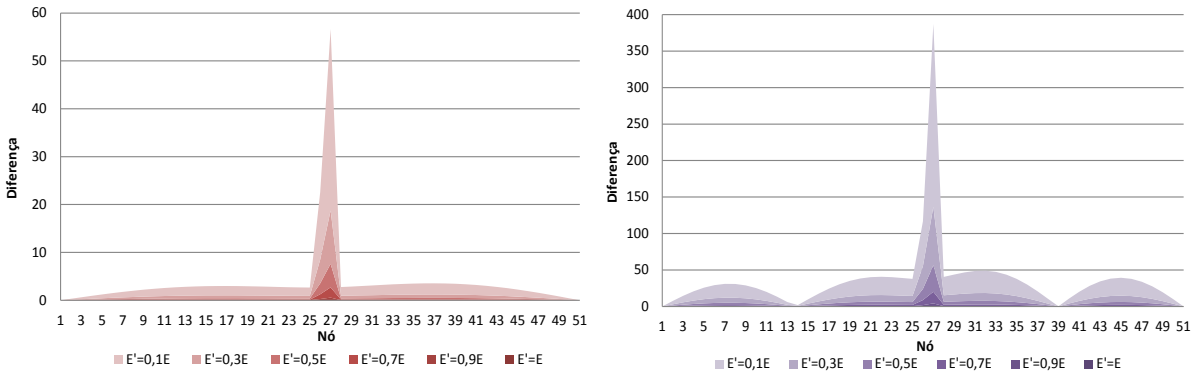
(b) Somatório das diferenças.

Figura A.15: Pórtico de 60 elementos danificado no 1º pilar.

.

A.4 Comparação Entre Diferentes Graus de Dano

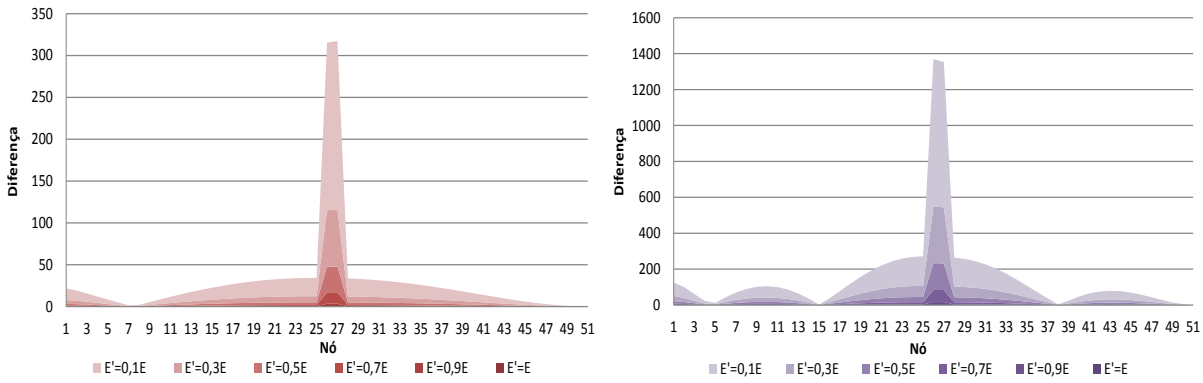
Nesta secção encontram-se os gráficos das diferenças absolutas entre as curvaturas do 2º e 4º modo de vibração, que são os modos que não constam nos exemplos dados no capitulo 3. São também apresentados os resultados relativos ao somatório das diferenças das curvaturas entre para cada um dos 2 primeiros modos de vibração da viga e consola, com 2 danos em zonas distintas.



(a) Somatório das diferenças do 2º modo.

(b) Somatório das diferenças do 4º modo.

Figura A.16: Somatório das diferenças absolutas na viga de 50 elementos.

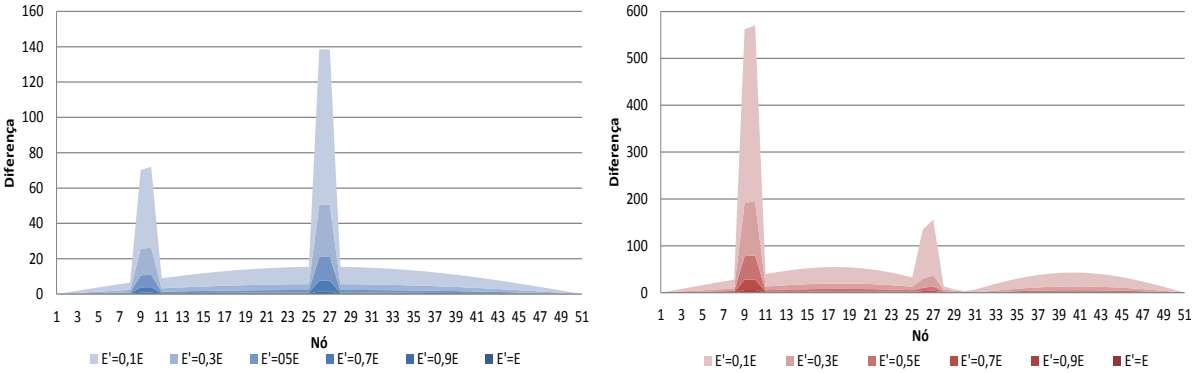


(a) Somatório das diferenças do 2º modo.

(b) Somatório das diferenças do 4º modo.

Figura A.17: Somatório das diferenças absolutas na consola de 50 elementos no modelo com 1 dano.

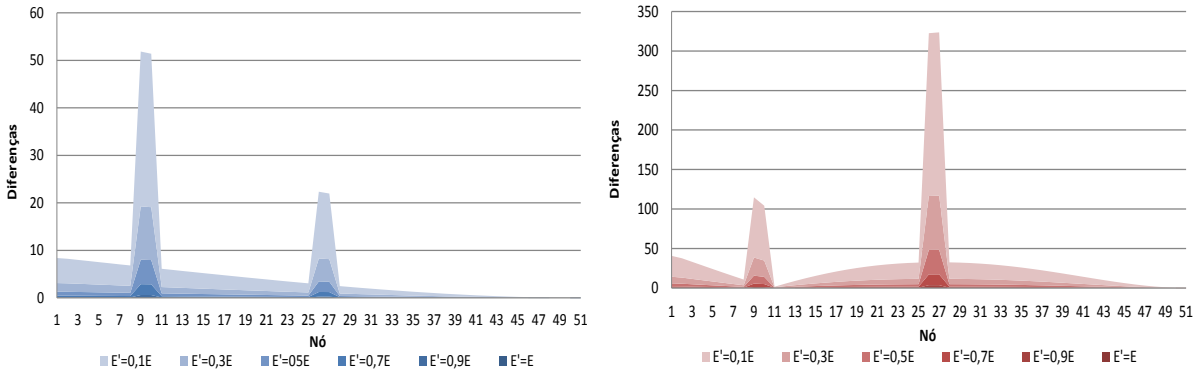
Nos gráficos seguintes encontram-se as diferenças absolutas entre as curvaturas no modelo da viga e consola com 2 danos de vários graus. Optou-se apenas por mostrar os 2 primeiros modos por ilustrarem suficientemente bem a situação.



(a) Curvatura do 1º modo.

(b) Curvatura do 2º modo.

Figura A.18: Somatório das diferenças absolutas no modelo da viga com 50 elementos com 2 danos.



(a) Curvatura do 1º modo.

(b) Curvatura do 2º modo.

Figura A.19: Somatório das diferenças absolutas no modelo da consola com 50 elementos com 2 danos.

Anexo B

Tabela das Forças de Fixação

No presente anexo é apresentada uma tabela onde se encontram as forças de fixação que se desenvolvem nas extremidades das barras com deslocamentos impostos. Os casos da barra bi-encastrada e encastrada-rotulada permitiram construir as matrizes de rigidez locais dos elementos da viga bi-apoiada e da viga em consola.

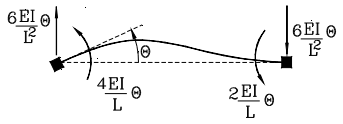
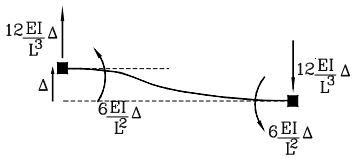
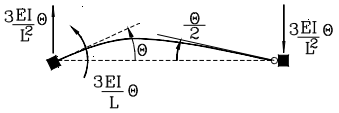
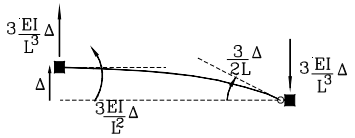
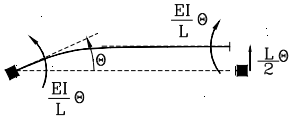
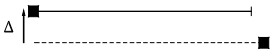
Tipo de barra	Imposição de rotação à esquerda	Imposição de deslocamento transversal
bi-encastrada		
encastrada-rotulada		
encastrada-enc desliz.		

Figura B.1: Tabela das Forças de Fixação.

Anexo C

Código Fonte do Programa de Cálculo

No anexo que segue encontra-se definido o código fonte desenvolvido no âmbito da criação do programa de cálculo em *Java* que permite fazer uso das características do método das curvaturas. Este anexo está dividido em 4 capítulos, correspondentes às 4 classes principais: *Método_das_Curvaturas*, *Selecao_de_Dano*, *Interface* e *Gráficos_Curvaturas*.

O utilizador pode aceder à aplicação em *https://dl.dropboxusercontent.com/u/6153977/dist/launch.html*

C.1 Método das Curvaturas

Com vista a facilitar a consulta deste capitulo do Anexo B, é apresentada uma lista onde se encontram as funções e respectiva ordem pela qual surgem no documento. As funções são então as seguintes:

- Build_GDL;
- Build_Klocal_in;
- Build_Global_K_Consola;
- Build_Cond_K_Consola;
- Build_Massa_Consola;
- Curvatura_Consola;
- Consola;
- Build_GDL_Viga;
- Build_Klocal_Vigaex1;
- Build_Klocal_Vigaex2;
- Build_Global_K_Viga;

- Build_Cond_K_Viga;
- Build_Massa_Viga;
- Viga;
- Funções auxiliares;
- Plot.

Em seguida encontra-se o código *Java* correspondente a cada uma das funções anteriormente mencionadas. Refira-se que, embora estas funções estejam separadas das outras classes referidas no inicio do anexo, funcionam todas em conjunto.

- Pacotes importados para o NetBeans IDE

```
package dissertacao;

import javax.swing.JFrame;
import java.util.Scanner;
import Jama.*;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Polygon;
import javax.swing.JPanel;
import java.applet.*;
import java.awt.*;
```

- Funções que permitem definir os modelos nominal e danificado da viga e consola com 10, 20 e 50 elementos e as frequências de vibração de cada um deles

```
public class Metodo_Das_Curvaturas extends Applet{

    public static float [] Viga10 = new float [10];
    public static float [] Viga20 = new float [20];
    public static float [] Viga50 = new float [50];
    public static float [] Viga10_N = new float [10];
    public static float [] Viga20_N = new float [20];
    public static float [] Viga50_N = new float [50];

    public static float [] Consola10 = new float [10];
    public static float [] Consola20 = new float [20];
    public static float [] Consola50 = new float [50];
    public static float [] Consola10_N = new float [10];
    public static float [] Consola20_N = new float [20];
```

```
public static float [] Consola50_N = new float [50];

public static double fn_n;
public static double fn_d;
public static double fn;

private static JFrame NewJFrame;
```

- Função que permite calcular a matriz GDL para um qualquer número de elementos. Esta matriz define a ordem com que os valores das matrizes de rigidez locais dos elementos vão entrar na matriz de rigidez global da consola

```
public static int[][] Build_GDL(int elementos){

        int [][] GDL = new int[elementos] [4];
        for (int n = 0; n < elementos; n++){
            GDL[n][0]=n;
            GDL[n][1]=n+1;
            GDL[n][2]=n+elementos;
            GDL[n][3]=n+elementos+1;
        }
    return GDL;
    };
```

- Matriz local de um elemento tipo interior com 4 graus de liberdade, tanto para a consola como para a viga

```
public static float [][] Build_KLocal_in(float EI, float L){

        float[][] KLocal = new float [4][4];

        KLocal[0][0] = 12*EI/(L*L*L);
        KLocal[0][1] = -12*EI/(L*L*L);
        KLocal[1][0] = -12*EI/(L*L*L);
        KLocal[0][2] = 6*EI/(L*L);
        KLocal[2][0] = 6*EI/(L*L);
        KLocal[0][3] = 6*EI/(L*L);
        KLocal[3][0] = 6*EI/(L*L);
        KLocal[1][1] = 12*EI/(L*L*L);
        KLocal[1][2] = -6*EI/(L*L);
        KLocal[2][1] = -6*EI/(L*L);
        KLocal[1][3] = -6*EI/(L*L);
        KLocal[3][1] = -6*EI/(L*L);
        KLocal[2][2] = 4*EI/L;
        KLocal[2][3] = 2*EI/L;
        KLocal[3][2] = 2*EI/L;
```

```
        KLocal[3][3] = 4*EI/L;

            return KLocal;
    }
```

- Função que calcula a matriz de rigidez global da consola

```
public static float [][] Build_Global_K_Consola(
int elementos, float [] EI){

float L = (float) (1.0/elementos);
float [][] global_k = new float [2*elementos][2*elementos];
int [][] GDL = new int [elementos][4];
float [][] klocal = new float [4][4];

GDL= Build_GDL(elementos);

for(int ie=0; ie<elementos; ie++){
  klocal = Build_KLocal_in((float)EI[ie],L);
  if (ie==0){
        for(int ii=1; ii<4; ii=ii+2){
                for(int jj=1; jj<4; jj=jj+2){
                    global_k[(GDL[ie][ii])-1][(GDL[ie][jj])-1]=
                    (float)global_k[(GDL[ie][ii])-1]
                    [(GDL[ie][jj])-1] + klocal[ii][jj];
                }
        }
}
else{
        for(int ii=0; ii<4; ii++){
                for(int jj=0; jj<4; jj++){
                    global_k[(GDL[ie][ii])-1][(GDL[ie][jj])-1]=
                    (float)global_k[(GDL[ie][ii])-1]
                    [(GDL[ie][jj])-1] + klocal[ii][jj];
                }
        }
    }
}
return global_k;
};
```

- Função que permite condensar a matriz de rigidez global da consola para uma matriz com dimensão correspondente ao número de graus de liberdade verticais

```
public static Matrix Build_Cond_K_Consola(int elementos,
float [][] global_k){

        float [][] k11 = new float [elementos][elementos];
        float [][] k12 = new float [elementos][elementos];
        float [][] k21 = new float [elementos][elementos];
        float [][] k22 = new float [elementos][elementos];

        for (int ii=0; ii<elementos; ii++)
            for(int jj=0; jj<elementos ;jj++){
                k11[ii][jj]=global_k[ii][jj];
            };

      for (int ii=elementos; ii<2*elementos; ii++)
            for(int jj=0; jj<elementos ;jj++){
                k21[ii-elementos][jj]=global_k[ii][jj];
            };

      for (int ii=0; ii<elementos; ii++)
            for(int jj=elementos; jj<2*elementos ;jj++){
                k12[ii][jj-elementos]=global_k[ii][jj];
            };

      for (int ii=elementos; ii<2*elementos; ii++)
            for(int jj=elementos; jj<2*elementos ;jj++){
                k22[ii-elementos][jj-elementos]=global_k[ii][jj];
            };

      Matrix K11 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k11));
      Matrix K21 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k21));
      Matrix K12 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k12));
      Matrix K22 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k22));

      Matrix K=K11.minus(K21.transpose()
      .times(K22.inverse()).times(K21));

      return K;
    };
```

- Função que permite calcular a matriz de massa diagonal da consola

```
public static float [][] Build_Massa_Consola(int elementos){

      float [][] M = new float [elementos][elementos];

      for(int ii=0; ii<elementos; ii++)
          for (int jj=0; jj<elementos; jj++)
              if (ii==jj)
                  M[ii][jj]=(float) 1/elementos;
              else
                  M[ii][jj]=0;

     M[elementos-1][elementos-1]=M[elementos-1][elementos-1]/2;
      return M;
};
```

- Função que permite usar o operador de diferenças finitas para obter a matriz das curvaturas da consola através da matriz de vectores próprios

```
public static float [][] Curvatura_Consola (Matrix eigenVectors_,
 int elementos){

        float [][] C = new float [elementos+1][elementos];
        float [][] eigenVectors = new float [elementos+1][elementos];

        for (int ii=0; ii<elementos+1 ; ii++)
          for (int jj=0; jj<elementos; jj++)
            if (ii==0)
              eigenVectors[ii][jj]=0;
            else
              eigenVectors[ii][jj]=(float)eigenVectors_.get(ii-1,jj);

      for(int jj=0; jj<elementos; jj++)
          for(int ii=0; ii<elementos+1; ii++)
          if(ii==0)
              C[ii][jj]=(float)(2*eigenVectors[ii+1][jj]
              -2*eigenVectors[ii][jj])*(elementos*elementos);
          else
              if(ii<elementos)
                  C[ii][jj]=(float)(eigenVectors[ii-1][jj]
                  +eigenVectors[ii+1][jj]-2*eigenVectors[ii][jj])
                  *(elementos*elementos);
              else
                  C[ii][jj]=0;
      return C;
```

```
};
```

- Função que executa as funções já criadas e calcula a matriz dinâmica, a matriz de valores próprios, matriz de vectores próprios e retorna a matriz das curvaturas no modelo da consola

```
public static float [][] Consola (float [] EI, int elementos){

        1. Matriz de Rigidez
        float [][] global_k = Build_Global_K_Consola(elementos,EI);

        2. Matriz de Ridigez Condensada
        Matrix K = Build_Cond_K_Consola(elementos,global_k);

        3. Matriz de Massa
        Matrix M = Jama.Matrix.constructWithCopy(
        ConvertFF2DD(Build_Massa_Consola(elementos)));

        4. MAtriz Dinâmica
        Matrix D = K.inverse().times(M);

        5. Valores Próprios
        Matrix Valores_proprios = D.eig().getD();

        6. Vectores Próprios
        Matrix Vectores_proprios = D.eig().getV();

        7. Curvatura
        float [][] C = Curvatura(Vectores_proprios,elementos);

        fn = 1/sqrt(Valores_proprios.get(modo,modo));

        return C;
    };
```

- Função que permite calcular a matriz GDL para um número qualquer de elementos. Esta matriz define a ordem com que os valores das matrizes de rigidez locais dos elementos vão entrar na matriz de rigidez global da viga

```
public static int[][] Build_GDL_Viga(int elementos){

        int [][] GDL = new int[elementos] [4];

        for (int n = 0; n < elementos; n++){
            GDL[n][0]=n;
            GDL[n][1]=n+1;
            GDL[n][2]=n+elementos;
```

```
                GDL[n][3]=n+elementos+1;
            }

            return GDL;
        };
```

- Matriz local do elemento extremo do lado esquerdo da viga. Apesar da dimensão desta matriz, este elemento tem apenas 2 graus de liberdade, um de translação e um de rotação

```
public static float [][] Build_KLocal_Vigaex1(float EI, float L){

                float[][] KLocal = new float [4][4];

                KLocal[0][0] = 0;
                KLocal[0][1] = 0;
                KLocal[1][0] = 0;
                KLocal[0][2] = 0;
                KLocal[2][0] = 0;
                KLocal[0][3] = 0;
                KLocal[3][0] = 0;
                KLocal[1][1] = (3*EI/(L*L*L));
                KLocal[1][2] = 0;
                KLocal[2][1] = 0;
                KLocal[1][3] = (-3*EI/(L*L));
                KLocal[3][1] = (-3*EI/(L*L));
                KLocal[2][2] = 0;
                KLocal[2][3] = 0;
                KLocal[3][2] = 0;
                KLocal[3][3] = (3*EI/(L));

                return KLocal;
    };
```

- Matriz local do elemento extremo do lado direito da viga. Apesar da dimensão desta matriz, este elemento tem apenas 2 graus de liberdade, um de translação e um de rotação

```
public static float [][] Build_KLocal_Vigaex2(float EI, float L){

                float[][] KLocal = new float [4][4];

                KLocal[0][0] = 3*EI/(L*L*L);
                KLocal[0][1] = 0;
                KLocal[1][0] = 0;
                KLocal[0][2] = 3*EI/(L*L);
                KLocal[2][0] = 3*EI/(L*L);
                KLocal[0][3] = 0;
                KLocal[3][0] = 0;
```

```
                KLocal[1][1] = 0;
                KLocal[1][2] = 0;
                KLocal[2][1] = 0;
                KLocal[1][3] = 0;
                KLocal[3][1] = 0;
                KLocal[2][2] = 3*EI/L;
                KLocal[2][3] = 0;
                KLocal[3][2] = 0;
                KLocal[3][3] = 0;

                return KLocal;
    };
```

- Função que calcula a matriz de rigidez global da viga

```
public static float [][] Build_Global_K_Viga(
int elementos, float [] EI){

float L = (float) (1.0/elementos);
float [][]global_k = new float [2*elementos][2*elementos];
float [][] global_k_ = new float[(2*elementos)-2][(2*elementos)-2];
int [][] GDL = new int [elementos][4];
int [][] GDL_V = new int [elementos][4];

float [][] klocal = new float [4][4];

GDL= Build_GDL(elementos);
GDL_V= Build_GDL_Viga(elementos);

for(int ie=0; ie<elementos; ie++){
    if ((ie==0)){
         klocal = Build_KLocal_Vigaex1((float)EI[ie],L);
         for(int ii=1; ii<4; ii=ii+2){
                 for(int jj=1; jj<4; jj=jj+2){
                    global_k[(GDL[ie][ii])-1][(GDL[ie][jj])-1]=
                    (float)global_k[(GDL[ie][ii])-1]
                    [(GDL[ie][jj])-1] + klocal[ii][jj];
                 }
         }
    }

    else
        if (ie==elementos-1){
            klocal = Build_KLocal_Vigaex2((float)EI[ie],L);
            for(int ii=0; ii<4; ii=ii+2){
                 for(int jj=0; jj<4; jj=jj+2){
                    global_k[(GDL_V[ie][ii])-1][(GDL_V[ie][jj])-1]=
```

```
                    (float)global_k[(GDL_V[ie][ii])-1]
                    [(GDL_V[ie][jj])-1] + klocal[ii][jj];
                }
            }
    }

    else{
            klocal = Build_KLocal_in((float)EI[ie],L);
            for(int ii=0; ii<4; ii++){
                for(int jj=0; jj<4; jj++){
                    global_k[(GDL[ie][ii])-1][(GDL[ie][jj])-1]=
                    (float)global_k[(GDL[ie][ii])-1]
                    [(GDL[ie][jj])-1] + klocal[ii][jj];
                }
            }
        }
    }
return global_k_;
};
```

- Função que permite condensar a matriz de rigidez global da viga para uma matriz com dimensão correspondente ao número de graus de liberdade verticais

```
public static Matrix Build_Cond_K_Viga(
int elementos, float [][] global_k){

        float [][] k11 = new float [elementos][elementos];
        float [][] k12 = new float [elementos][elementos];
        float [][] k21 = new float [elementos][elementos];
        float [][] k22 = new float [elementos][elementos];

        for (int ii=0; ii<elementos; ii++)
            for(int jj=0; jj<elementos ;jj++){
                k11[ii][jj]=global_k[ii][jj];
            };

       for (int ii=elementos; ii<2*elementos; ii++)
            for(int jj=0; jj<elementos ;jj++){
                k21[ii-elementos][jj]=global_k[ii][jj];
            };

       for (int ii=0; ii<elementos; ii++)
            for(int jj=elementos; jj<2*elementos ;jj++){
                k12[ii][jj-elementos]=global_k[ii][jj];
            };

       for (int ii=elementos; ii<2*elementos; ii++)
```

```
            for(int jj=elementos; jj<2*elementos ;jj++){
                k22[ii-elementos][jj-elementos]=global_k[ii][jj];
            };

        Matrix K11 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k11));
        Matrix K21 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k21));
        Matrix K12 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k12));
        Matrix K22 = Jama.Matrix.constructWithCopy(ConvertFF2DD(k22));
        Matrix K=K11.minus(K21.transpose()
        .times(K22.inverse()).times(K21));

        return K;
    };
```

- Função que permite calcular a matriz de massa diagonal da viga

```
public static float [][] Build_Massa_Viga(int elementos){

        float [][] M = new float [elementos][elementos];

        for(int ii=0; ii<elementos; ii++)
            for (int jj=0; jj<elementos; jj++)
                if (ii==jj)
                    M[ii][jj]=(float) 1/elementos;
                else
                    M[ii][jj]=0;

        M[elementos-1][elementos-1]=M[elementos-1][elementos-1];
        M[0][0]=M[0][0];
        return M;
};
```

- Função que permite usar o operador de diferenças finitas para obter a matriz das curvaturas da viga através da matriz de vectores próprios

```
public static float [][] Curvatura_Viga
(Matrix eigenVectors_, int elementos){

float [][] C = new float [elementos+2][elementos];

float [][] eigenVectors = new float [elementos+2][elementos];

for (int ii=0; ii<elementos+2 ; ii++)
    for (int jj=0; jj<elementos; jj++)
        if ((ii==0)|(ii==elementos+1))
            eigenVectors[ii][jj]=0;
```

```
            else
                eigenVectors[ii][jj]=(float)eigenVectors_.get(ii-1,jj);

    for(int ii=0; ii<elementos+2; ii++)
        for(int jj=0; jj<elementos; jj++)
            if((ii==0)|(ii==elementos+1))
                C[ii][jj]=0;
            else
                C[ii][jj]=(float)(((eigenVectors[ii+1][jj]
                -(2*eigenVectors[ii][jj])+eigenVectors[ii-1][jj]))*100);

    return C;
    };
```

- Função que executa as funções já criadas e calcula a matriz dinâmica, a matriz de valores próprios, matriz de vectores próprios e retorna a matriz das curvaturas no modelo da viga

```
    public static float [][] Viga (float [] EI, int elementos){

            1. Matriz de Rigidez
            float [][] global_k = Build_Global_K_Viga(elementos,EI);

            2. Matriz de Ridigez Condensada
            Matrix K = Build_Cond_K_Consola(elementos-1,global_k);

            3. Matriz de Massa
            Matrix M = Jama.Matrix.constructWithCopy(
            ConvertFF2DD(Build_Massa_Viga(elementos-1)));

            4. Matriz Dinâmica
            Matrix D = K.inverse().times(M);

            5. Valores Próprios
            Matrix Valores_proprios = D.eig().getD();

            6. Vectores Próprios
            Matrix Vectores_proprios = D.eig().getV();

            7. Curvatura
            float [][] V = Curvatura_Viga_2(Vectores_proprios,elementos-1);

            fn = 1/sqrt(Valores_proprios.get(modo,modo));

            return V;
        }
```

- Conjunto de funções auxiliares. A função *initVariables* a rigidez com valor unitário para todos os casos dos modelos da viga e da consola, tanto nominal como danificado. A função *ConvertFF2DD* permite converter *floats* em *doubles* para poder correr o pacote *Jama* que permite realizar operações entre matrizes. A função *RemoveZeros* permite remover duas linhas e duas colunas de zeros na matriz de rigidez global da viga, que são dispensáveis para os cálculos.

```
public static void initVariables(){
            for (int n=0; n<Viga10.length; n++){
            Viga10[n]=(float)1.0;
            Viga10_N[n]=(float)1.0;
            }
            for (int n=0; n<Viga20.length; n++){
            Viga20[n]=(float)1.0;
            Viga20_N[n]=(float)1.0;
            }
            for (int n=0; n<Viga50.length; n++){
            Viga50[n]=(float)1.0;
            Viga50_N[n]=(float)1.0;
            }

            for (int n=0; n<Consola10.length; n++){
            Consola10[n]=(float)1.0;
            Consola10_N[n]=(float)1.0;
            }
            for (int n=0; n<Consola20.length; n++){
            Consola20[n]=(float)1.0;
            Consola20_N[n]=(float)1.0;
            }
            for (int n=0; n<Consola50.length; n++){
            Consola50[n]=(float)1.0;
            Consola50_N[n]=(float)1.0;
            }

        }

private static double [][] ConvertFF2DD (float[][] A){

 double [][] B = new double[A.length][A.length];

 for (int ii=0; ii<A.length; ii++)
     for (int jj=0; jj<A.length; jj++)
        B[ii][jj]=Double.parseDouble(Float.toString(A[ii][jj]));

 return B;
};
```

```
private static float [][] RemoveZeros
(float [][] KGlobal, int elementos){

float [][] KGlobal_ = new float[(2*elementos)-2][(2*elementos)-2];

Remover zeros interiores
for(int ii=0;ii<elementos; ii++)
   for(int jj=0; jj<elementos; jj++)
        KGlobal_[ii][jj]=KGlobal[ii][jj];
Remover ultima coluna de zeros
for(int ii=0;ii<elementos; ii++)
   for(int jj=elementos; jj<2*elementos-1; jj++)
        KGlobal_[ii][jj-1]=KGlobal[ii][jj];

Remover zeros interiores
for(int ii=elementos;ii<2*elementos-1; ii++)
   for(int jj=0; jj<elementos; jj++)
        KGlobal_[ii-1][jj]=KGlobal[ii][jj];
Remover ultima coluna de zeros
for(int ii=elementos;ii<2*elementos-1; ii++)
   for(int jj=elementos; jj<2*elementos-1; jj++)
        KGlobal_[ii-1][jj-1]=KGlobal[ii][jj];

return KGlobal_;
}
```

- A função *plot* é a função principal desta classe e está encarregue tanto de chamar e executar as funções que fazem o cálculo (função *Viga* e função *Consola*), como de definir detalhes gráficos e técnicos da interface

```
public static void plot (int estrutura, int modo){
        float [][] Nominal;
        float [][] Danificada;

        if (estrutura ==1){
            System.out.println("Nominal");
            Nominal = Viga(Viga10_N,Viga10.length);
            fn_n=fn;
            System.out.println("Danificada");
            Danificada = Viga(Viga10,Viga10.length);
            fn_d=fn;
            Graficos_Curvaturas frame = new Graficos_Curvaturas
            (Nominal,Danificada,modo,fn_n,fn_d);
            frame.setSize(580,380);
            int modo_=modo+1;
            if (modo < 5)
                  frame.setTitle("Viga (10 Elementos) -
```

```
            " + modo_ + "º Modo de Vibração");
        else
          frame.setTitle("Viga (10 Elementos) -
          Somatório das diferenças das curvaturas");
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.setLocationRelativeTo(null);
      frame.setVisible(true);
}

if (estrutura ==2){
      System.out.println("Nominal");
      Nominal = Viga(Viga20_N,Viga20.length);
      fn_n=fn;
      System.out.println("Danificada");
      Danificada = Viga(Viga20,Viga20.length);
      fn_d=fn;
      Graficos_Curvaturas frame = new Graficos_Curvaturas
      (Nominal,Danificada,modo,fn_n,fn_d);
      frame.setSize(580,380);
      int modo_=modo+1;
      if (modo < 5)
            frame.setTitle("Viga (20 Elementos) -
             " + modo_ + "º Modo de Vibração");
        else
          frame.setTitle("Viga (20 Elementos) -
          Somatório das diferenças das curvaturas");
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.setLocationRelativeTo(null);
      frame.setVisible(true);
}

if (estrutura ==3){
      System.out.println("Nominal");
      Nominal = Viga(Viga50_N,Viga50.length);
      fn_n=fn;
      System.out.println("Danificada");
      Danificada = Viga(Viga50,Viga50.length);
      fn_d=fn;
      Graficos_Curvaturas frame = new Graficos_Curvaturas
      (Nominal,Danificada,modo,fn_n,fn_d);
      frame.setSize(580,380);
      int modo_=modo+1;
      if (modo < 5)
            frame.setTitle("Viga (50 Elementos) -
             " + modo_ + "º Modo de Vibração");
        else
          frame.setTitle("Viga (50 Elementos) -
```

```
                Somatório das diferenças das curvaturas");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }

    if (estrutura ==4){
        System.out.println("Nominal");
        Nominal = Consola(Consola10_N,Consola10.length);
        fn_n=fn;
        System.out.println("Danificada");
        Danificada = Consola(Consola10,Consola10.length);
        fn_d=fn;
        Graficos_Curvaturas frame = new Graficos_Curvaturas
        (Nominal,Danificada,modo,fn_n,fn_d);
        frame.setSize(580,380);
        int modo_=modo+1;
        if (modo < 5)
            frame.setTitle("Consola (10 Elementos) -
             " + modo_ + "º Modo de Vibração");
          else
            frame.setTitle("Consola (10 Elementos) -
             Somatório das diferenças das curvaturas");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
    if (estrutura ==5){
        System.out.println("Nominal");
        Nominal = Consola(Consola20_N,Consola20.length);
        fn_n=fn;
        System.out.println("Danificada");
        Danificada = Consola(Consola20,Consola20.length);
        fn_d=fn;
        Graficos_Curvaturas frame = new Graficos_Curvaturas
        (Nominal,Danificada,modo,fn_n,fn_d);
        frame.setSize(580,380);
        int modo_=modo+1;
        if (modo < 5)
            frame.setTitle("Consola (20 Elementos) -
             " + modo_ + "º Modo de Vibração");
         else
            frame.setTitle("Consola (20 Elementos) -
             Somatório das diferenças das curvaturas");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
```

```
            }
            if (estrutura ==6){
                System.out.println("Nominal");
                Nominal = Consola(Consola50_N,Consola50.length);
                fn_n=fn;
                System.out.println("Danificada");
                Danificada = Consola(Consola50,Consola50.length);
                fn_d=fn;
                Graficos_Curvaturas frame = new Graficos_Curvaturas
                (Nominal,Danificada,modo,fn_n,fn_d);
                frame.setSize(580,380);
                int modo_=modo+1;
                if (modo < 5)
                        frame.setTitle("Consola (50 Elementos) -
                         " + modo_ + "º Modo de Vibração");
                      else
                        frame.setTitle("Consola (50 Elementos) -
                         Somatório das diferenças das curvaturas");
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            }

        }
```

C.2 Seleção de Dano

Esta classe é a que define o funcionamento da janela de seleção de dano que surge quando selecionado um elemento do modelo da viga ou consola.

```
package dissertacao;

 * @author Mauro
 */
public class Selecao_de_Dano extends javax.swing.JFrame {

    public Selecao_de_Dano() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
```

- Código que está associado ao botão *Ok*, que permite selecionar o dano e induzir as alterações na rigidez do modelo

```
private void jButton1ActionPerformed(
java.awt.event.ActionEvent evt) {

        if(jComboBox1.getSelectedItem().equals("0%")){
            Interface.setItemOnDissertacao(1);
        }
        else if(jComboBox1.getSelectedItem().equals("10%")){
            Interface.setItemOnDissertacao((float)0.9);
        }
        else if(jComboBox1.getSelectedItem().equals("30%")){
            Interface.setItemOnDissertacao((float)0.7);
        }
        else if(jComboBox1.getSelectedItem().equals("50%")){
            Interface.setItemOnDissertacao((float)0.5);
        }
        else if(jComboBox1.getSelectedItem().equals("70%")){
            Interface.setItemOnDissertacao((float)0.3);
        }
        else if(jComboBox1.getSelectedItem().equals("90%")){
            Interface.setItemOnDissertacao((float)0.1);
        }
        this.dispose();
}
```

- Função *main* de execução da janela de seleção de dano

```
        public static void main(String args[]) {

            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new Selecao\_de\_Dano().setVisible(true);
                }
            });
        }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
```

C.3 Gráficos Curvaturas

Nesta classe são usados os pacotes e funções necessários para executar os *plots* das curvaturas dos modelos da viga e consola, intacto e danificado.

- Pacotes importados necessários para executar os gráficos

```
package dissertacao;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Polygon;
import static java.lang.Math.abs;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

- Função que expande a classe *JFrame* onde estão definidas como variáveis de entrada as curvaturas, o modo de vibração e as frequências correspondentes aos modelos nominal e danificado para poderem ser definidos os gráficos

```
public class Graficos_Curvaturas extends JFrame{

    public Graficos_Curvaturas(float[][] Nominal,
    float[][] Danificado, int MV, double fn_n, double fn_d) {

    setLayout(new BorderLayout());
    add(new DrawSine(Nominal,Danificado,Nominal.length,MV,fn_n,fn_d),
    BorderLayout.CENTER);
}
}
```

- Função que expande a classe *JPanel* e permite abrir a janela do gráfico e executar os *plots*. É nesta função que são executadas todas as exigências referidas no Capitulo 5, que permitem chamar variáveis, ajustar escalas e valores

```
    class DrawSine extends JPanel {

    private float[][] n; //Vector Nominal
    private float[][] d; //Vector Danificado
    private int i; // Número de Elementos
    private int MV;//Modo de Vibração
    private double fn_n; //Frequência Natural Nominal
    private double fn_d; //Frequência Natural Danificada
```

```
    public static float Escala_Y(
    float [][] N, float [][] D, int modo, int elementos){

    float max = 0;
    for (int i=0; i<elementos; i++)
        if (abs(N[i][modo])>max)
                max=abs(N[i][modo]);

    for (int i=0; i<elementos; i++)
        if (abs(D[i][modo])>max)
                max=abs(D[i][modo]);

    return 105/max;
    }

    public static float Escala_D(float [] A, int elementos){

        float max = 0;
        for (int i=0; i<elementos; i++)
            if (abs(A[i])>max)
                    max=abs(A[i]);
        return 105/max;
        }

    public static float [] Somatorio(
    float [][] N,float [][] D,int elementos){

        float[] S = new float[elementos];

        for(int ii=0;ii<elementos;ii++)
            S[ii]=abs(abs(N[ii][0])-abs(D[ii][0]))+
                    abs(abs(N[ii][1])-abs(D[ii][1]))+
                    abs(abs(N[ii][2])-abs(D[ii][2]))+
                    abs(abs(N[ii][3])-abs(D[ii][3]))+
                    abs(abs(N[ii][4])-abs(D[ii][4]));
            return S;
        };

    public static double round(double value, int places) {
    if (places < 0) throw new IllegalArgumentException();

    long factor = (long) Math.pow(10, places);
    value = value * factor;
    long tmp = Math.round(value);
    return (double) tmp / factor;
}
```

```
public DrawSine(float[][] n,
float[][] d,int i, int MV, double fn_n, double fn_d){

    this.n = n;
    this.d = d;
    this.i = i;
    this.MV = MV;
    this.fn_n = fn_n;
    this.fn_d = fn_d;
}

@SuppressWarnings("empty-statement")
@Override
protected void paintComponent(Graphics g) {

    super.paintComponent(g);
    Polygon pn = new Polygon();
    Polygon pd = new Polygon();
    Polygon df = new Polygon();

    float escala_y = Escala_Y(n,d,MV,i);

    Eixos
    g.setColor(Color.black);

    //Eixo X
    g.drawLine(25, 150, 500, 150);

    //Eixo Y
    g.drawLine(25, 45, 25, 255);

    float [] s=new float[i];
            float escala_d;

    if (MV<5){

      for (int x=0;x<i; x++){
        pn.addPoint(25 +(int)(x*450/(i-1)),
        150-(((int)(1000*(n[x][MV]*escala_y)))/1000));
    }

      g.setColor(Color.blue);
      g.drawPolyline(pn.xpoints, pn.ypoints, pn.npoints);

      for (int x=0;x<i; x++){
        pd.addPoint(25 +(int)(x*450/(i-1)),
```

```
      150-(((int)(1000*(d[x][MV]*escala_y)))/1000));
    };

    g.setColor(Color.red);
    g.drawPolyline(pd.xpoints, pd.ypoints, pd.npoints);


    for (int x=0;x<i; x++){
      df.addPoint(25 +(int)(x*450/(i-1)),
      150-abs(abs(((int)(1000*(n[x][MV]*escala_y)))/1000)
      -abs(((int)(1000*(d[x][MV]*escala_y)))/1000)));
  };

  else{


              s=Somatorio(n,d,i);
              escala_d = Escala_D(s,i);
              for (int x=0;x<i; x++){
                  df.addPoint(25 +(int)(x*450/(i-1)),
                  150-(((int)(1000*(s[x]*escala_d)))/1000));
              };
  }

  float [] hsb = Color.RGBtoHSB(34,177,76,null);


  g.setColor(Color.getHSBColor(hsb[0],hsb[1],hsb[2]));
  g.drawPolyline(df.xpoints, df.ypoints, df.npoints);

  g.setColor(Color.black);

  //Frequência Natural
  g.drawString("FnN = "+round(fn_n,5)+" rad/s ;",180,330);
  g.drawString("FnD = "+round(fn_d,5)+" rad/s",290,330);

  //Legenda
  g.drawString("Diferença",355,300);
  g.drawString("Nominal",155,300);
  g.drawString("Danificado",255,300);

  g.setColor(Color.getHSBColor(hsb[0],hsb[1],hsb[2]));
  g.drawLine(330, 295, 350, 295);
  g.setColor(Color.blue);
  g.drawLine(130, 295, 150, 295);
  g.setColor(Color.red);
  g.drawLine(230, 295, 250, 295);
```

```
        g.setColor(Color.black);

        //Seta X
        g.drawLine(495, 147, 500, 150);
        g.drawLine(495, 153, 500, 150);

        //Seta Y
        g.drawLine(22,50,25,45);
        g.drawLine(28,50,25,45);

        //Legenda eixos
        g.drawString("Comp.", 514, 145);
        g.drawString("unitário", 513, 165);
        g.drawString("Curvatura", 5, 30);

        //Escala X
        g.drawLine(70,145,70,155);
        g.drawString("0.1",61,175);
        g.drawLine(115,145,115,155);
        g.drawString("0.2",106,175);
        g.drawLine(160,145,160,155);
        g.drawString("0.3",151,175);
        g.drawLine(205,145,205,155);
        g.drawString("0.4",196,175);
        g.drawLine(250,145,250,155);
        g.drawString("0.5",241,175);
        g.drawLine(295,145,295,155);
        g.drawString("0.6",286,175);
        g.drawLine(340,145,340,155);
        g.drawString("0.7",331,175);
        g.drawLine(385,145,385,155);
        g.drawString("0.8",376,175);
        g.drawLine(430,145,430,155);
        g.drawString("0.9",421,175);
        g.drawLine(475,145,475,155);
        g.drawString("1",471,175);
    }
}
```

C.4 Interface

Nesta classe encontram-se as funções associadas à definição da interface do programa.

- Pacotes importados necessários para correr o código da interface

```
package dissertacao;

import java.awt.Color;
import java.awt.Image;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.AbstractButton;
import javax.swing.Icon;
import javax.swing.ImageIcon;
```

- Função que estabelece a ligação da classe *Interface* ao *JFrame* e define quais os componentes gráficos visíveis quando iniciada a aplicação

```
public class Interface extends javax.swing.JFrame {
    private String Image;
    public static int selectedItem;
    public static int selectedStructure;
    public static int selectedVMode;
    public static int selectedBox;
    public static Metodo_Das_Curvaturas diss;

    public Interface() {
        initComponents();
            jPanel1.setVisible(false);
            jPanel2.setVisible(false);
            jPanel3.setVisible(false);
            jPanel4.setVisible(false);
            jPanel5.setVisible(false);
            jPanel6.setVisible(false);
            jPanel7.setVisible(false);
            jPanel8.setVisible(true);
            jPanel10.setVisible(false);
            this.diss = new Metodo_Das_Curvaturas();
    }
```

- Função associada à *combobox* que permite escolher os modelos da viga ou consola. Esta função permite tornar visíveis ou invisíveis os painéis onde foram desenhados cada um dos modelos individualmente. Tome-se como exemplo a primeira condição. Se o utilizador escolher o caso da viga bi-apoiada com 10 elementos apenas o *jPanel1* se torna visível, enquanto que os restantes, correspondentes aos outros modelos, permanecem invisíveis.

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {


        if(jComboBox1.getSelectedItem().
        equals("Viga bi-apoiada (10 elementos)")){

            jPanel1.setVisible(true);
            jPanel2.setVisible(false);
            jPanel3.setVisible(false);
            jPanel4.setVisible(false);
            jPanel5.setVisible(false);
            jPanel6.setVisible(false);
            jPanel7.setVisible(false);
            jPanel10.setVisible(false);
            selectedStructure=1;
        };

        if(jComboBox1.getSelectedItem().
        equals("Viga bi-apoiada (20 elementos)")){

            jPanel1.setVisible(false);
            jPanel2.setVisible(true);
            jPanel3.setVisible(false);
            jPanel4.setVisible(false);
            jPanel5.setVisible(false);
            jPanel6.setVisible(false);
            jPanel7.setVisible(false);
            jPanel10.setVisible(false);
            selectedStructure=2;
        };

        if(jComboBox1.getSelectedItem().
        equals("Viga bi-apoiada (50 elementos)")){

            jPanel1.setVisible(false);
            jPanel2.setVisible(false);
            jPanel3.setVisible(true);
            jPanel4.setVisible(false);
            jPanel5.setVisible(false);
            jPanel6.setVisible(false);
```

```
        jPanel7.setVisible(false);
        jPanel10.setVisible(false);
        selectedStructure=3;
    };

    if(jComboBox1.getSelectedItem().
    equals("Consola (10 elementos)")){

        jPanel1.setVisible(false);
        jPanel2.setVisible(false);
        jPanel3.setVisible(false);
        jPanel4.setVisible(true);
        jPanel5.setVisible(false);
        jPanel6.setVisible(false);
        jPanel7.setVisible(false);
        jPanel10.setVisible(false);
        selectedStructure=4;
    };

    if(jComboBox1.getSelectedItem().
    equals("Consola (20 elementos)")){

        jPanel1.setVisible(false);
        jPanel2.setVisible(false);
        jPanel3.setVisible(false);
        jPanel4.setVisible(false);
        jPanel5.setVisible(true);
        jPanel6.setVisible(false);
        jPanel7.setVisible(false);
        jPanel10.setVisible(false);
        selectedStructure=5;
    };

    if(jComboBox1.getSelectedItem().
    equals("Consola (50 elementos)")){

        jPanel1.setVisible(false);
        jPanel2.setVisible(false);
        jPanel3.setVisible(false);
        jPanel4.setVisible(false);
        jPanel5.setVisible(false);
        jPanel6.setVisible(true);
        jPanel7.setVisible(false);
        jPanel10.setVisible(false);
        selectedStructure=6;
    };
```

- *Combobox* que permite atribui um dano constante ao modelo através da alteração da rigidez de flexão de todos os elementos. O valor definido em cada caso é substituído na função *DanoConstante* dependendo do modelo em estudo

```
private void jComboBox4ActionPerformed(java.awt.event.ActionEvent evt) {

        switch(jComboBox4.getSelectedIndex()){

            case 0:DanoConstante((float)1);break;
            case 1:DanoConstante((float)0.9);break;
            case 2:DanoConstante((float)0.7);break;
            case 3:DanoConstante((float)0.5);break;
            case 4:DanoConstante((float)0.3);break;
            case 5:DanoConstante((float)0.1);break;
        }
    }
```

- Função onde é atribuído um valor de dano ao elemento selecionado

```
private static void DanoConstante (float f){

    switch(selectedStructure){
        case 1:
            for(int i = 0;i<10;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
            break;
        case 2:
            for(int i = 0;i<20;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
            break;
        case 3:
            for(int i = 0;i<50;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
            break;
        case 4:
            for(int i = 0;i<10;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
        break;
        case 5:
            for(int i = 0;i<20;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
            break;
        case 6:
            for(int i = 0;i<50;i++){
                selectedItem=i;
                setItemOnDissertacao(f);
            };
            break;
}
```

- Função que permite definir o dano nos elementos

```
public static void setItemOnDissertacao(float f){

        if (selectedStructure==1)
            diss.Viga10[selectedItem] = f;
        else if (selectedStructure==2)
            diss.Viga20[selectedItem] = f;
        else if (selectedStructure==3)
            diss.Viga50[selectedItem] = f;
        else if (selectedStructure == 4)
            diss.Consola10[selectedItem] = f;
        else if (selectedStructure==5)
            diss.Consola20[selectedItem] = f;
        else if (selectedStructure==6)
            diss.Consola50[selectedItem] = f;

}
```

- O *NetBeans IDE* tem a grande vantagem de possuir componentes na seção de *design* que estão ligados diretamente ao código fonte. Por exemplo, quando selecionado um botão (correspondente a um elemento do modelo) é possível atribuir-lhe diretamente as características pretendidas sem ser necessário definir a sua função. Este programa tem várias dezenas de botões, pelo que se torna repetitivo introduzir aqui o código de cada um individualmente. De seguida é dado como exemplo o aspecto tipo do código de um destes botões, que quando selecionado acede à janela de seleção de dano e é identificada a sua posição relativa aos outros elementos. No caso da viga de 10 elementos caso fosse escolhido o último botão (mais à direita) o que mudaria seria o *selectedItem*, que neste caso seria igual a 9 (9 porque a contagem no java começa no elemento 0).

```
private void button01ActionPerformed(
java.awt.event.ActionEvent evt) {

        Selecao_de_Dano a = new Selecao_de_Dano();
        selectedItem = 0;
        a.setVisible(true);
}
```

- Botão Calcular, que evoca à classe *Método_Curvaturas* e chama a função *plot*

```
 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        diss.plot(selectedStructure,selectedVMode);
}
```

- Botão Reiniciar, que evoca à classe *Método_Curvaturas* e faz com que o modelo em causa volte ao estado inicial

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

        diss.initVariables();
}
```

- Função *main*, é a função principal do programa

```
public static void main(String args[]) {

        diss.initVariables();
        selectedStructure = 1;
        selectedVMode = 0;

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interface().setVisible(true);
            }
        });
}
```