UNIVERSIDADE
BEIRA INTERIOR

Utilização de Programas de Cálculo Estrutural Através de Aplicações Externas – Contributo para a Análise Avançada de Estruturas

Sabino Mananci Sequeira Dumbo

Dissertação para obtenção do Grau de Mestre em

Engenharia Civil

(2º ciclo de estudos ou mestrado integrado)

Orientador: Prof. Doutor Clemente Martins Pinto

Abril de 2023

Declaração de Integridade

Eu, Sabino Mananci Sequeira Dumbo, que abaixo assino, estudante com o número de inscrição 40081 do curso Mestrado Integrado em Engenharia Civil da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 28 /04 /2023

Agradecimentos

Gostaria de expressar os meus mais sinceros agradecimentos primeiramente a DEUS, por tudo.

Ao meu orientador, Professor Doutor Clemente Martins Pinto, pelo empenho, interesse, cordialidade e disponibilidade que demonstrou na realização desta dissertação.

À minha mãe por todo o amor, por toda motivação e apoio. Sem ela não estaria a terminar essa parte da minha carreira académica.

Às minhas irmãs e ao meu irmão, a minha namorada, amigos(as) e colegas, pelos excelentes momentos vividos e pelo apoio nesta caminhada.

Finalmente agradecer, a todas as pessoas que de uma forma direta e indireta, ajudaram-me neste percurso.

Resumo

A modelação e análise de estruturas pode implicar um conjunto de tarefas repetidas, quer em pré-processamento, quer em pós-processamento. Este conjunto de tarefas é demorado. Por outro lado, certos tipos de problemas de análise estrutural implicam procedimentos de modelagem que não podem ser dissociados do uso de computação na fase de pré-processamento. Entre diversos exemplos, pode-se referir à modelação de estruturas com forma complexa, ou à análise com modelos de comportamento não linear.

A presente dissertação estuda a utilização de programas de cálculo estrutural a partir de aplicações externas, utilizadas para a modelação estrutural ou para o pós-processamento de resultados. Dentro desse objetivo são apresentados conceitos básicos de programação de computadores e a tecnologia base para o tipo de utilização referido.
A utilização de programas de análise estrutural a partir de aplicações exteriores é possível através da tecnologia de API (Application Programming Interface). Através da API, é possível realizar a interoperabilidade entre programas externos e analisadores estruturais, permitindo a sua funcionalidade e a criação de algoritmos e rotinas correspondentes para facilitar o uso desses programas. Para isso, é utilizado o padrão aberto (Open Standard), ou seja, APIs existentes em programas comerciais como o Robot Strucutral Analysis® da Autodesk e o Microsoft® Excel®. Assim, a dissertação tem como objetivo criar algoritmos e rotinas que possam ser facilmente utilizados pelos usuários para minimizar o tempo de entrada de dados, cálculo e modelagem, tornando a utilização de programas de análise estrutural mais eficiente e produtiva.

O presente trabalho focou-se essencialmente em minimizar o tempo de cálculo utilizando ferramentas existentes, desenvolvendo conceitos básicos em linguagem de programação para melhorar a execução de cálculos em análise avançada de estruturas, e todas as vantagens daí resultantes.

Palavras-chave:
API, Visual Basic, Robot Structural Analysis, OpenSees, OpenSeesPy, Formato .vtk, ParaView.

Abstract

The modeling and analysis of structures can imply a set of repeated tasks, either in pre-processing or in post-processing. This set of tasks is time consuming. On the other hand, certain types of structural analysis problems imply modeling procedures that cannot be dissociated from the use of computation in the pre-processing phase. Among several examples, one can refer to the modeling of structures with a complex shape, or to the analysis with models of non-linear behavior.

The present dissertation studies the use of structural calculation programs from external applications, used for structural modeling or for the post-processing of results. Within this objective, basic concepts of computer programming and the basic technology for the type of use mentioned are presented. The use of structural analysis programs from external applications is possible through API (Application Programming Interface) technology. Through the API, it is possible to perform interoperability between external programs and structural analyzers, allowing their functionality and the creation of corresponding algorithms and routines to facilitate the use of these programs. For this, the open standard (Open Standard) is used, that is, APIs existing in commercial programs such as Robot Strucutral Analysis® from Autodesk and Microsoft® Excel®. Thus, the dissertation aims to create algorithms and routines that can be easily used by users to minimize data entry, calculation and modeling time, making the use of structural analysis programs more efficient and productive.

The present work was essentially focused on minimizing calculation time using existing tools, developing basic concepts in programming language to improve the execution of calculations in advanced analysis of structures, and all the resulting advantages.

Key words:
API, Visual Basic, Robot Structural Analysis, OpenSees, OpenSeesPy, Format .vtk, ParaView.

Índice

Lista de Figuras

Lista de Quadros

Lista de Acrónimos

API: inglesa Application Programming Interface
Vtk: Visualization Toolkit
VB: Visual Basic
POO: Programação Orientada a Objetos
N: Esforço axial
Ux: Deslocamento do nó
My: Momentos fletores
HSLM: High Speed Load Model
RSA: Robot Strucutral Analysis®
COM: Component Object Model
ROM: Robot Object Model

Capítulo 1

Introdução

Em análise avançada de estruturas através de programas de cálculo automático, pode ter particular utilidade o recurso a aplicações externas. Essas aplicações podem tirar de modo eficiente a modelação estrutural, o controlo da análise estrutural, a obtenção e tratamento de resultados e os ajustes do modelo inicial em função dos objetivos pretendidos.

A interação entre programas de cálculo automáticos e aplicativos externos de vários tipos é geralmente estabelecida através de interfaces de programação de aplicações (API). Uma API é como uma biblioteca de rotinas e funções para a criação e desenvolvimento de aplicativos em determinadas linguagens de programação.

Diferentes programas de análise estrutural de natureza comercial possibilitam a utilização de API específicas para realizar todo o tipo de tarefas relacionadas com análise de estruturas. Essa utilização é particularmente útil para problemas mais complexos em que a modelação estrutural implique repetição de muitas tarefas, ou um pós-processamento de resultados mais elaborado. Na categoria de ferramentas comerciais, considerou-se para o presente trabalho o Robot Robot Strucutral Analysis® (RSA) [25]. O RSA permite a modelação e análise mais ou menos complexa, com um nível de desempenho adequado a diferentes tipos de problemas de projeto estrutural. A opção por essa ferramenta teve essencialmente a ver com a facilidade de acesso para utilização para fins educacionais.

No caso das aplicações com natureza não comercial optou-se pelo OpenSeesPy [29], em que o programa de análise estrutural OpenSees [30] é utilizado através de aplicações em linguagem Python [23]. OpenSees [30] é um programa de análise estrutural por elementos finitos com uma biblioteca de elementos muito completa e com algoritmos de análise adequado a problemas complexos. As suas potencialidades permitem simular o desempenho de sistemas estruturais sujeitos a diferentes tipos de solicitações, como é o caso da ação sísmica, em que é particularmente eficaz e completo. A escolha do OpenSees deveu-se à sua ampla utilização em diferentes contextos, desde o uso mais simples ao mais avançado e ao modo de acesso disponibilizado aos utilizadores. Embora se possam considerar evidentes as vantagens do recurso a APIs na modelação e análise estrutural, e particularmente na análise avançada de estruturas, não existem muitos trabalhos que apresentem conceitos, descrevam as APIs e agrupem exemplos, desde os

mais básicos aos mais complexos. Esse facto faz com que o recurso à API na análise estrutural, assim como das técnicas de programação, não sejam práticas correntes entre muitos profissionais da análise estrutural e particularmente, entre estudantes das áreas de Engenharia Civil.

1.1. Objetivos

Esta dissertação pretende apresentar conceitos e ferramentas associadas à utilização de API, desenvolver aplicações e apresentar exemplos que facilitem a utilização de um leque de possibilidades que sejam capazes de facilitar o trabalho de análise estrutural de casos complexos, bem como agilizar procedimentos repetitivos, entre outros.

1.2. Metodologia

Tendo em conta os objetivos supracitados, a metodologia utilizada consiste na apresentação de conceitos básicos, na seleção e no desenvolvimento de aplicações exemplificativas básicas. Posteriormente pretende-se desenvolver aplicações mais complexas, que permitam aferir as potencialidades e utilidade do tipo de procedimentos proposto.

Uma vez que a utilização de API implica procedimentos de programação, serão abordadas de modo sumário as linguagens de programação necessárias para os objetivos da presente dissertação, nomeadamente o Visual Basic (VB) e a linguagem Python.

Para além dos objetivos acima, pretende-se desenvolver soluções de visualização de resultados, essencialmente para as aplicações com o OpenSeesPy, recorrendo a ferramentas de acesso livre. Para tal, utiliza-se a biblioteca de representação gráfica vtk (Visualization Toolkit) [41].

1.3. Organização do Trabalho

Esta dissertação está dividida em 5 capítulos, incluindo esta inicial.

O Capítulo 2 descreve os conceitos básicos da API, funcionalidade da API do RSA, também fala sobre o software amplamente utilizado para análise de elementos finitos em engenharia civil e estrutural, ou seja, OpenSees, e as linguagens de programação utilizadas neste trabalho, principalmente VB e Python.

O Capítulo 3 é capítulo de aplicações, onde são descritas as aplicações com a API do RSA.

O Capítulo 4 é capítulo sobre análise de elementos finitos com OpenSees. Descreve a utilização de OpenSees e a biblioteca OpenSeesPy em Python, define o software utilizado para a modelação de geometria e criação de malhas, e a possibilidades de visualização. O conteúdo apresentado é suportado por aplicações exemplificativas.

No capítulo 5 são apresentadas as conclusões e respetivos desenvolvimentos futuros tendo em conta a utilização de programas de cálculo externos para análise avançada de estruturas.

Capítulo 2

Conceitos de API

2.1. Tecnologia API

A sigla API [1] deriva da expressão inglesa Application Programming Interface. Traduzindo na Língua portuguesa é Interface de Programação de Aplicações. A Tecnologia API ou simplesmente API [1] é definida como o conjunto de rotinas e padrões estabelecidas por um software para a utilização das suas funcionalidades por outros aplicativos. Ou seja, API é uma forma de comunicação entre sistemas, permitindo a integração e funcionamento entre eles, em que um deles fornece informações e serviços que podem ser utilizados pelo outro sem a necessidade de o sistema que consome o API, conhecer detalhes de implementação do software [1].

A API é responsável pelas ligações entre os recursos necessários para o bom desempenho de um software [2].

2.1.1. Função de uma API

Com o uso das APIs, não é necessário criar códigos personalizados para cada função que um programa for executar. Assim, as APIs agilizam os procedimentos repetitivos, simplificando um conjunto de tarefas. Ou seja, a utilização de uma aplicação pode ser feita através de outra, ou outras, que sejam mais eficientes para a realização das tarefas a realizar. No caso do RSA a API é utilizada a partir do Microsoft® Excel®, recorrendo a códigos em Visual Basic. No caso do OpenSees [30] a API é o OpenSeesPy [29], em que o OpenSees [30] é operado a partir de aplicações em linguagem Python, com as vantagens daí inerentes, dada a versatilidade e o espectro de ferramentas que essa linguagem de programação oferece.

2.2. Linguagem de Programação Visual Basic

Visual Basic (VB) é mais do que apenas uma linguagem de programação com o desenvolvimento do padrão Windows. O visual basic torna possível interagir com o teclado, tela mouse e impressora. A nova versão Visual Basic disponibiliza a criação de uma interface gráfica entre a aplicação computacional, o programa e o seu utilizador [15].

Visual Basic é uma linguagem guiada por eventos usados para desenvolver aplicações que correm em ambientes Microsoft Windows [15].

Como a Programação Orientada a Objetos (POO) [11] é uma abordagem de programação que pode ser aplicada a várias linguagens, incluindo VB [17], na presente dissertação os exemplos da linguagem VB foram realizados criando um exemplo de POO [11] relacionado com a criação classes e objetos.

Para se chegar aos objetivos pretendidos no presente trabalho, é importância conhecer a funcionalidade e aplicação do VB como ferramenta externa para o cálculo avançado de estruturas.

2.2.1. Visual Basic a partir do EXCEL

A utilização do VB na presente dissertação foi feita a partir do Microsoft Excel® [19], no contexto de programação orientada a objetos. Tendo em conta os objetivos da dissertação e o melhor entendimento das aplicações desenvolvidas, apresenta-se um exemplo básico de classes e objetos [12], relacionados com a modelação e análise de estruturas.

As barras de uma estrutura que incorporam o mesmo tipo de propriedades, mas que os seus valores variam de barra para barra podem ser entendidos como classes, a partir dos quais se vão definir objetos. Note-se que as barras de um sistema estrutural incorporam as variáveis, propriedades, procedimentos e eventos de um objeto. Dessa classe podem ser criados diferentes objetos, que são os tipos de barras com valores específicos das propriedades.

No caso de modelação e análise de sistemas estruturais, podem referir-se as classes, barras, nós e cargas, como possíveis classes a partir das quais se vão definir os diversos objetos. Os esquemas seguintes exemplificam a descrição acima para uma classe Barra e o objeto viga_ 1. Na modelação de um sistema estrutural serão definidos tantos objetos de classe barra como o número de barras desse sistema. Abaixo exemplifica-se uma classe barra e um objeto dessa classe (Quadro 1 e 2).

Quadro 1: Definição de uma classe barra e um objeto dessa classe de uma forma genérica.

Barra
Nome
Número
Área
Momento de Inércia
Nó 1
Nó 2
Material

Quadro 2: Definição de uma classe barra e um objeto dessa classe.

Viga_1 Como objeto da classe Barra	
Nome	viga_1
Número	1
Área	0.01
Momento de Inércia	0.001
Nó 1	1
Nó 2	2
Material	Aço

O Quadro 3, em VB, mostra a definição de uma classe de barra com as diferentes propriedades que podem ser atribuídas a uma barra. Os códigos Get e Let permitem receber e atribuir os dados que definem o objeto da classe barra.

Quadro 3: Definição do bloco de código de um objeto da classe barra.

```
Private Type ClassType
    Nome  As String
    Numero As Integer
    N_esq As Integer
    N_dir As Integer
    area As Double
    inercia As Double
End Type
Private bar_prop As ClassType
Public Property Get Nome() As String
    Nome = bar_prop.Nome
End Property
Public Property Let Nome(Value As String)
    bar_prop.Nome = Value
End Property
Public Property Get Numero() As Integer
    Numero = bar_prop.Numero
End Property
Public Property Let Numero(Value As Integer)
    bar_prop.Numero = Value
End Property
Public Property Get N_esq() As Integer
    N_esq = bar_prop.N_esq
End Property
Public Property Let N_esq(Value As Integer)
    bar_prop.N_esq = Value
End Property
Public Property Get N_dir() As Integer
    N_dir = bar_prop.N_dir
End Property
Public Property Let N_dir(Value As Integer)
    bar_prop.N_dir = Value
End Property
Public Property Get area() As Double
    area = bar_prop.area
End Property
Public Property Let area(Value As Double)
    bar_prop.area = Value
End Property
Public Property Get inercia() As Double
    inercia = bar_prop.inercia
End Property
Public Property Let inercia(Value As Double)
    bar_prop.inercia = Value
End Property
```

O Quadro 4 mostra a definição de um objeto da classe barra, em que são atribuídas as propriedades da barra.

Quadro 4: Aplicação do bloco de código de um objeto da classe barra.

```
Sub test()
    Dim Viga_1 As New Barra
    Dim no_1 As New No
    Viga_1.Nome = "Viga_2"
    Viga_1.Numero = 1
    Viga_1.N_esq = 1
    Viga_1.N_dir = 2
    Viga_1.area = 0.01
    Viga_1.inercia = 0.0001
    Folha1.Cells(1, 1) = Viga_1.Nome
    no_1.Numero = 1
    no_1.coor_X = 0
End Sub
```

A Figura 1 mostra o resultado da leitura do Quadro 4, onde pode ser vista a definição de uma classe de barra chamada Viga_2, conforme descrito no código.

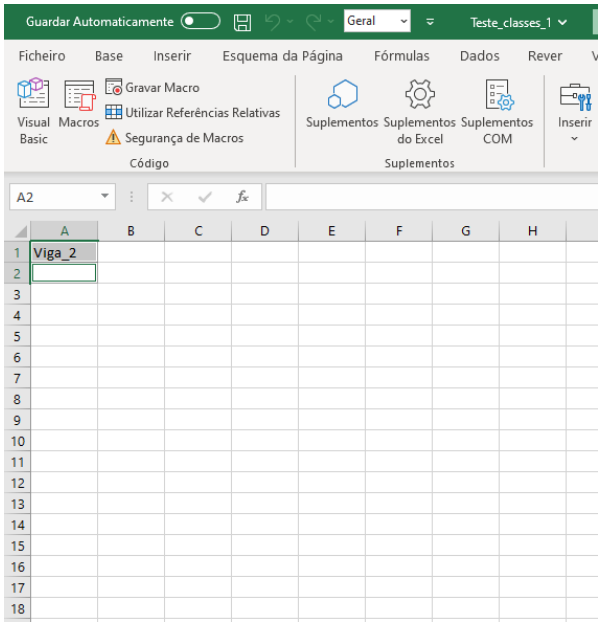


Figura 1: Representação de uma classe barra.

No Quadro 5 define-se uma classe para os nós do sistema estrutural e um objeto dessa classe. O Quadro 6 monstra as componentes a incorporar nessa classe e que definem cada objeto da mesma.

Quadro 5: Definição de um objeto da classe nó genérico.

NO
Número
Coordenada X (X_coor)
Coordenada Y (Y_coor)
Coordenada Z (Z_coor)
Apoiado ou não apoiado (tipo_apoio): Código para cada um dos tipos de apio possível

Quadro 6: Definição de um objeto da classe nó.

no_1 como objeto da classe NO	
Número	1
X_coor	0
Y_coor	0
Z_coor	0
tipo_apoio	1

O Quadro 7 mostra a definição de um objeto da classe nó.

Quadro 7: Definição do bloco de código de um objeto da classe nó.

```
Private Type ClassType
   Numero As Integer
   coor_X As Integer
   coor_Y As Integer
   coor_Z As Double
   tipo_apoio As Integer
End Type
Private node_prop As ClassType
Public Property Get Numero() As Integer
   Numero = node_prop.Numero
End Property
Public Property Let Numero(Value As Integer)
   node_prop.Numero = Value
End Property
Public Property Get coor_X() As Double
   coor_X = node_prop.coor_X
End Property
Public Property Let coor_X(Value As Double)
   node_prop.coor_X = Value
```

```
End Property
Public Property Get coor_Y() As Double
  coor_Y = node_prop.coor_Y
End Property
Public Property Let coor_Y(Value As Double)
  node_prop.coor_Y = Value
End Property
Public Property Get coor_Z() As Double
  coor_Z = node_prop.coor_Z
End Property
Public Property Let coor_Z(Value As Double)
  node_prop.coor_Z = Value
End Property
Public Property Get tipo_apoio() As Integer
  tipo_apoio = node_prop.tipo_apoio
End Property
Public Property Let tipo_apoio(Value As Integer)
  node_prop.tipo_apoio = Value
End Property
```

Para os objetos de uma classe, pode ser necessário realizar operações como o cálculo do comprimento de uma barra em função das coordenadas dos nós. Nesse caso, é preciso incluir uma função na classe barra que execute a operação a partir dos dados de entrada, no caso as coordenadas dos nós. Note-se que essa é uma função que só é usada para barras e não para nós. Assim, faz sentido que seja definida dentro da classe barra, o que torna o código principal da aplicação mais simples e claro.

Em relação ao código anterior da classe barra, é criada a função length_bar que executa o cálculo do comprimento das barras. Uma vez que essa função é para ser acedida a partir de diferentes aplicações que trabalham com barras, é definida com Public Function. Adicionalmente o resultado da função length pode ter que ser passado para diferentes aplicações que usem objetos da classe. Assim, a variável length é definida como Public length as Double.

O bloco de código abaixo mostra a adaptação da classe Barra, que passa a chamar-se Barra_length.

Quadro 8: Definição do bloco de código que apresenta a adaptação da classe Barra para Barra_length.

```
Private Type ClassType
   Nome  As String
   Numero As Integer
   N_esq As Integer
   N_dir As Integer
   area As Double
   inercia As Double
   x_cor_1 As Double
   y_cor_1 As Double
   z_cor_1 As Double
   x_cor_2 As Double
   y_cor_2 As Double
   z_cor_2 As Double
End Type
Public length As Double
Private bar_prop As ClassType
Public Property Get Nome() As String
   Nome = bar_prop.Nome
End Property
Public Property Let Nome(Value As String)
   bar_prop.Nome = Value
End Property
Public Property Get Numero() As Integer
   Numero = bar_prop.Numero
End Property
Public Property Let Numero(Value As Integer)
   bar_prop.Numero = Value
End Property
Public Property Get N_esq() As Integer
   N_esq = bar_prop.N_esq
End Property
Public Property Let N_esq(Value As Integer)
   bar_prop.N_esq = Value
End Property
Public Property Get N_dir() As Integer
   N_dir = bar_prop.N_dir
End Property
Public Property Let N_dir(Value As Integer)
   bar_prop.N_dir = Value
End Property
Public Property Get area() As Double
   area = bar_prop.area
End Property
Public Property Let area(Value As Double)
   bar_prop.area = Value
```

```
End Property
Public Property Get inercia() As Double
  inercia = bar_prop.inercia
End Property
Public Property Let inercia(Value As Double)
  bar_prop.inercia = Value
End Property
Public Function length_bar(x_cor_1, y_cor_1, z_cor_1, x_cor_2, y_cor_2, z_cor_2)
  length = ((x_cor_1 - x_cor_2) ^ 2 + (y_cor_1 - y_cor_2) ^ 2 + (z_cor_1 - z_cor_2) ^ 2) ^ 0.5
End Function
```

O código do Quadro 9 mostra a definição e utilização de objetos da classe adaptada anteriormente, produzindo o resultado correspondente ao comprimento de uma barra a partir das coordenadas dos nós de extremidade.

Quadro 9: Utilização do bloco de código que apresenta a adaptação da classe Barra para Barra_length.

```
Sub test_length_1()
  Dim Viga_1 As New Barra_length
  Dim no_1 As New No
  Dim no_2 As New No
  Dim aux As Double
  Viga_1.Nome = "Viga_2"
  Viga_1.Numero = 1
  Viga_1.N_esq = 1
  Viga_1.N_dir = 2
  Viga_1.area = 0.01
  Viga_1.inercia = 0.0001
  Folha1.Cells(1, 1) = Viga_1.Nome
  no_1.Numero = 1
  no_1.coor_X = 0
  no_1.coor_Y = 0
  no_1.coor_Z = 0
  no_2.Numero = 2
  no_2.coor_X = 5
  no_2.coor_Y = 5
  no_2.coor_Z = 5
  aux = Viga_1.length_bar(no_1.coor_X, no_1.coor_Y, no_1.coor_Z, no_2.coor_X, no_2.coor_Y,
no_2.coor_Z)
  Debug.Print "O comprimento da barra" & Viga_1.length
End Sub
```

A Figura 2 descreve o resultado correspondente ao comprimento de uma barra a partir das coordenadas das extremidades.

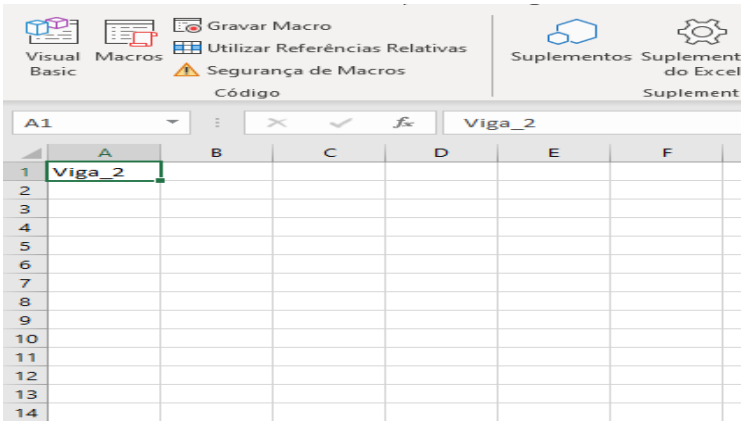


Figura 2: Representação do resultado após a leitura do código.

O objetivo seguinte é definir a classe material e diferentes objetos dessa classe. A classe material deve incorporar um nome do material, um identificador e as propriedades necessárias como o módulo de elasticidade, coeficiente de Poisson, módulo de distorção e densidade, conforme os Quadro 10 e 11.

Quadro 10: Definição genérica de uma classe material.

MATERIAL
Nome
Identificador
ELM
POISSON
DISTOR
DENSIDADE

Quadro 11: Descrição da classe material de um objeto.

MAT_1 como objeto da classe MATERIAL	
Nome	ACO
Identificador	1
ELM	210
POISSO	0.2
DISTOR	87.5
DENSIDADE	77

Os quadros seguidos (Quadro 12 e 13), definem a classe MATERIAL e um objeto associado com conjunto de propriedades indicados acima.

Quadro 12: Definição do bloco de código da classe material de um objeto.

```
Private Type ClassType
   Nome_mat  As String
   M_Identificador As Integer
   ELM As Double
   POISSON As Double
   DISTOR As Double
   DENSIDADE As Double
End Type
'Public M_Identificador As Integer
Private mat_prop As ClassType
Public Property Get Nome_mat() As String
   Nome_mat = mat_prop.Nome_mat
End Property
Public Property Let Nome_mat(Value As String)
   mat_prop.Nome_mat = Value
End Property
Public Property Get M_Identificador() As Integer
  M_Identificador = mat_prop.M_Identificador
End Property
Public Property Let M_Idenfificador(Value As Integer)
 mat_prop.M_Identificador = Value
End Property
Public Property Get ELM() As Double
   ELM = mat_prop.ELM
End Property
Public Property Let ELM(Value As Double)
   mat_prop.ELM = Value
End Property
Public Property Get POISSON() As Double
```

```
  POISSON = mat_prop.POISSON
End Property
Public Property Let POISSON(Value As Double)
  mat_prop.POISSON = Value
End Property
Public Property Get DISTOR() As Double
  DISTOR = mat_prop.DISTOR
End Property
Public Property Let DISTOR(Value As Double)
  mat_prop.DISTOR = Value
End Property
Public Property Get DENSIDADE() As Double
  DENSIDADE = mat_prop.DENSIDADE
End Property
Public Property Let DENSIDADE(Value As Double)
  mat_prop.DENSIDADE = Value
End Property
```

Quadro 13: Utilização do bloco de código da classe material de um objeto.

```
Public Sub test_material()
   Dim Mat_1 As New MATERIAL
   Mat_1.Nome_mat = "ACO"
   Mat_1.M_Idenfificador = 1
   Mat_1.ELM = 210
   Mat_1.POISSON = 0.2
   Mat_1.DISTOR = 87.5
   Mat_1.DENSIDADE = 77
   Debug.Print "O NOME DO MATERIAL É" & Mat_1.Nome_mat
End Sub
```

Cada barra do sistema estrutural tem associado um tipo de material com as propriedades correspondentes. Assim sendo, torna-se necessário ligar as caraterísticas do material a cada uma das barras. Para o efeito, existem diferentes possibilidades de fazer essa ligação, nomeadamente dentro da classe barra com os identificadores do material ou no programa principal. Em ambos os casos é necessário adaptar a classe barra.

Nos quadros 1 e 2, a classe barra é definida, sem atribuir suas propriedades. Nos quadros 5 e 6 mostram as propriedades necessárias que compõem a classe material, bem como sua formação. A combinação dos dois casos resulta na adaptação do código abaixo, que estabelece a conexão dentro da aplicação principal com as adaptações da classe barra, que agora inclui propriedades relacionadas ao material (Quadro 14). A definição dos objetos barra e objetos material é descrita nos quadros seguidos (Quadro 14 e 15), mostrando ao utilizador a densidade do material.

Quadro 14: Definição do bloco de código que mostra a relação entre a classe de barra e a classe de material.

```
Private Type ClassType
Nome As String
Numero As Integer
N_esq As Integer
N_dir As Integer
area As Double
inercia As Double
x_cor_1 As Double
y_cor_1 As Double
z_cor_1 As Double
x_cor_2 As Double
y_cor_2 As Double
z_cor_2 As Double
b_ELM As Double
b_POISSON As Double
b_DISTOR As Double
b_DENSIDADE As Double
End Type
Public length As Double
Private bar_prop As ClassType
Public Property Get Nome() As String
Nome = bar_prop.Nome
End Property
Public Property Let Nome(Value As String)
bar_prop.Nome = Value
End Property
Public Property Get Numero() As Integer
```

```
Numero = bar_prop.Numero
End Property
Public Property Let Numero(Value As Integer)
bar_prop.Numero = Value
End Property
Public Property Get N_esq() As Integer
N_esq = bar_prop.N_esq
End Property
Public Property Let N_esq(Value As Integer)
bar_prop.N_esq = Value
End Property
Public Property Get N_dir() As Integer
N_dir = bar_prop.N_dir
End Property
Public Property Let N_dir(Value As Integer)
bar_prop.N_dir = Value
End Property
Public Property Get area() As Double
area = bar_prop.area
End Property
Public Property Let area(Value As Double)
bar_prop.area = Value
End Property
Public Property Get inercia() As Double
inercia = bar_prop.inercia
End Property
Public Property Let inercia(Value As Double)
bar_prop.inercia = Value
End Property
Public Function length_bar(x_cor_1, y_cor_1, z_cor_1, x_cor_2, y_cor_2, z_cor_2)
length = ((x_cor_1 - x_cor_2) ^ 2 + (y_cor_1 - y_cor_2) ^ 2 + (z_cor_1 - z_cor_2) ^ 2) ^ 0.5
End Function
Public Property Get b_ELM() As Double
b_ELM = bar_prop.b_ELM
End Property
Public Property Let b_ELM(Value As Double)
bar_prop.b_ELM = Value
End Property
Public Property Get b_POISSON() As Double
b_POISSON = bar_prop.b_POISSON
End Property
Public Property Let b_POISSON(Value As Double)
bar_prop.b_POISSON = Value
End Property
Public Property Get b_DISTOR() As Double
b_DISTOR = bar_prop.b_DISTOR
```

```
End Property
Public Property Let b_DISTOR(Value As Double)
bar_prop.b_DISTOR = Value
End Property
Public Property Get b_DENSIDADE() As Double
b_DENSIDADE = bar_prop.b_DENSIDADE
End Property
Public Property Let b_DENSIDADE(Value As Double)
bar_prop.b_DENSIDADE = Value
End Property
```

Quadro 15: Aplicação do bloco de código que mostra a relação entre a classe de barra e a classe de material.

```
Sub test_material_1()
Dim Viga_2 As New BARRA_MATERIAL
Dim Mat_1 As New MATERIAL
Mat_1.ELM = 210
Mat_1.POISSON = 0.2
Mat_1.DISTOR = 87.5
Mat_1.DENSIDADE = 77
Viga_2.b_ELM = Mat_1.ELM
Viga_2.b_POISSON = Mat_1.POISSON
Viga_2.b_DISTOR = Mat_1.DISTOR
Viga_2.b_DENSIDADE = Mat_1.DENSIDADE
Debug.Print "A densidade da viga 2" &
Viga_2.b_DENSIDADE
End Sub
```

O código apresentado no Quadro 16 mostra uma possibilidade de atribuir materiais às barras. O modo exemplificado permite definir o material uma vez e usá-lo várias vezes no código atribuído a diferentes barras.

Quadro 16: Aplicação do bloco de código que permite atribuir materiais às barras.

```
Sub test_material_1()
Dim Viga_2 As New BARRA_MATERIAL
Dim Viga_3 As New BARRA_MATERIAL
Dim Mat_1 As New MATERIAL
Dim Mat_2 As New MATERIAL
Mat_1.Nome_mat = "Aço"
Mat_1.ELM = 210
Mat_1.POISSON = 0.2
Mat_1.DISTOR = 87.5
Mat_1.DENSIDADE = 77
Mat_2.ELM = 38
Mat_2.POISSON = 0.15
Mat_2.DISTOR = 15
Mat_2.DENSIDADE = 25
Viga_2.b_ELM = Mat_1.ELM
Viga_2.b_POISSON = Mat_1.POISSON
Viga_2.b_DISTOR = Mat_1.DISTOR
Viga_2.b_DENSIDADE = Mat_1.DENSIDADE
Viga_3.b_ELM = Mat_2.ELM
Viga_3.b_POISSON = Mat_2.POISSON
Viga_3.b_DISTOR = Mat_2.DISTOR
Viga_3.b_DENSIDADE = Mat_2.DENSIDADE
Debug.Print "A densidade da viga 2" & Viga_2.b_DENSIDADE
Debug.Print "A densidade da viga 3" & Viga_3.b_DENSIDADE
End Sub
```

2.2.2. Linguagem de Programação Python

O Python [24], é uma linguagem de programação interpretada, orientada à objetos de alto nível e com semântica dinâmica.

O Python é uma das linguagens que mais tem crescido à sua compatibilidade, à sua facilidade para funcionar na maioria dos sistemas operacionais e capacidade de auxiliar outras linguagens [20].

2.2.2.1. Programação orientada aos objetos em Python

A linguagem Python é orientada a objetos e apresenta grande potencial para o desenvolvimento de aplicações relacionadas com a análise de estruturas. O exemplo de

código abaixo mostra a utilização de classes e objetos para o cálculo de curvas parabólicas. O tipo de curva parabólica pode ser considerado como uma classe da qual resultam diferentes objetos que são as parábolas com um comprimento, flecha e número de pontos associados.

Quadro 17: Definição genérica de um objeto para o cálculo de uma parabólica.

PARABOLA
Número de pontos
Flecha
Vão

Quadro 18: Definição de um objeto para o cálculo de uma parabólica.

Parabola_1 Como objeto da classe parábola	
Número de pontos	10
Flecha	2
Vão	20

Para além das propriedades de cada parábola, associa-se à classe a fórmula de cálculo da coordenada z em função das coordenadas x e y. Por outro lado, a classe deve conter um processo de fornecimento dos resultados. O código do Quadro 25 mostra o cálculo de uma parábola definindo uma classe parábola e objetos dessa classe. A partir do exemplo, é possível constatar que numa aplicação mais complexa que implique o uso de diferentes parábolas bastará definir objetos dessa classe, sem ser necessário programar as expressões matemáticas associadas.

Conforme [22, 23], a função_init_ () é usada para atribuir valores às propriedades dos objetos da classe. No caso de uma parábola, os valores a considerar são a flecha, o vão e o número de pontos. Assim, justifica-se que os argumentos da função sejam flecha, vão e os números de pontos. O argumento self é uma referência que permite aceder às variáveis que pertencem à classe. As variáveis que definem as propriedades do objeto, flecha, vão e pontos são escritas no código da classe como self.flecha, self.vao e self.pontos. Para cada objeto da classe parábola terão de ser realizadas operações ou procedimentos em função das necessidades. Por exemplo, pode ser necessário mostrar ao utilizador os valores introduzidos para a flecha, para o vão e para o número de pontos. Adicionalmente, é necessário determinar as coordenadas da curva em função dos parâmetros introduzidos. Em programação orientada à objetos, essas operações

uniformes a todos os objetos da classe definem-se como métodos. No exemplo em causa são definidos métodos para mostrar os parâmetros fornecidos, def show(self) e para determinar as coordenadas da parábola, def cot(self).

No método cota são realizadas operações matemáticas e para isso torna-se necessário utilizar a biblioteca de funções numéricas numpy [47]. Desse modo, essa biblioteca tem de ser inicializada no início do código e a utilização das suas funções é feita através do prefixo numpy.

As coordenadas de parábola serão numa estrutura de dados (array) denominada cota. As cotas da parábola são inicializadas com zero através da função numpy.zero, que tem como argumentos o número de linhas que é igual ao número de pontos da parábola e uma coluna. Na função numpy.zero, para além da indicação do número de linhas e de colunas, é necessário indicar o tipo de dados. No caso, serão reais, definidos como dtype=float. Assim, a estrutura de dados cota é inicializada a zeros através de numpy.zeros((self.npontos,1),dtype=float). Para calcular os valores de cota da parábola, é utilizado um ciclo "for" que se repete self.npontos+1 vezes. A cada iteração do ciclo, é calculado um novo valor de cota correspondente à abcissa atual. A variável aux é inicializada com o valor de "zero" e é atualizada a cada iteração do ciclo com o valor de self.vao/self.npontos. A cada iteração do ciclo, o valor da cota é atribuído à matriz cota na linha correspondente à abcissa atual. O valor da abcissa atual é obtido a partir do valor de aux, que é atualizado a cada iteração.

O valor da cota para cada valor da abcissa é atribuído à estrutura de dados cota, que em cada passo do ciclo for atribui o respetivo valor à cota[i,0], ou seja, linha i e primeira coluna.

O valor da abcissa tem de ser atualizado após o cálculo de mais uma cota da parábola, o é feito por aux=aux+self.vao/self.npontos. Ou seja, o novo valor de aux é igual ao anterior para um segmento correspondente à divisão do vão pelo número de pontos.
O resultado das operações realizadas dentro da função é passado para fora da mesma através da instrução return(cota). Sem essa instrução, as coordenadas da parábola eram calculadas, mas não havia acesso às mesmas.

Na classe parábola é possível definir métodos adicionais que respondam às necessidades do utilizador. Por exemplo, pode definir-se que seja criado um ficheiro com as cotas da parábola cada vez que é criado um objeto parábola. Para tal, recorre-se à função

numpy.savetxt da biblioteca numpy, com os argumentos correspondentes ao nome do ficheiro e a estrutura de dados cota que é aquela que se quer arquivar.

Na definição do método show, dão-se instruções para que sejam apresentados a flecha e o vão, recorrendo à função print. Assim, o valor da flecha é mostrado através do código print(self.flecha), verificando-se o prefixo self como referência para a variável flecha.

Quadro 19: Aplicação do método show e do código self.

```
import numpy
class Parabola:
  def __init__(self, flecha, vao, npontos):
    self.flecha = flecha
    self.vao = vao
    self.npontos = npontos
  def cota(self):
    cota = numpy.zeros((self.npontos + 1, 1), dtype=float)
    aux = 0
    for i in range(self.npontos + 1):
      cota[i, 0] = 4 * aux * self.flecha / (self.vao * self.vao) * (self.vao - aux)
      aux=aux + self.vao / self.npontos
      #print(a)
    return cota
  def show(self):
    print("Flecha")
    print(self.flecha)
    print("Vão")
    print(self.vao)
    print("Cota")
    print(self.cota())
```

Depois de criada a classe, torna-se possível criar diferentes objetos dessa classe no programa principal sem ter de programar os métodos associados. No exemplo do Quadro 20, dois objetos da classe parábola chamados parábola 1 e parábola 2 são definidos.

Quadro 20: Definição da classe parábola.

```
parabola_1=Parabola()
parabola_1.flecha=4
parabola_1.npontos=10
parabola_1.vao=2
parabola_1.cota()
parabola_2=Parabola()
parabola_2.flecha=8
parabola_2.npontos=20
parabola_2.vao=3
parabola_2.cota()
```

Os dois objetos da classe parábola são definidos por parábola_1=parabola() e parábola_2=parabola(). Se a aplicação for executada, é possível constatar que o utilizador não recebe nenhuma informação. Tal acontece porque não é chamado o método da classe criado para apresentar resultados, no caso o método show(def). Se o código acima for adaptado com a chamada do método parábola_1. show() e parábola_2.show() (Quadro 21) será possível constatar que passam a ser mostrados ao utilizador os elementos de cada uma das parábolas.

Quadro 21: Aplicação do método show(def), para mostrar os resultados.

```
parabola_1=Parabola()
parabola_1.flecha=4
parabola_1.npontos=10
parabola_1.vao=2
parabola_1.cota()
parabola_2=Parabola()
parabola_2.flecha=8
parabola_2.npontos=20
parabola_2.vao=3
parabola_2.cota()
parabola_1.show()
  parabola_2.show()
```

No exemplo seguinte são criadas e definidas classes e objetos que podem ser considerados na modelação e análise de um sistema estrutural, correspondentes a barras, materiais e nós. O Quadro 22 mostra a criação da classe barra e a definição de um objeto dessa classe.

Quadro 22: Definição da classe barra.

```
class barra():
   def __init__(self, numero, n_esq, n_dir, area, inercia):
      self.numero = numero
      self.n_esq = n_esq
      self.n_dir = n_dir
      self.area = area
      self.inercia = inercia
barra_1=barra()
barra_1.numero=1
barra_1.n_esq=1
barra_1.n_dir=2
barra_1.area=0.01
barra_1.inercia=0.001
print(barra_1.numero)
```

Na variante seguinte, tal como efetuado com o exemplo VB, adiciona-se a possibilidade de determinar o comprimento da barra, o que implica adicionar os argumentos correspondentes às coordenadas dos nós e uma função para determinar o comprimento. No caso, é possível definir uma nova classe barra definida como barra_length. Nesse caso, podem definir-se objetos de cada uma das classes em função das necessidades. Os Quadros 23 e 24 mostram a definição dessa classe e a sua utilização criando o objeto barra_2 e mostrando o comprimento.

Quadro 23: Definição da classe barra_length.

```
class barra_length():
   def __init__(self, numero, n_esq, n_dir, x_1, y_1, z_1, x_2, y_2, z_2, area, inercia):
      self.numero = numero
      self.n_esq = n_esq
      self.n_dir = n_dir
      self.area = area
      self.inercia = inercia
      self.x_1 = x_1
      self.y_1 = y_1
      self.z_1 = z_1
      self.x_2 = x_2
      self.y_2 = y_2
      self.z_2 = z_2
      def length(self):
      length = ((self.x_1 - self.x_2)^2 + (self.y_1 - self.y_2)^2 + (self.z_1 - self.z_2)^2)**0.5
    return length
```

Quadro 24: Definição da classe barra_length da barra 2.

```
  barra_2=barra_length()
  barra_2.x_1=0
  barra_2.y_1=0
  barra_2.z_1=0
  barra_2.x_2=0
  barra_2.y_2=0
  barra_2.z_2=0
  barra_2.length()
print(barra_2.length())
```

Os códigos apresentados acima aparecem completos no Quadro 25. Observe que nos Quadros 20 a 24, conforme os objetos da classe são customizados separadamente. Enquanto o Quadro 25 simplificou esta descrição conforme definido nas linhas 18 e 20 do Quadro 25.

Quadro 25: Aplicação geral dos códigos.

```
import numpy
class Parabola:
   def __init__(self, flecha, vao, npontos):
      self.flecha = flecha
      self.vao = vao
      self.npontos = npontos
   def cota(self):
      cota = numpy.zeros((self.npontos + 1, 1), dtype=float)
      aux = 0
      for i in range(self.npontos + 1):
         cota[i, 0] = 4 * aux * self.flecha / (self.vao * self.vao) * (self.vao - aux)
         aux += self.vao / self.npontos
      return cota
   def show(self):
      print("Flecha:", self.flecha)
      print("Vão:", self.vao)
      print("Cota:", self.cota())
parabola_1 = Parabola(4, 2, 10)
parabola_2 = Parabola(8, 3, 20)
parabola_1.show()
parabola_2.show()
class Barra:
   def __init__(self, numero, n_esq, n_dir, area, inercia):
      self.numero = numero
      self.n_esq = n_esq
      self.n_dir = n_dir
      self.area = area
```

```
        self.inercia = inercia
barra_1 = Barra(1, 1, 2, 0.01, 0.001)
print(barra_1.numero)
class BarraLength:
    def __init__(self, numero, n_esq, n_dir, x_1, y_1, z_1, x_2, y_2, z_2, area, inercia):
        self.numero = numero
        self.n_esq = n_esq
        self.n_dir = n_dir
        self.area = area
        self.inercia = inercia
        self.x_1 = x_1
        self.y_1 = y_1
        self.z_1 = z_1
        self.x_2 = x_2
        self.y_2 = y_2
        self.z_2 = z_2
    def length(self):
        length = ((self.x_1 - self.x_2)^2 + (self.y_1 - self.y_2)^2 + (self.z_1 - self.z_2)^2)**0.5
        return length
barra_2 = BarraLength(2, 1, 2, 0, 0, 0, 5, 5, 5, 0.01, 0.001)
print(barra_2.length())
```

2.3. Exemplos de ferramentas com API

2.3.1. API do Robot Strucutral Analysis

A API do RSA [25, 44] incorpora um conjunto de classes e funções que permitem a sua completa utilização a partir de aplicações exteriores. Uma das possibilidades de operar com a API do RSA é a partir de módulos em VB através do Microsoft® Excel®. Todas as operações que se podem realizar no ambiente do próprio programa, desde a modelação, controlo e análise estrutural e pós-processamento de resultados, podem também ser feitas num modo de API. Assim, torna-se possível a modelação de estruturas complexas ou com operações repetidas e o pós-processamento de resultados através de aplicações que facilitem o trabalho de análise estrutural.

A API é baseada em tecnologia COM (Component Object Model) e permite que outras aplicações possam automatizar tarefas ou realizar análises estruturais de forma programada às especificidades do problema [25, 44]. Os membros do programa e os tipos de dados usados para definir estruturas são descritos em Robot Object Model (ROM) por meio de interfaces apropriadas [25, 44]. Uma interface, é entendida como, um conjunto de dados, mutuamente ligados de maneira lógica, e um conjunto de operações que

podem ser executados nos dados [25, 44]. Os dados são chamados de atributos ou membros, e as operações que podem ser executadas são chamadas de funções ou métodos da interface [25, 44]. O conjunto de funções da interface define sua funcionalidade. Cada interface tem um nome exclusivo.

Algumas das funcionalidades que podem ser acessadas por meio da API do RSA incluem:

- Criação e modificação de modelos estruturais;
- Execução de análises estáticas e dinâmicas;
- Acesso aos resultados de análises;
- Exportação de resultados para outros formatos de dados;
- Acesso a informações sobre materiais, seções transversais e outras propriedades de elementos estruturais.

Conforme [25, 44], algumas das principais classes e interfaces na API do RSA incluem:

- IRobotApplication: A interface principal para a aplicação RSA, que permite aceder a objetos e métodos na API.
- IRobotDocument: A interface para documentos RSA, que permite abrir, fechar e salvar modelos estruturais.
- IRobotModel: A interface para modelos estruturais, que permite criar e modificar elementos estruturais, como vigas, colunas e lajes.
- IRobotAnalysis: A interface para análises de estruturas, que permite executar análises de cargas e análises de elementos finitos.
- IRobotSelection: A interface para seleções de elementos estruturais, que permite selecionar e modificar elementos específicos no modelo.
- IRobotResults: A interface para resultados de análises, que permite aceder aos resultados de análises de estruturas, como tensões, deformações e forças internas.

A API do RSA incorpora também diferentes classes e interfaces que permitem a interação com outras funcionalidades. No Anexo A-Robot Open Standard da referência [25, 44], encontram-se descritas todas essas interfaces e funcionalidades. Nesta secção é feita uma descrição geral que será complementada com os exemplos apresentados no capítulo 3.

2.3.2. OpenSees (openseespy)

O OpensSeesPy [29] é uma interface que permite trabalhar com o OpenSees [30] através de aplicações escritas em linguagem Python. Desse modo, torna-se possível associar as potencialidades de análise estrutural do OpenSees com a versatilidade de multiplicidade de aplicações e bibliotecas do Python. No presente trabalho é mostrada a utilização do OpenSees [30] a partir de dados processados em folhas de cálculo. Adicionalmente, são criadas aplicações de visualização quer do modelo estrutural que dos resultados. Note-se que o OpenSees [30] não tem interface gráfica, o que em certos casos constitui uma limitação.

A referência [29] apresenta as ferramentas do OpenSeesPy [29] que permitem a modelação, análise estrutural e pós-processamentos de resultados. Essas ferramentas correspondem a diferentes classes das quais resultam diferentes objetos, conforme as necessidades. Na criação de modelos em OpenSeesPy [29], tal como noutra linguagem, é importante que o utilizador tenha uma noção de mecânica estrutural que facilmente lhe permita perceber os dados de entrada necessários, dependendo do tipo de modelo e análise estrutural. A sequência é lógica, assim como os parâmetros a considerar na definição dos objetos de determinada classe, seja para definição de tipos de elementos finitos, materiais, propriedades geométricas, nós, apoios, etc… Nas secções seguintes descreve-se de forma sequencial um exemplo de modelação e análise estrutural disponível no ponto 4 e adaptado da referência [43].

Numa aplicação em Python o primeiro passo é a importação dos módulos que serão utilizados, dentro do alargado conjunto de módulos disponível em Python. Esses módulos terão de ser previamente instalados.

As três linhas do Quadro 26 importam os módulos relativos ao OpenSeesPy [29], numpy [47], com funções e métodos de análise numérica, e matplotlib.pyplot [48], com funções de representação gráfica. Por exemplo, o módulo numpy fica associado ao identificador np que é usado como prefixo np. ao longo do código para utilizar as diferentes funções.

Quadro 26: Importação dos módulos necessários.

```
From openseespy.opensees import*
Import numpy as np
Import matplotlib.pyplot as plt
```

O Quadro 27 mostra a aplicação do comando wipe() que limpa modelos anteriores, ou seja, evita que variáveis associadas a esses modelos sejam consideradas numa nova análise. Quando esta preocupação não é tida em conta, podem ocorrer erros de resultados ou outros de difícil identificação ou resolução.

Quadro 27: Definição do comando Wipe().

```
Wipe()
```

O problema considerado é bidimensional com dois graus de liberdade por nó. Em OpenSeesPy [29] isso é feito através do comando model, que se descreve no Quadro 28. Considerou-se um problema 2D com dois graus de liberdade por nó. Adicionalmente, é considerado um tipo de modelo "basic". O código do quadro 28 faz a definição do modelo, em que no caso -ndm corresponde à dimensão do modelo e -ndf ao parâmetro opcional de graus de liberdade por nó, igual a 2.

Quadro 28: Descrição do comando model.

```
Model('basic', ´-ndm', 2, ´-ndf', 2)
```

Após a definição dos parâmetros globais do modelo, passa-se à definição da geometria, nomeadamente dos nós da estrutura, criando objetos da classe node, que tem como argumentos o número do nó, as coordenadas correspondentes e outras propriedades como aceleração, velocidade ou massa. Cada uma das propriedades é definida através de um identificador correspondente que precede o valor da respetiva propriedade (Quadro 29).

Quadro 29: Descrição do comando para a criação dos nós da estrutura.

```
Node(node Tag,*crds, '-ndf', 'ndf','-mass',*mass,'-disp',*disp,'-vel',*vel,'-accel',*accel)
```

O código do Quadro 30 cria os nós da estrutura. Note-se que no caso das coordenadas são apenas introduzidas a abcissa e a ordenada, uma vez que o problema foi previamente definido como bidimensional.

Quadro 30: Definição dos nós da estrutura.

```
node (1, 0.0, 0.0)
node (2, 5, 0.0)
node (3, 10, 0.0)
node (4, 5, 5)
```

Em Python as linhas de código acima podem ser reescritas de forma alternativa, associando as coordenadas dos nós a listas, que podem ser utilizadas noutras partes do código. Assim, sempre que for necessário utilizar as coordenadas dos nós, bastará chamar as listas de dados definidas.

Quadro 31: Descrição das listas dos nós.

```
no_1= [0,0]
no_2= [5,0]
no_3= [10,0]
no_4= [5,5]
node (1, *no_1)
node (2, *no_2)
node (3, *no_3)
node (4, *no_4)
```

Seguindo a lógica de modelação de um sistema estrutural, o passo seguinte corresponde a definição das condições de apoio, recorrendo ao comando fix. Os parâmetros a considerar são o número de apoio e os graus de liberdade a restringir, sendo que o argumento 1 é para restrição do grau de liberdade e zero é para não restrição, conforme o Quadro 32. Note-se que o número de elementos a considerar para caraterizar o tipo de apoio de cada nó, depende do número de graus de liberdade definido previamente. No exemplo em análise bastará indicar dois valores, associados aos dois graus de liberdade por nó previamente definidos. No Quadro 32 definem-se apoios duplos, com translações impedidas nos nós 1, 2 e 3 previamente definidos.

Quadro 32: Definição dos apoios da estrutura.

```
fix (1, 1, 1)
fix (2, 1, 1)
fix (3, 1, 1)
```

Antes de se definiram os elementos do sistema estrutural, como é o caso das barras, é necessário definir os modelos de comportamento dos materiais que lhes estarão associados. O OpenSees [30], contém uma vasta biblioteca de modelos de materiais,

possibilitando análise avançada de diversos tipos de problemas. Cada tipo de material corresponde a uma classe da qual são definidos objetos. Os argumentos a considerar na definição de cada tipo de material dependem do próprio modelo associado e da sua definição. Assim, a utilização avançada do OpensSees [30] implica uma análise da diversa documentação [29], que descreve cada um dos tipos de modelos de material.

O OpenSees [30] contempla duas classes de material que são a base para a definição de objetos relativos a diferentes tipos de material, com diferentes modelos de comportamento e, consequentemente, diferentes parâmetros associados. Uma das classes é a classe "uniaxialMaterial" que define relações uniaxiais entre tensão e extenção na caraterização do comportamento mecânico dos materiais. A outra classe é a "nDMaterial" que, conforme [29], define relações tensão-extensão nos pontos de gauss de um elemento contínuo.

Um objeto classe "uniaxialMaterial" é criado conforme se no Quadro 33, de acordo com [29]. O argumento "matType" define o tipo de material pretendido, desde aço a betão ou outro. Assim, é necessário conhecer denominações adotadas no OpenSees [30] para cada tipo de material. Por exemplo, um dos modelos para o aço é denominado de "Steel01" e para o betão "Concrete01". O argumento "matTag" é o identificador do material a ser usado na análise pretendida. Trata-se de uma variável inteira que é utilizada noutros comandos para "chamar" o material previamente definido. Numa mesma aplicação o que identifica os diferentes modelos de materiais adotados é o identificador "matTag".

Quadro 33: Definição da classe uniaxialMaterial.

```
uniaxialMaterial(matType, matTag, *matArgs)
```

Por exemplo, os objetos do tipo "Steel01" [29] são definidos conforme se demostra no Quadro 34. Os argumentos iniciais indicam o tipo de material e o identificador e os restantes parâmetros as propriedades específicas do material, sendo algumas delas opcionais. No Quadro 34 consta a definição de um tipo de aço com tensão limite de cedência de 355 MPa e 206000 MPa, com comportamento elástico-perfeitamente plástico, o que corresponde a b=0. Esse material é identificado por "matTag" igual a 1.

Quadro 34: Definição dos objetos do tipo "Steel01".

```
uniaxialMaterial('Steel01', matTag, Fy, E0, b, a1, a2, a3, a4)
```

No Quadro 35 mostra-se a definição de um tipo de aço com limite de cedência de 355 MPa e módulo de elasticidade de 206000 MPa, com comportamento elástico-perfeitamente plástico. O parâmetro b define a relação entre a deformação plástica e a tensão de escoamento do material (de aço), ou seja, define a resistência ou o endurecimento do material, em que um valor de zero corresponde ao comportamento elástico-perfeitamente plástico. O tipo de aço definido tem associado o identificador "matTag" igual a 1.

Quadro 35: Aplicação dos objetos do tipo "Steel01".

```
uniaxialMaterial('Steel01', 1, 355, 206000, 0)
```

A definição de um objeto da classe "uniaxialMaterial" [29] do tipo betão é feita como se indica no Quadro 36. Note-se que em comparação ao aço os parâmetros a considerar são diferentes, o que está relacionado com as especificidades do tipo de material, que têm de ser conhecidas pelo utilizador. No caso "fpc" corresponde à resistência à compressão aos 28 dias (introduzida com valor negativo), "epsc0" a extensão correspondente à tensão resistente máxima, "fpcu" a resistência última do betão e "epsU" extensão última do betão. Conforme [29], este material implica a consideração de resistência à tração do betão nula.

Quadro 36: Definição da classe uniaxialMaterial. Do tipo "Betão".

```
uniaxialMaterial('Concrete01', matTag, fpc, epsc0, fpcu, epsU)
```

No exemplo de aplicação em análise surge definido um material elástico com módulo de elasticidade igual a 30000 MPa, com identificador "matTag" igual a 2. O tipo de material nesse caso é definido por "Elastic" no Quadro 37.

Quadro 37: Definição de um material elástico.

```
uniaxialMaterial("Elastic", 1, 206000)
```

Após a definição dos nós, e do material é possível definir os elementos do sistema estrutural através da biblioteca de elementos do OpenSees [30]. Tal como a biblioteca de materiais, o software tem uma vasta biblioteca de elementos finitos. A definição de um elemento finito é feita de forma similar a um material tendo como argumentos o tipo de elemento ("eleType"), identificador ("eleTag"), nós do elemento ("eleNodes") e os parâmetros específicos que relativos a cada tipo de elemento, como propriedades geométricas, material ou outros. Em termos genéricos a definição de um elemento finito é feita conforme se descreve no Quadro 38.

Quadro 38: Definição de um elemento finito.

```
element(eleType, eleTag, *eleNodes, *eleArgs)
```

O número de nós e os argumentos dependem do tipo de elemento finito. Por exemplo para um elemento de barra com nós articulados é apenas necessária a área de secção, num elemento plano de viga a área de secção e o momento de inércia e num momento de barra tridimensional a área de secção e os momentos de inércia em torno de todos os eixos. O número de nós também é dependente do tipo de elemento. No elemento do tipo barra são necessários dois nós e um elemento tipo placa três ou mais nós.

Os argumentos correspondentes a nós e aos parâmetros do elemento podem ser introduzidos diretamente na função ou através de listas de dados, conforme já se exemplificou. Nos quadros 39 a 41 apresentam-se três formas distintas de definir um elemento do tipo treliça. No caso, será o elemento identificado como 1, ligando os nós 1 e 4, com área de secção de 0.05 m2 e material previamente definido com o identificador 1. Na segunda possibilidade (Quadro 40), os nós do elemento 1 são definidos numa lista e as propriedades noutra lista. Esta opção implica a criação de mais variáveis, mas permite aceder às propriedades desse elemento em parte da aplicação, o que implica vantagens relativamente ao controlo e manipulação do modelo.

Quadro 39: Aplicação do comando que descreve um elemento finito.

```
element("Truss",1,1,4,0.05,1)
```

Quadro 40: Aplicação do comando que descreve um elemento finito em lista.

```
nod_el1=[1,4]
el1_prop=[0.05,1]
element("Truss",1,*nod_el1,*el1_prop)
```

Uma possibilidade alternativa corresponde a definir uma variável para a área de secção definida por a_sec_1 e depois colocar essa variável na função que cria o elemento finito. Desse modo, é possível utilizar o mesmo tipo de secção noutros elementos finitos.

Quadro 41. Definição da área da secção.

```
nod_el1=[1,4]
a_el1=0.05
el1_prop=[a_el1,1]
element("Truss", 1, *nod_el1,*el1_prop)
```

Com o conjunto de código descrito acima fica definido o modelo estrutural, e pode passar-se à definição das cargas. No caso do OpenSees [30] e tendo em conta as capacidades de análise em problemas mais avançados, a definição de cargas tem de ser precedida de outros parâmetros que a seguir se explicam.

Em primeiro lugar é necessário definir o domínio tempo e o fator de carga associado aos carregamentos. Desse modo, indica-se o tipo de aplicação de carga relativo ao problema estrutural em análise. O tipo de variação de carga no tempo é diverso, podendo ser definido por uma função constante, uma função linear, trigonométrica, triangular, retangular, na forma de impulso ou com uma função específica definida pelo utilizador. As especificidades de cada uns dos tipos acima surgem definidas na referência [29] .

Numa análise estática simples podem ser considerados apenas o tipo de variação, constante ou linear e o identificador. Os quadros seguidos (Quadro 42 e 42), definem uma função temporal constante, com identificador 1 e fator de carga 1. Numa variação linear é obtido o mesmo resultado com o código do Quadro 43.

Quadro 42: Definição de uma função temporal constante.

```
timeSeries("Constant", 1,'factor', 1)
```

Quadro 43: Definição de uma função temporal linear.

```
timeSeries('Linear', 1,'factor', 1)
```

Antes da definição das cargas é preciso definir o tipo de solicitação, se é simples (“plain pattern”), uma excitação uniforme (“UniformExcitation Pattern”) e excitação em vários apoios (“Multi-Support Excitation Pattern”). Essa definição é feita com o comando “pattern” que tem como argumentos o tipo de solicitação, o identificador da função temporal previamente definida e um fator de carga opcional (Quadro 44). O código do Quadro 45 mostra a definição de um tipo de carga simples, com o identificador 1, identificador da série temporal igual 1 e sem fator de carga.

Quadro 44: Definição do tipo de solicitação da carga.

```
pattern('Plain', patternTag, tsTag, '-fact', fact)
```

Quadro 45: Aplicação da definição da solicitação do tipo de caraga.

```
pattern("Plain", 1, 1)
```

Após a definição do tipo de solicitação é possível definir as cargas que ficam associadas a esse tipo. No caso de existirem cargas com diferentes funções temporais, é necessário utilizar o comando “pattern” com os argumentos correspondentes e depois definir a respetiva carga associada.

Tal como qualquer programa de cálculo estrutural, as cargas podem ser aplicadas nos nós, em barras, nas arestas ou faces de elementos finitos. O comando load define uma carga nodal e tem como argumentos o nó a que é atribuída a carga e o valor da carga de acordo com o grau de liberdade associado. O código do Quadro 46 define a carga aplicada no nó 4 e correspondente a uma carga vertical descendente de 100 kN.

Quadro 46: Definição da carga na estrutura.

```
load(4, 0.0, -100)
```

O passo seguinte corresponde à definição dos parâmetros de análise, devendo ter-se em conta que o OpenSees [30] é uma ferramenta de análise avançada de estruturas, pelo que são múltiplos os parâmetros a definir.

O comando “system()” define o modo como é construído o sistema de equações para o cálculo do sistema estrutural. Conforme [29], o OpenSees [30] contempla diferentes formas de construir esse sistema de equações que, dependendo do problema, implica maior ou menor facilidade e eficiência da análise. No exemplo em análise é considerada a variante “BandSPD”, conforme o Quadro 47.

Quadro 47: Definição do comando "system".

```
system("BandSPD")
```

Para além da indicação da forma como é construído o sistema de equações, é necessário criar um objeto relativo à numeração dos graus de liberdade do problema. Esse objeto determina como os graus de liberdade são numerados, o que tem influência na eficiência da realização da análise estrutural, em função do tipo e dimensão do problema. A variante “plain” como argumento da função numberer (Quadro 48) corresponde ao modo mais simples de numerar os graus de liberdade e adequa-se apenas para problemas simples e de pequena dimensão. Quando os problemas são complexos e de grande dimensão é necessário escolher um modo de numeração dos graus de liberdade mais sofisticado, em função das opções disponibilizadas pelo software.

Quadro 48: Definição da numeração dos graus de liberdade.

```
numberer("Plain")
```

Para além dos graus de liberdade é necessário definir como as restrições do modelo são incluídas na análise, o que se faz através do comando "constraints()", tendo como argumentos as diferentes possibilidades fornecidas pelo programa. O código do Quadro 49 descreve a definição da possibilidade "plain", aplicável nos problemas mais simples. Note-se que contrariamente a muitos softwares de análise estrutural não basta modelar os tipos de apoios, como fixo simples ou outros, sendo também necessário definir como lidar com os apoios ou outras restrições na análise. Este facto, que parece ser pouco relevante em problemas simples, é relevante em problemas complexos, como por exemplo problemas de contacto entre elementos ou outros. O sucesso da análise e a qualidade dos resultados depende da manipulação das diferentes possibilidades e parâmetros.

Quadro 49: Definição das restrições do modelo.

```
constraints("Plain")
```

Numa análise estrutural, o curso da análise pode ser controlado pelo valor das cargas, pelo valor dos deslocamentos e outros. Em problemas estruturais não lineares a análise tem que se feita por passos a partir de um valor inicial até um valor final, podendo ser controlada por cargas, por deslocamentos, com incrementos constantes ou variáveis. Há determinados problemas em que a análise deve ser controlada pelo valor das cargas aplicadas e outros em que é controlada por valor de deslocamentos em determinados nós da estrutura. Uma das possibilidades é o controlo por "Arc-Length" em que o valor do incremento de carga é ajustado ao longo da análise em função da convergência. Essa opção é normalmente necessária em problemas de análise não-linear como pode ser o caso de um problema de instabilidade por encurvadura.

Para problemas simples é suficiente a análise controlada pela carga, como exemplifica o código do Quadro 50. O primeiro argumento define o tipo de controlo, o argumento "incr" o fator de carga, "numiter" define o número de iterações por incremento, "minIncr" o valor mínimo do incremento e "maxIncr" o valor máximo do incremento. Os diferentes parâmetros só têm efeito se for escolhida uma função temporal que não seja constante.

Quadro 50: Definição de um integrador de controle de carga.

```
integrator('LoadControl', incr, numIter=1, minIncr=incr, maxIncr=incr)
```

O código do Quadro 51 mostra a definição com o controlo por cargas, com um fator de carga igual a 1.

Quadro 51: Definição com o controlo por cargas.

```
integrator("LoadControl", 1.0)
```

Antes de proceder à análise é necessário definir o tipo de análise, que pode ser estática, dinâmica, com incrementos de tempo contantes ou variáveis. Essa definição é feita pela função “analysis” em que o argumento é o tipo de análise, podendo ser “Static”, “Transient” ou “VariableTransient”. O código do Quadro 52 mostra a definição de uma análise estática.

Quadro 52: Definição do tipo de análise.

```
analysis("Static")
```

Com o código definido acima é possível proceder à análise da estrutura através do comando “analyze” conforme a descrição abaixo. O argumento “numIncr” define o número de incrementos de carga, “dt” o incremento de tempo quando a análise é dinâmica e os argumentos “dtMin” e “dtMax” as variações máximas e mínimas dos incrementos de tempo quando a análise é dinâmica com incrementos de tempo variáveis. Para uma análise estática simples basta o código descrito no Quadro 54 com o argumento relativo ao número de incrementos igual a 1.

Quadro 53: Definição do comando "analyze" para análise da estrutura.

```
analyze(numIncr=1, dt=0.0, dtMin=0.0, dtMax=0.0, Jd=0)
```

Quadro 54: Aplicação do comando "analyze".

```
analyze(1)
```

Após a análise é necessário programar o acesso aos resultados, tendo em conta que o OpenSees não tem componente gráfica para modelação da estrutura. No entanto, existem possibilidades de visualização através de diferentes ferramentas.

Um dos primeiros resultados a que é útil aceder é ao valor dos deslocamentos nodais, o que pode ser feito com o comando "nodeDisp", indicando o número do nó e o grau de liberdade correspondente ao deslocamento pretendido, conforme se exemplifica no Quadro 55, extraindo-se o deslocamento vertical do nó 4 e que é atribuído a uma variável definida como uy. Note-se, no entanto, que o comando nodeDisp não fornece o valor do deslocamento ao utilizador, apenas acede ao mesmo e permite atribuí-lo a uma variável definida na aplicação. Para ter conhecimento do valor do deslocamento podem utilizar-se as ferramentas de comunicação de uma aplicação com o utilizador como é o caso do comando "print" ou por transferência dos valores para outra aplicação como pode ser o caso do Microsoft® Excel® [19].

Quadro 55: Definição do comando "nodeDisp" para cessar ao deslocamento do nó

```
uy = nodeDisp(4,2)
print(uy)
```

O acesso ao esforço axial nas barras pode ser feito pelo comando "eleForce" que tem como argumento o número de elemento finito e o grau de liberdade pretendido, conforme mostra o código do Quadro 56.

Quadro 56: Definição do comando "eleForce" para cessar ao esforço axial.

```
NX=eleForce(1,2)
print(NX)
```

Para obter o esforço no sistema local do elemento é usado o comando "basicForce" que tem como argumento o número do elemento finito.

Quadro 57: Definição do comando "basicForce".

```
NX1=basicForce(1)
print(NX1)
```

Os códigos apresentados acima surgem completos no Quadro 58 que contém toda a rotina para modelar a estrutura, calcular e mostrar resultados.

Quadro 58: Aplicação dos códigos estudos.

```
from openseespy.opensees import *
import numpy as np
import opsvis as opsv
import matplotlib.pyplot as plt
wipe()
model('basic', '-ndm', 2, '-ndf', 2)
no_1=[0,0]
no_2=[5,0]
no_3=[10,0]
no_4=[5,5]
node(1, *no_1)
node(2, *no_2)
node(3, *no_3)
node(4, *no_4)
fix(1, 1, 1)
fix(2, 1, 1)
fix(3, 1, 1)
uniaxialMaterial("Elastic", 1, 206000.0)
nod_el1=[1,4]
a_el1=0.05
el1_prop=[a_el1,1]
element("Truss",1,*nod_el1,*el1_prop)
element("Truss",2,2,4,0.05,1)
element("Truss",3,3,4,0.05,1)
timeSeries("Linear", 1,'factor',1)
pattern("Plain", 1, 1)
load(4,0.0,-100)
system("BandSPD")
numberer("Plain")
constraints("Plain")
integrator("LoadControl",1)
algorithm("Linear")
analysis("Static")
analyze(1)
ux = nodeDisp(4,1)
uy = nodeDisp(4,2)
NX=eleForce(1,2)
NX1=basicForce(1)
print(uy)
print(NX)
print(NX1)
```

Conforme [29] estão desenvolvidas para o OpenSeesPy [29] funções de visualização gráfica que permitem aceder à geometria da estrutura, disponíveis na biblioteca opsvis.

A sua utilização é possível se essa biblioteca for importada, através da função import, no início do código, conforme se descreve no Quadro 59.

Quadro 59: Definição da biblioteca "opsvis".

```
import opsvis as opsv
import matplotlib.pyplot as plt
```

O Quadro 60 define a representação na deformada da estrutura e os diagramas de esforços axiais.

Quadro 60: Aplicação da biblioteca "opsvis" no caso em análise.

```
opsv.plot_defo()
sfacN, sfacV, sfacM = 5.e-5, 5.e-5, 5.e-5
opsv.section_force_diagram_2d('N', sfacN)
plt.title('Axial force distribution')
```

Assim adicionado os códigos dos Quadros 59 e 60 ao Quadro 58, obtem-se o bloco de código do Quadro 61, que modela a estrutura, faz a sua análise e apresenta os resultados.

Quadro 61: Aplicação dos códigos, visualização da estrutura e resultados.

```
from openseespy.opensees import *
import numpy as np
import opsvis as opsv
import matplotlib.pyplot as plt
wipe()
model('basic', '-ndm', 2, '-ndf', 2)
no_1=[0,0]
no_2=[5,0]
no_3=[10,0]
no_4=[5,5]
node(1, *no_1)
node(2, *no_2)
node(3, *no_3)
node(4, *no_4)
fix(1, 1, 1)
fix(2, 1, 1)
fix(3, 1, 1)
uniaxialMaterial("Elastic", 1, 206000.0)
nod_el1=[1,4]
a_el1=0.05
el1_prop=[a_el1,1]
```

```
element("Truss",1,*nod_el1,*el1_prop)
element("Truss",2,2,4,0.05,1)
element("Truss",3,3,4,0.05,1)
timeSeries("Linear", 1,'factor',1)
pattern("Plain", 1, 1)
load(4,0.0,-100)
system("BandSPD")
numberer("Plain")
constraints("Plain")
integrator("LoadControl",1)
algorithm("Linear")
analysis("Static")
analyze(1)
ux = nodeDisp(4,1)
uy = nodeDisp(4,2)
NX=eleForce(1,2)
NX1=basicForce(1)
print(uy)
print(NX)
print(NX1)
import opsvis as opsv
import matplotlib.pyplot as plt
opsv.plot_defo()
sfacN, sfacV, sfacM = 5.e-5, 5.e-5, 5.e-5
opsv.section_force_diagram_2d('N', sfacN)
plt.title('Axial force distribution')
```

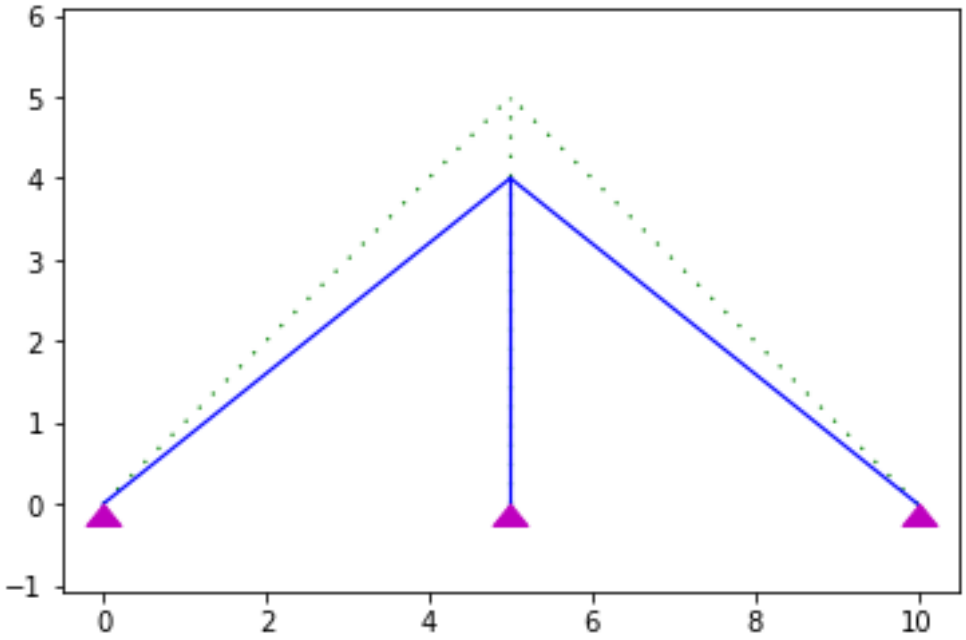


Figura 3: Representação do diagrama da deformada da estrutura adaptado de [29].

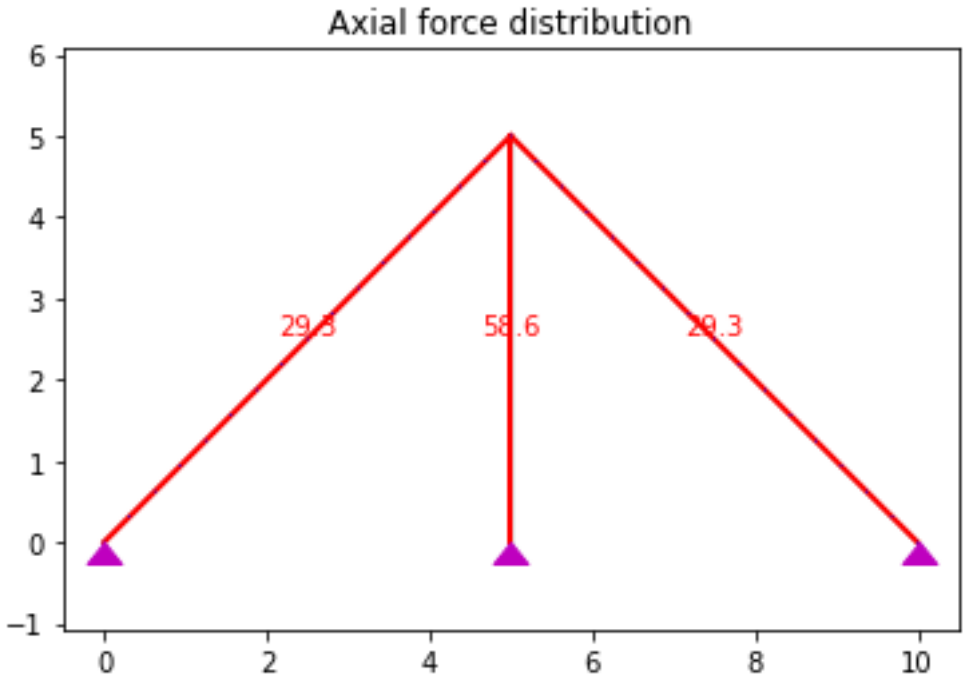


Figura 4: Representação do diagrama do esforço axial da estrutura adaptado de [29].

Capítulo 3

Aplicações do API do Robot

3.1. Descrição

Na presente secção são apresentadas diferentes aplicações da API do RSA com diferentes graus de complexidade. A um nível mais básico são apresentadas aplicações que descrevem a utilização da API na modelação, análise e tratamento de resultados de estruturas simples. Numa perspetiva de análise mais avançada de estruturas apresenta-se a modelação e análise de estruturas com ligações semirrígidas e a modelação e análise de uma estrutura de ponte sujeita a uma carga dinâmica, correspondente a um comboio de alta velocidade. Enquanto no primeiro caso simples as operações podiam ser facilmente realizadas diretamente no RSA, nas aplicações mais complexas existem claras vantagens em recorrer à API.

3.2. Pórtico plano isostático

3.2.1. Aplicação 1: Modelação da geometria do pórtico

A primeira aplicação corresponde à modelação de um pórtico plano isostático (Figura 5). A descrição da aplicação é estruturada em blocos de código desde a modelação dos nós e das barras. Os blocos de código apresentados aparecem sequencialmente.

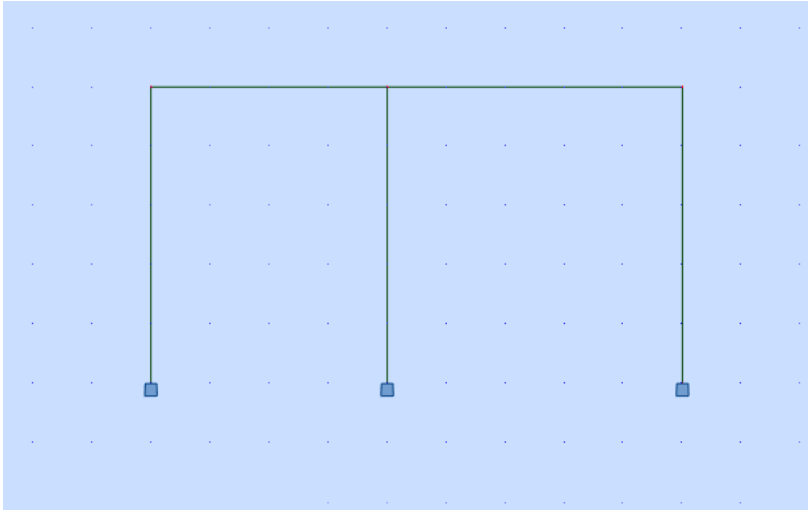


Figura 5: Modelação de um pórtico simples.

A definição dos nós e os restantes procedimentos é incluída numa nova aplicação definida como robot_ap1, conforme a segunda linha de código. Os diferentes

procedimentos surgem associados a aplicação robot_ap1 desde a criação dos nós, das barras, entre outras. Na aplicação robot_ap1 está incluído um projeto de estrutura na qual se criam nós com as coordenadas previamente definidas pelo utilizador, conforme as linhas de código 18 a 23 do Quadro 62, em que surge nas linhas de código robot_ap1.Project.Structure [25].

Quadro 62: Criação dos nós da estrutura.

```
Sub ap_1()
'Define robot_ap1 como uma nova aplicação do programa Robot
Dim robot_ap1 As New RobotApplication
'Coordenadas dos nós do pórtico
AX = 0
AY = 0
BX = 0
BY = 5
CX = 4
CY = 0
DX = 4
DY = 5
EX = 9
Ey = 0
FX = 9
FY = 5
robot_ap1.Project.Structure.Nodes.Create 1, AX, 0, AY
robot_ap1.Project.Structure.Nodes.Create 2, BX, 0, BY
robot_ap1.Project.Structure.Nodes.Create 3, CX, 0, CY
robot_ap1.Project.Structure.Nodes.Create 4, DX, 0, DY
robot_ap1.Project.Structure.Nodes.Create 5, EX, 0, Ey
robot_ap1.Project.Structure.Nodes.Create 6, FX, 0, FY
```

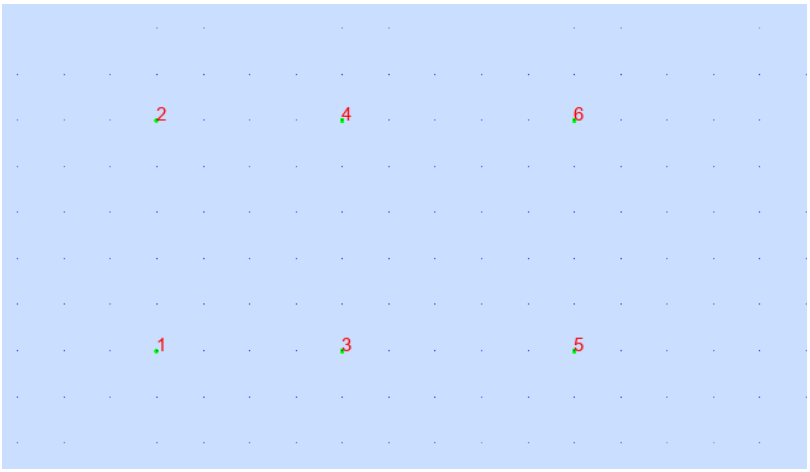


Figura 6: Representação dos nós após a leitura do quadro 62.

A partir dos nós, é possível definir barras com linhas de código similares às que constam acima para a definição dos nós. Na linha de código adicionam-se barras criadas com os nós previamente definidos (Quadro 63).

Quadro 63: Criação das barras.

```
robot_ap1.Project.Structure.Bars.Create 1, 1, 2
robot_ap1.Project.Structure.Bars.Create 2, 2, 4
robot_ap1.Project.Structure.Bars.Create 3, 3, 4
robot_ap1.Project.Structure.Bars.Create 4, 4, 6
robot_ap1.Project.Structure.Bars.Create 5, 5, 6
```

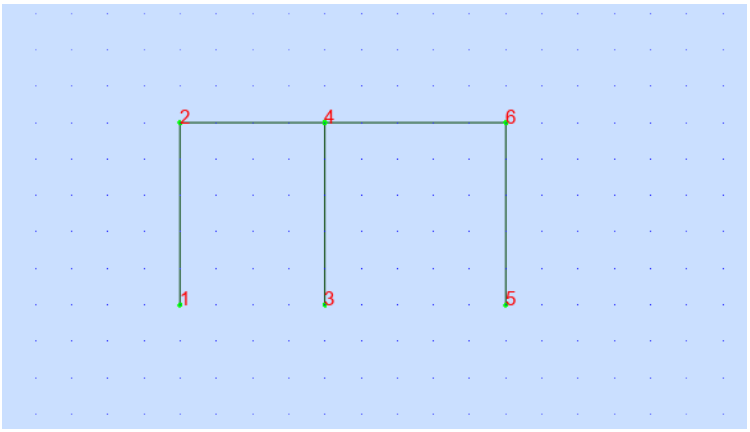


Figura 7: Figura 2: Representação dos nós e das barras, após a leitura do quadro 63.

Nas aplicações anteriores são definidos apenas os nós e as barras da estrutura. Seguindo o procedimento de modelação estrutural típico, torna-se necessário exemplificar a modelação das propriedades geométricas e mecânicas das barras, assim como dos apoios e das cargas. Tal como na utilização normal do RSA, torna-se necessário definir o tipo de

problema estrutural a modelar e analisar, nomeadamente, se se trata de um pórtico bidimensional, tridimensional ou outro tipo de estrutura. Esse tipo de definição é feito na linha de código 3, correspondente a robot_ap2.Project.New (Quadro 64) seguida do tipo de problema [25]. No caso de um pórtico bidimensional usa-se o identificador pré-definido na API I_PT_FRAME_2D [25].

A modelação automática das propriedades das barras implica a definição prévia de servidores de atributos, conforme a designação constante em [25]. No bloco de código do Quadro 64 são definidos servidores de atributos para as propriedades geométricas das secções das barras, para os materiais, para os apoios e para os casos de carga. Para além dos servidos de atributos, torna-se necessário definir os objetos das diferentes classes incorporadas na API do RSA. No caso, são definidos objetos da classe RobotBarSectionData [25] para as propriedades das barras, para o tipo de material, para os apoios e para os casos de carga.

Quadro 64: Definição dos códigos para atribuição das propriedades mecânicas das barras.

```
Sub ap_2()
'Define robot_ap2 como uma nova aplicação do programa Robot
Dim robot_ap2 As New RobotApplication
'Definição de um novo projeto de pórtico plano
'robot_ap2.Project.New I_PT_FRAME_2D
'Definição de um servidor de atributos de barras
'Dim Labelbars As IRobotLabel
'Criação de um objeto da classe RobotBarSectionData com as propriedades das barras
'Dim bar_sec As RobotBarSectionData
'Definição de um servidor de atributos de material
'Dim LabelMaterial As RobotLabel
'Criação de um objeto da classe RobotMaterialData com os dados do material
'Dim mat_prop As RobotMaterialData
'Definição de um servidor para apoios
'Dim LabelApoios As RobotLabel
'Definição de um objeto da classe RobotNodeSupportData com os dados dos apoios
'Dim apoios_prop As RobotNodeSupportData
'Criação de um objeto correspondente a um caso de carga simples para carga permanente
'Dim CASE_Permanente As RobotSimpleCase
'Criação de uma variável que recebe os dados de cargas associados a um determinado caso de carga
'Dim LoadREC_perm As RobotLoadRecord
'Criação de um objeto correspondente a um caso de carga simples para carga variável
'Dim CASE_Live As RobotSimpleCase
'Dim LoadREC_live As RobotLoadRecord
```

Após a definição dos servidores de atributos de barras e dos objetos das diferentes classes, é possível definir as propriedades das barras, dos materiais e dos apoios. O Quadro 65 define e atribui as propriedades às barras da estrutura. Os objetos previamente definidos são relacionados com os servidores de atributos para cada um dos tipos de propriedades. Por exemplo o código bar_sec.SetValue [25] atribui uma propriedade à barra, em que a identificação do tipo de propriedade e o valor correspondente são colocados de seguida a esse código. A identificação de cada um dos tipos de propriedades é feita com identificadores previamente definidos na API do RSA. A definição da área da secção é feita pelo identificador I_BSDV_AX [25], seguida do valor associado.

Quadro 65: Definição das propriedades mecânicas das barras (áreas e inércia).

```
'Definição das propriedades mecânicas das barras, área, inércia
Set Labelbars = robot_ap2.Project.Structure.Labels.Create(I_LT_BAR_SECTION, "section")
'Instrução que define a atribuição das propriedades das barras ao objeto bar_sec
Set bar_sec = Labelbars.Data
'Definição da área de secção da barra igual a 100 cm2
bar_sec.SetValue I_BSDV_AX, 0.01
'Definição dos momentos de inércia iguais a 5000 cm4
bar_sec.SetValue I_BSDV_IX, 0.00005
bar_sec.SetValue I_BSDV_IY, 0.00005
bar_sec.SetValue I_BSDV_IZ, 0.00005
bar_sec.SetValue I_BSDV_IY, 0.00005
bar_sec.SetValue I_BSDV_IZ, 0.00005
'O servidor Labelbars é guardado na aplicação em definição
robot_ap2.Project.Structure.Labels.Store Labelbars
'Atribuição da secção definida às barras que previamente foram criadas no modelo
robot_ap2.Project.Structure.Bars.Get(1).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(3).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(4).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(5).SetLabel I_LT_BAR_SECTION, "section"
```

Figura 8: Quadro capturado no RSA, que representa a definição e atribuição das propriedades da secção transversal.

A definição e atribuição dos materiais estruturais é feita de modo similar ao que foi apresentado acima para as secções transversais, tendo por base o servidor LabelMaterial [25] e o objeto mat_prop [25] definidos previamente (Quadro 66).

Quadro 66: Definição do material das barras.

```
'Definição do material a atribuir às barras
Set LabelMaterial = robot_ap2.Project.Structure.Labels.Create(I_LT_MATERIAL, "material")
'Instrução que define a atribuição das propriedades do material ao objeto mat_prop
Set mat_prop = LabelMaterial.Data
'Definição do tipo de material, no caso betão
mat_prop.Type = I_MT_CONCRETE
'Definição do módulo de elasticidade
'Para introduzir 30000 MPa no Robot, no código deve introduzir-se 30000000000
mat_prop.E = 30000000000#
'Definição do coeficiente de Poisson
mat_prop.NU = 1 / 6
'Definição do peso volumico
mat_prop.RO = 25000
'Definição do módulo de distorção
mat_prop.Kirchoff = mat_prop.E / (2 * (1 + mat_prop.NU))
'O servidor LabelsMaterial é guardado na aplicação em definição
robot_ap2.Project.Structure.Labels.Store LabelMaterial
'O material criado é atribuído às barras previamente modeladas
robot_ap2.Project.Structure.Bars.Get(1).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(2).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(3).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(4).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(5).SetLabel I_LT_MATERIAL, "material"
```

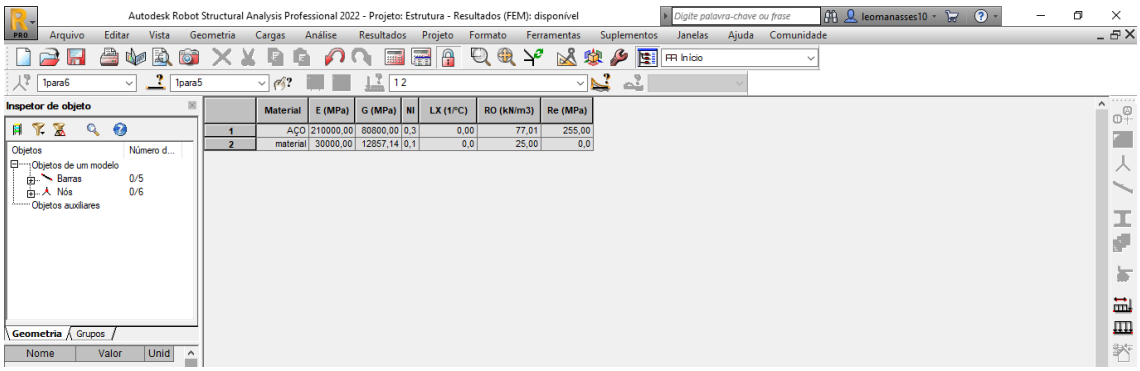


Figura 9: Quadro capturado no RSA, que representa a definição e atribuição dos materiais estruturais.

A definição e atribuição dos apoios nos nós é baseada no servidor LabelApoios [25], e no objeto apoios_prop [25], realizando-se de modo similar ao apresentado acima para as

secções transversais das barras e respetivos materiais. O Quadro 67 descreve a definição e atribuição de apoios fixos a dois nós da estrutura.

Quadro 67: Definição e atribuição dos apoios a nós.

```
'Definição de um tipo de apoio
Set LabelApoios = robot_ap2.Project.Structure.Labels.Create(I_LT_SUPPORT, "apoio_def")
Set apoios_prop = LabelApoios.Data
'Os valores abaixo permitem definir um apoio encastrado
apoios_prop.UX = 1
apoios_prop.UY = 1
apoios_prop.UZ = 1
apoios_prop.RX = 1
apoios_prop.RY = 1
apoios_prop.RZ = 1
robot_ap2.Project.Structure.Labels.Store LabelApoios
'O apoio definido é atribuído aos nós 1,3 e 5
robot_ap2.Project.Structure.Nodes.Get(1).SetLabel I_LT_SUPPORT, "apoio_def"
robot_ap2.Project.Structure.Nodes.Get(3).SetLabel I_LT_SUPPORT, "apoio_def"
robot_ap2.Project.Structure.Nodes.Get(5).SetLabel I_LT_SUPPORT, "apoio_def"
```

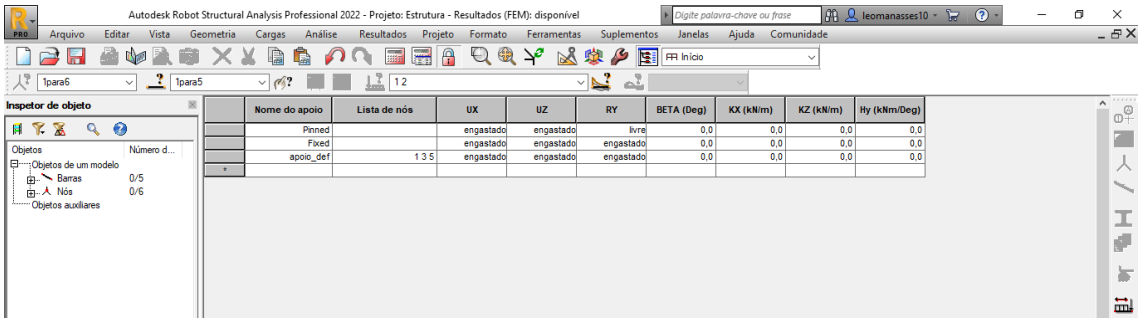


Figura 10: Quadro capturado no RSA, que representa a definição e atribuição de apoios a nós.

A definição de casos de carga e cargas implica a definição prévia de casos de carga e de variáveis que recebem os dados relativos aos casos de carga. O Quadro 68 define um caso de carga permanente e uma sobrecarga variável.

O caso de carga permanente é atribuído a todas as barras, sendo definido com base nos objetos CASE_Permanente e LoadREC_perm [25]. No caso dos casos de carga podem ser criados casos simples ou cados complexos.

O caso de carga permanente corresponde ao objeto CASE_Permanente, através do código robot_ap2.Project.Structure.Cases.CreateSimple, que funciona como uma função com argumentos correspondentes ao número identificador do caso, a designação do caso e dois argumentos identificadores da API do RSA que indicam o caso como permanente

para uma análise estática linear, por I_CN_PERMANENT [25] e I_CAT_STATIC_LINEAR [25]. No caso pretende-se que o caso de carga seja correspondente ao peso próprio da estrutura, o que é feito pelo código CASE_Permanente.Records.New [25] em que o identificador I_LRT_DEAD [25] surge associado na API do RSA a cargas devidas ao peso próprio. O caso de carga é guardado na aplicação e atribuído ao objeto LoadREC_perm [25], que assume o caso de carga identificado com o número 1. A direção e o sentido da carga são definidas por LoadREC_perm.SetValue I_DRV_z,-1 [25] em que se define a direção do eixo Z no sentido descendente. Finalmente, indica-se que a carga é aplicada a todos os elementos da estrutura. O Quadro 68 descreve uma rotina que faz a definição e atribuição dos casos de carga conforme referido acima.

Quadro 68: Definição de caso de carga e atribuição de cargas à barra.

```
'Definição de um caso de carga permanente para análise linear com o nome Caso 1
Set CASE_Permanente = robot_ap2.Project.Structure.Cases.CreateSimple(1, "Caso 1", I_CN_PERMANENT, I_CAT_STATIC_LINEAR)
'Definição no programa de cálculo da carga do tipo dead load que corresponde ao peso próprio da estrutura
CASE_Permanente.Records.New I_LRT_DEAD
'Atribuição dos dados do caso de carga permanente ao objeto LoadREC_perm
Set LoadREC_perm = CASE_Permanente.Records.Get(1)
'Definição da direção e sentido da carga permente
LoadREC_perm.SetValue I_DRV_Z, -1
'Atribuição do caso de carga a todos os elementos da estrutura
LoadREC_perm.SetValue I_DRV_ENTIRE_STRUCTURE, True
```

A definição de uma sobrecarga variável é feita de forma similar, conforme descreve o bloco de código do Quadro 69. No caso é definida uma carga uniformemente distribuída aplicada numa barra e uma carga concentrada aplicada num nó. As duas cargas correspondem ao mesmo caso de carga, pelo que é comum o objeto LoadREC_live [25] associado. A definição de uma carga concentrada é feita de forma similar, conforme o Quadro 70.

Quadro 69: Definição de uma sobrecarga variável.

```
'Definição de uma sobrecarga variável
Set CASE_Live = robot_ap2.Project.Structure.Cases.CreateSimple(2, "Live", I_CN_EXPLOATATION, I_CAT_STATIC_LINEAR)
'Definição do tipo de carga uniformente distribuída associado ao caso de carga e atribuído à variável uniform
Uniform = CASE_Live.Records.New(I_LRT_BAR_UNIFORM)
'Atribuição ao objeto LoadREC_live da propriedade de carga uniforme em função
'da qual serão definidos os dados correspondentes
Set LoadREC_live = CASE_Live.Records.Get(Uniform)
'Definição do valor da carga uniforme na direção X
LoadREC_live.SetValue I_URV_PX, 0
'Definição do valor da carga uniforme na direção Y
LoadREC_live.SetValue I_URV_PZ, -10000
'Atribuição da carga à barra 2
LoadREC_live.Objects.FromText ("2")
```

Quadro 70: Definição do tipo de carga concentrada.

```
'Definição do tipo de carga concentrada associado ao caso de carga e atribuído à variável Concentrated
Concentrated = CASE_Live.Records.New(I_LRT_NODE_FORCE)
'Atribuição ao objeto LoadREC_live da propriedade de carga uniforme em função
'da qual serão definidos os dados correspondentes
Set LoadRECLIVE = CASE_Live.Records.Get(Concentrated)
'Definição do valor da carga concentratada na direção X
LoadRECLIVE.SetValue I_URV_PX, 10000
'Atribuição da carga ao nó 2
LoadRECLIVE.Objects.FromText ("2")
```

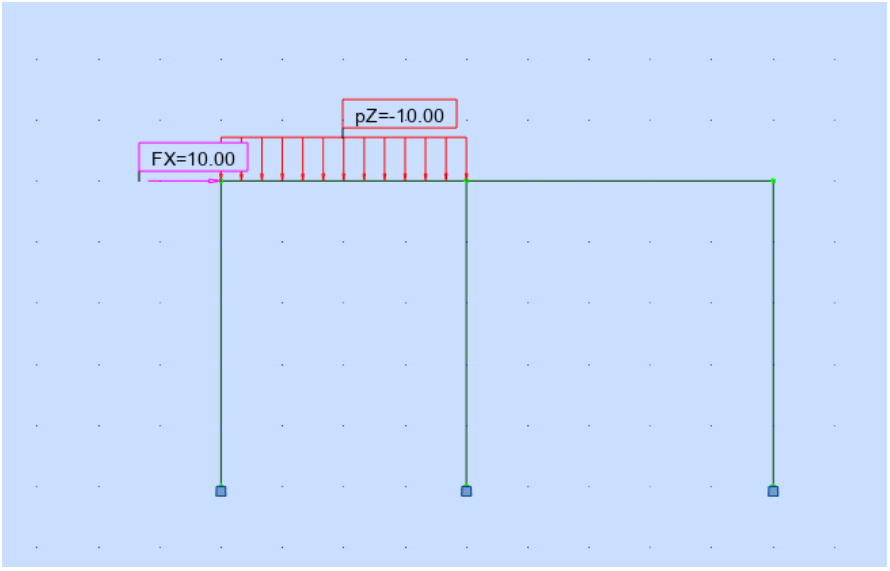


Figura 11: Representação do pórtico com a estrutura e atribuição de casos de cargas à barra gerados a partir do código que consta no 1 do Anexo B - Aplicação do API do RS.

O código do Quadro 71 define de modo automático a realização da análise da estrutura. Esse tipo de opção pode ser útil para cálculos repetidos com alteração sucessiva de parâmetros.

Quadro 71: Definição do comando para execução da análise.

```
'Comando para execução da análise
robot_ap2.Project.CalcEngine.Calculate
```

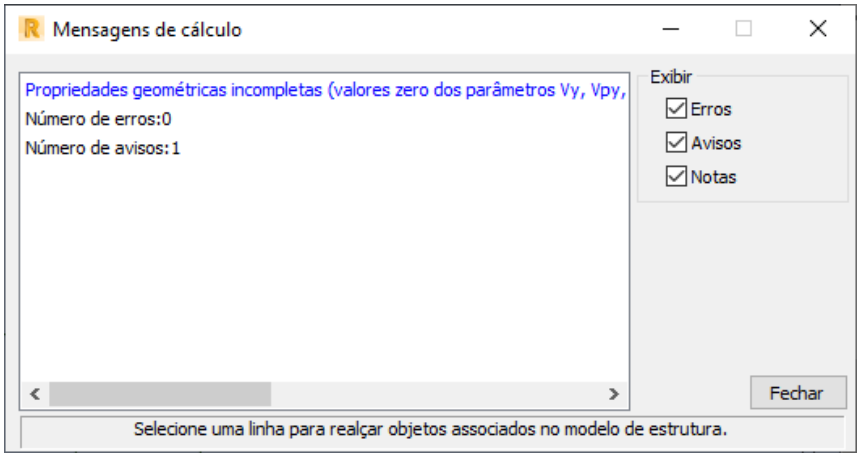


Figura 12: Representação da definição e execução da análise.

No anexo B - Aplicação do API do RSA, está a representação completa dos blocos de código citados.

3.3. Pós-processamento de resultados

Na análise de estruturas a partir de aplicações externas é particularmente relevante a possibilidade de extrair e utilizar os resultados em função das necessidades do utilizador. As aplicações para obtenção de resultados podem ser criadas de modo isolado para serem utilizadas sobre modelos estruturais existentes, independentemente do modo como são criadas. A obtenção dos resultados implica o acesso prévio aos elementos da estrutura, desde barras, nós, painéis, etc. No caso da API do RSA, esse acesso é possível através de estruturas de dados definidas como IRobotCollection [25], que permite o acesso a objetos das classes correspondentes às barras, nós, entre outros.
Os dados correspondentes às barras são acedidos através de objetos da classe IRobotBar [25] e os dados correspondentes aos nós são acedidos através de objetos da classe IRobotNode [25].

Em relação aos resultados são utilizadas variáveis de acesso a dados correspondentes a esforços nas barras e a dados correspondentes à deslocamentos nos nós. O bloco de código do Quadro 72 mostra a declaração das estruturas de dados e dos objetos.

Quadro 72: Definição dos códigos para o acesso dos valores dos esforços da estrutura.

```
Sub Read_results()
Dim robot_results As New RobotApplication
'Definição de uma estrutura de dados para os nós da estrutura
'Dim grupo_nos As IRobotCollection
'Definição de uma estrutura de dados para as barras da estrutura
'Dim grupo_barras As IRobotCollection
'Definição de um objeto da classe RobotNode
'Dim node As IRobotNode
'Definição de um objeto da classe RobotBar
'Dim bar As IRobotBar
'Definição de variável para esforços nas barras
'Dim force_bar As RobotBarForceData
'Definição de variável para deslocamentos nas barras
'Dim no_displacement As RobotNodeDisplacementData
'Acesso a todos os nós de uma estrutura
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n2 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n3 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n4 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
```

```
Dim deslc_X_n5 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n6 As Double
'Definição de uma variável para receber o valor do momento MY numa barra
Dim momen_y_b1 As Double
Dim momen_y_b2 As Double
Dim momen_y_b3 As Double
Dim momen_y_b4 As Double
Dim momen_y_b5 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_x_b1 As Double
Dim esf_z_b1 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b2 As Double
Dim esf_x_b2 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b3 As Double
Dim esf_x_b3 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b4 As Double
Dim esf_x_b4 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b5 As Double
Dim esf_x_b5 As Double
```

A obtenção dos dados dos nós é feita acedendo à estrutura de dados referente aos nós, usando como argumento de entrada o número identificador do nó. Assim, será possível aceder às informações dos nós, como o número e as coordenadas, conforme se exemplifica no bloco de código do Quadro 73.

Quadro 73: Descrição de acesso a todos os nós da estrutura.

```
'Acesso a todos os nós da estrutura atribuído à estrutura de dados grupo_nos
Set grupo_nos = robot_results.Project.Structure.Nodes.GetAll
'Atribuição ao objeto node das informações do nó 1
'Atribuição ao objeto node das informações do nó 2
'Atribuição ao objeto node das informações do nó 3
'Atribuição ao objeto node das informações do nó 4
'Atribuição ao objeto node das informações do nó 5
'Atribuição ao objeto node das informações do nó 6
'Para acesso de atributos de outros nós basta mudar o argumento do nó
'corresponde ao nó
Set node = grupo_nos.Get(1)
Set node = grupo_nos.Get(2)
Set node = grupo_nos.Get(3)
```

```
Set node = grupo_nos.Get(4)
Set node = grupo_nos.Get(5)
Set node = grupo_nos.Get(6)
'Atribuição à variável NN do número do nó
NN = node.Number
'Atribuição à variável X da abcissa do nó
X = node.X
Y = node.Y
Z = node.Z
```

O acesso a dados das barras é feito de modo similar ao descrito para os nós, conforme exemplifica o Quadro 74. O acesso aos diferentes dados de uma barra é feito através de objetos da classe barra, através do número da barra pretendida.

Quadro 74: Acesso as barras da estrutura.

```
'Acesso a todas as barras da estrutura
Set grupo_barras = robot_results.Project.Structure.Bars.GetAll
'Atribuição ao objeto bar das informações da barra 1
'Atribuição ao objeto bar das informações da barra 2
'Atribuição ao objeto bar das informações da barra 3
'Atribuição ao objeto bar das informações da barra 4
'Atribuição ao objeto bar das informações da barra 5
Set bar = grupo_barras.Get(1)
Set bar = grupo_barras.Get(2)
Set bar = grupo_barras.Get(3)
Set bar = grupo_barras.Get(4)
Set bar = grupo_barras.Get(5)
xk = bar.EndNode
'Atribuição à varial NA do número da barra 1
NA = bar.Number
LA = bar.Length
'Atribuição à varial NB do número da barra 2
NB = bar.Number
LB = bar.Length
'Atribuição à varial NC do número da barra 3
Nc = bar.Number
LC = bar.Length
'Atribuição à varial ND do número da barra 4
ND = bar.Number
LD = bar.Length
'Atribuição à varial NE do número da barra 5
NE = bar.Number
LE = bar.Length
```

O acesso a deslocamentos dos nós é feito através de um objeto da classe RobotNodeDisplacementData [25] com argumentos correspondentes ao caso de carga e ao número do nó pretendido. O Quadro 75 descreve o acesso aos deslocamentos dos nós para o caso de carga com o número 2.

Quadro 75. Acesso aos deslocamentos dos nós da estrutura.

```
'Acesso aos deslocamentos do nó 2 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 3 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 4 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 5 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 6 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'A variável deslc_X_n2 recebe o deslocamento horizontal do nó 2
deslc_X_n2 = no_displacement.UX
Folha1.Cells(1, 1) = deslc_X_n2
'Acesso aos deslocamentos do nó 3 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'A variável deslc_X_n3 recebe o deslocamento horizontal do nó 3
deslc_X_n3 = no_displacement.UX
Folha1.Cells(1, 2) = deslc_X_n3
'A variável deslc_X_n4 recebe o deslocamento horizontal do nó 4
deslc_X_n4 = no_displacement.UX
Folha1.Cells(1, 3) = deslc_X_n4
'A variável deslc_X_n5 recebe o deslocamento horizontal do nó 5
deslc_X_n5 = no_displacement.UX
Folha1.Cells(1, 4) = deslc_X_n5
'A variável deslc_X_n6 recebe o deslocamento horizontal do nó 6
deslc_X_n6 = no_displacement.UX
Folha1.Cells(1, 5) = deslc_X_n6
```

O acesso aos esforços nas barras é feito de forma similar, recorrendo a objetos da classe RobotBarForceData [25] com argumentos correspondentes ao número de barra, à secção pretendida e ao caso de carga. O Quadro 76 descreve a obtenção do valor do momento fletor, esforço axial e o esforço transverso para a secção final de uma carga, devido a um caso de carga.

Quadro 76: Acesso aos esforços das barras.

```
'Acesso aos esforços da barra 1, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(1, 2, 1)
'A variável momen_y_b1 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b1 = force_bar.MY
Folha1.Cells(3, 1) = momen_y_b1
'A variável esf_x_b1 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b1 = force_bar.FX
Folha1.Cells(4, 1) = esf_x_b1
'A variável esf_Z_b1 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_z_b1 = force_bar.FZ
Folha1.Cells(5, 1) = esf_z_b1
'Acesso aos esforços da barra 2, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(2, 2, 1)
'A variávelmomen_y_b2 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b2 = force_bar.MY
Folha1.Cells(3, 2) = momen_y_b2
'A variável esf_x_b2 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b2 = force_bar.FX
Folha1.Cells(4, 2) = esf_x_b2
'A variável esf_Z_b2 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b2 = force_bar.FZ
Folha1.Cells(5, 2) = esf_Z_b2
'Acesso aos esforços da barra 3, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(3, 2, 1)
'A variável momen_y_b3 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b3 = force_bar.MY
Folha1.Cells(3, 3) = momen_y_b3
'A variável esf_x_b3 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b3 = force_bar.FX
Folha1.Cells(4, 3) = esf_x_b3
'A variável esf_Z_b3 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b3 = force_bar.FZ
Folha1.Cells(5, 3) = esf_Z_b3
'Acesso aos esforços da barra 4, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(4, 2, 1)
'A variável momen_y_b4 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b4 = force_bar.MY
Folha1.Cells(3, 4) = momen_y_b4
'A variável esf_x_b4 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b4 = force_bar.FX
Folha1.Cells(4, 4) = esf_x_b4
'A variável esf_Z_b4 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b4 = force_bar.FZ
Folha1.Cells(5, 4) = esf_Z_b4
'Acesso aos esforços da barra 5, para o caso de carga 2, na secção 1
```

```
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(5, 2, 1)
'A variável momen_y_b5 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b5 = force_bar.MY
Folha1.Cells(3, 5) = momen_y_b5
'A variável esf_x_b5 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b5 = force_bar.FX
Folha1.Cells(4, 5) = esf_x_b5
'A variável esf_Z_b5 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b5 = force_bar.FZ
Folha1.Cells(5, 5) = esf_Z_b5
```

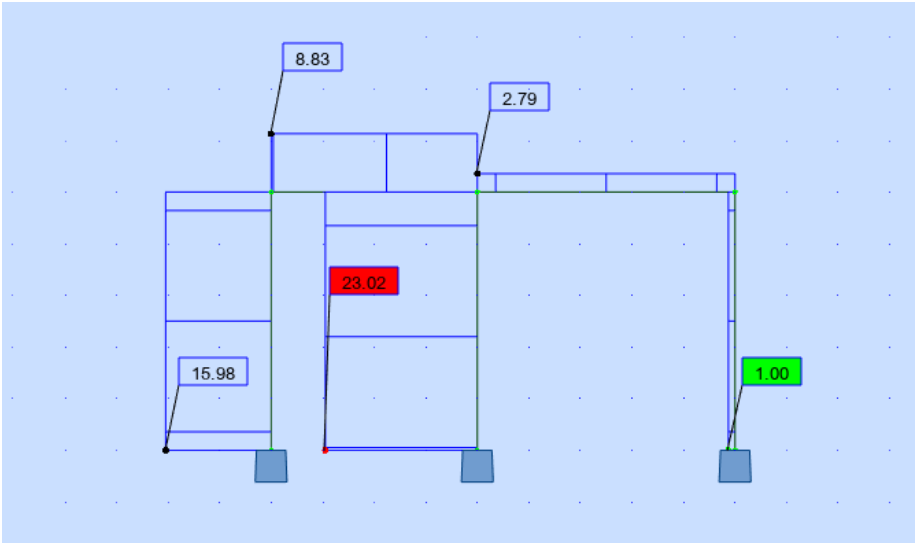


Figura 13: Representação do esforço axial nas barras com a estrutura e análise gerados a partir do código que consta no ponto 1 do Anexo B - Aplicação do API do RSA.

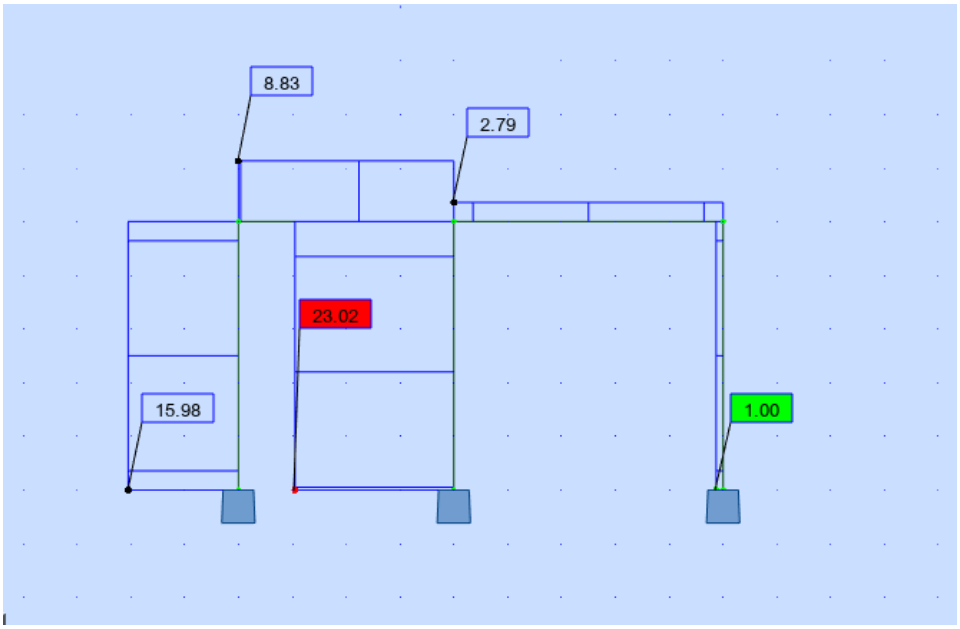


Figura 14: Representação do esforço transversa nas barras com a estrutura e análise gerados a partir do código que consta no ponto 1 do Anexo B - Aplicação do API do RSA.

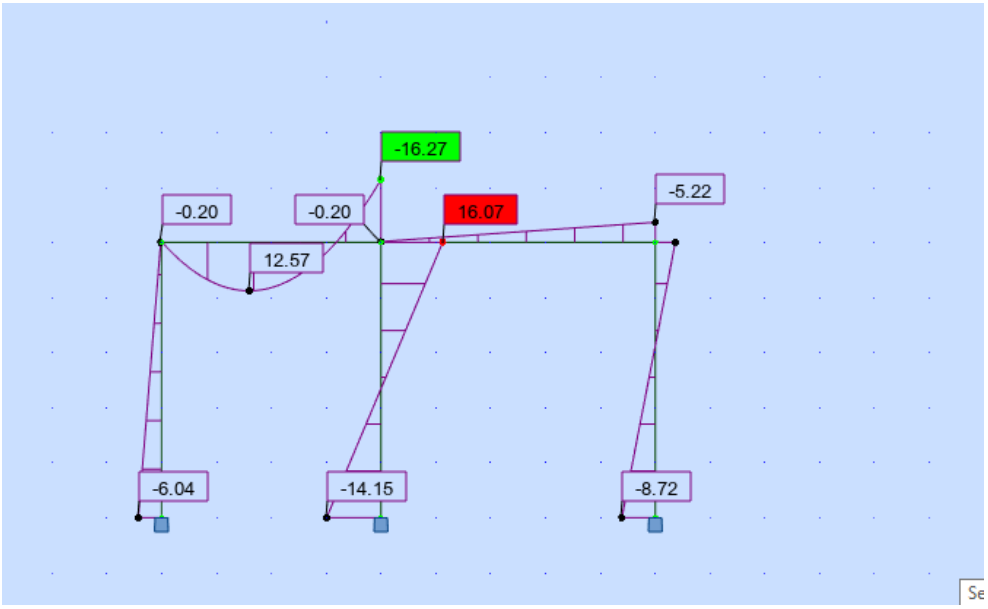


Figura 15: Representação do momento nas barras com a estrutura e análise gerados a partir do código que consta no ponto 1 do Anexo B - Aplicação do API do RSA.

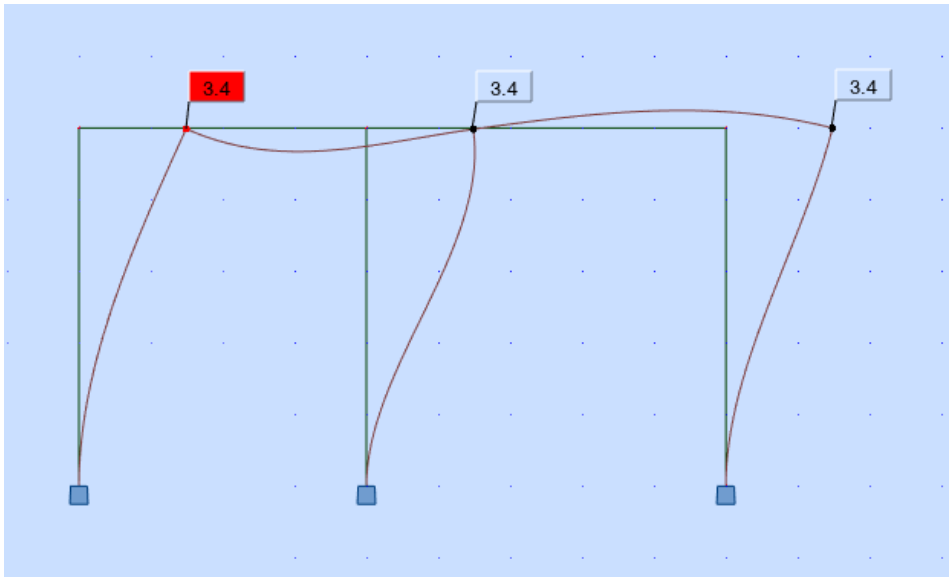


Figura 16: Representação do deslocamento das barras com a estrutura e análise gerados a partir do código que consta no ponto 1 do Anexo B - Aplicação do API do RSA.

3.4. Modelação de estruturas com ligações articuladas ou semirrígidas

3.4.1. Descrição da análise

A análise de estruturas com ligações articuladas e semirrígidas é importante porque as ligações têm um efeito significativo no comportamento global da estrutura. Em estruturas com ligações articuladas, os membros podem-se mover livremente, o que significa que as cargas são transmitidas de maneira mais simples e direta. Isso facilita a análise estrutural, pois a estrutura pode ser tratada como uma série de elementos independentes conectados por ligações articuladas. Por outro lado, em estruturas com ligações semirrígidas, os membros são conectados de tal maneira que oferecem alguma resistência ao movimento, o que significa que as tensões e cargas são distribuídas de forma mais complexa e indireta. Isso torna a análise estrutural mais complicada, pois é necessário levar em consideração a influência das ligações semirrígidas na distribuição de cargas e tensões em toda a estrutura. Nesta secção descreve-se a modelação e análise de estruturas com ligações semirrígidas através de uma API.

3.4.2. Modelação de ligações articuladas

A modelação de ligações entre barras do sistema estrutural implica a definição de um servidor de atributos para criar libertações de barras como o RobotLabel [25]. Posteriormente é criado um "grupo" de dados que recebe as proriedades da ligação como RobotBarReleaseData [25]. O Quadro 77 descreve essas definições.

Quadro 77: Definição de um servidor de atributos de libertações nas barras.

```
Dim release_label As RobotLabel
'Criação de um grupo de dados do tipo RobotBarReleaseData com as propriedades
'das libertações dos nós
Dim release_bar As RobotBarReleaseData
```

O Quadro 78, mostra a definição de um esquema de libertação que pode ser atribuído a uma ou mais barras. Atribuição de um esquema de libertações a cada barra tem de ser sempre efetuada para as duas extremidades das barras. No caso de uma das extremidades ser contínua torna-se necessário definir e atribuir uma articulação do tipo I_BERV_NONE [25]. O Quadro 78 define um esquema de articulação para rotações, articulado num dos nós e semirrígido no outro. Note-se que na extremidade semirrígida é necessária definição sucessiva de uma ligação articulada e depois da ligação semirrígida a que é necessário atribuir o valor da rigidez de rotação. Os códigos RY e HY definem libertação e rigidez para rotações em torno do eixo y. No caso de se pretender libertar rotações e deslocamentos noutras direções, usam-se diferentes códigos [25].

Quadro 78: Definição das propriedades de uma libertação

```
'Definiçaão das propriedades de uma libertação de barra com o nome CR_Release
Set release_label = robot_ap1.Project.Structure.Labels.Create(I_LT_BAR_RELEASE, "CR_Release")
'Definição do objeto release_bar que recebe os dados da libertação
Set release_bar = release_label.Data
'Indicação de que a ligação no nó de extremidade inicial é articulada
release_bar.StartNode.RY = I_BERV_STD
'Indicação do tipo de articulação do nó de extremidade final é articulada
'Esta definição tem que ser previamente definido antes dos tipos de ligações mais avançados
'como é o caso da ligação elástica
release_bar.EndNode.RY = I_BERV_STD
'Definição de um tipo de ligação elástica
release_bar.EndNode.RY = I_BERV_ELASTIC
'Definição do valor da rigidez da ligação
release_bar.EndNode.HY = 10000
'Atribuição da libertação à barra 2
robot_ap1.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_RELEASE, "CR_Release"
```

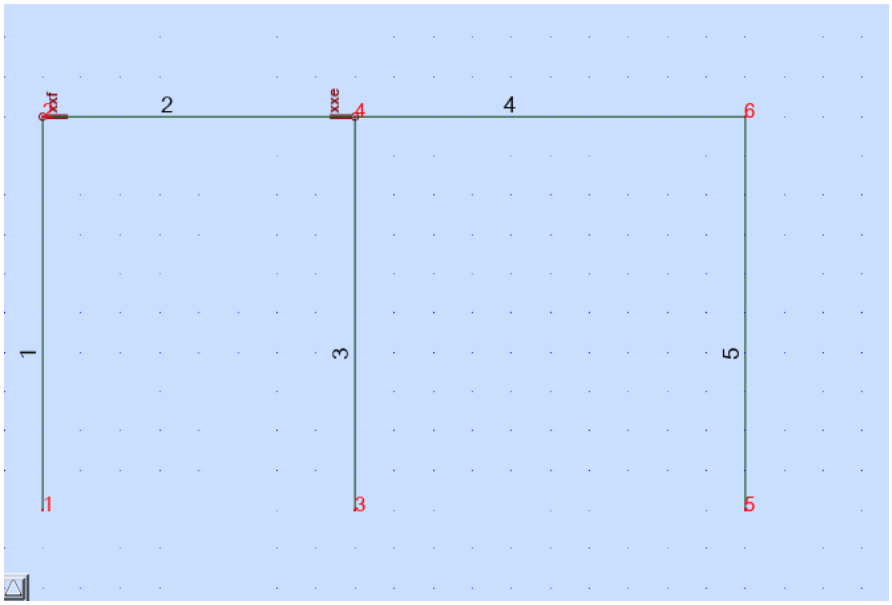


Figura 17: Representação da ligação articulada com a estrutura gerados a partir do código que consta no ponto 2 do anexo B - Aplicação do API do RSA.

3.4.3. Modelação de ligação não-linear

Um tipo de aplicação relevante corresponde à modelação de ligações com comportamento não linear, como é o caso de ligações metálicas semirrígidas. Nesses casos, a modelação estrutural implica a definição do comportamento não linear das ligações que se podem modelar através de relações momento-curvatura obtidas em função das caraterísticas do sistema de ligação. Tendo em consideração a necessidade de cálculos auxiliares prévios à modelação estrutural e o elevado conjunto de parâmetros a considerar, a utilização de rotinas automáticas exteriores pode ser útil, permitindo um trabalho mais rápido e eficiente.

O exemplo apresentado define uma ligação semirrígida com os parâmetros definidos através da aplicação COP [33], considerando-se as seguintes propriedades:
-Viga IPE 300;
-Pilar HEB 300;
-Chapas de extremidade espessura igual a 10 mm;
-Três linhas de parafusos M20, classe 10.9.

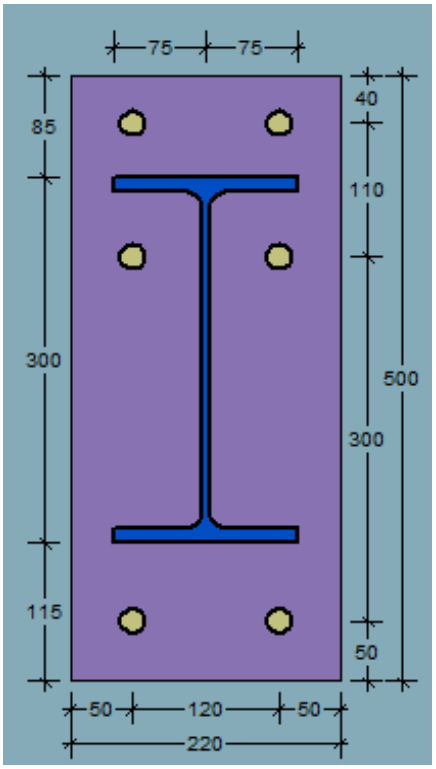


Figura 18: Perfil metálico capturado no Software COP [33].

O comportamento da ligação não-linear é traduzido pela relação Momento-Rotação conforme a Figura 15. No caso, o comportamento pode corresponder a uma ligação viga-pilar, com momentos negativos e positivos, como pode acontecer em ações sísmicas.

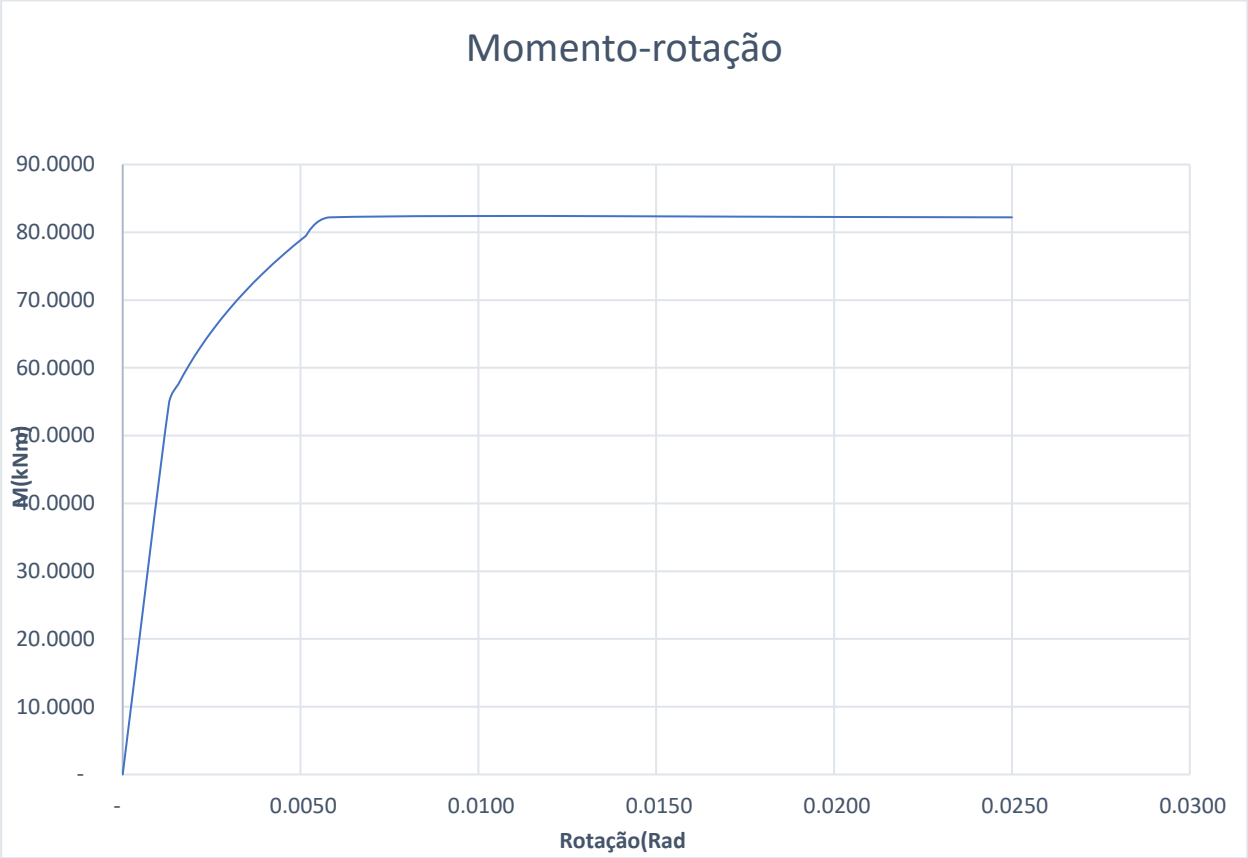


Figura 19: Gráfico Momento-Rotação.

O Quadro 79 define o modelo de ligação não linear baseado na curva momento-rotação atribuída à extremidade direita da viga do pórtico. Essa ligação não-linear é composta por uma curva momento-rotação com 13 pontos em cada direção (positiva e negativa), que cria uma liberação nas extremidades das barras, que permite que as barras se movam livremente em uma direção especificada, o que pode simular o comportamento de articulações estruturais. Conforme a segunda linha do Quadro 79, o código começa pela criação de um objeto rob_nlink da classe RobotApplication [25] para aceder aos recursos do RSA. Em seguida, são definidos objetos e variáveis para a ligação não-linear, bem como para a curva momento-rotação, incluindo as informações necessárias para definir a curva. Depois disso, a ligação não-linear é criada e são definidos seus parâmetros são definidos. Posteriormente, é atribuída a liberação nas extremidades das barras.

Quadro 79: Definição da ligação não-linear.

```
Sub Non_linear()
Dim rob_nlink As New RobotApplication
'Cria-se o objeto da classe ligação não linear
Dim RnlL As RobotNonlinearLink
'Cria-se variáve para os parâmetros da ligação não linear
Dim RnnLP As RobotNonlinearLinkParamsCustom
'Cria-se variável para os segmentos da ligação não linear
Dim RnnLS As RobotNonlinearLinkParamsCustomSegment
'Cria-se um label para libertação nas extremidades das barras
Dim release_label As RobotLabel
'Cria-se o objeto da classe libertação de barra
Dim release_bar As RobotBarReleaseData
'Nas linhas abaixo define-se a ligação não linear denominada como conection
'Remove uma ligação existente se já existir para definir uma nova
rob_nlink.Project.Structure.Nodes.NonlinearLinks.Remove
(rob_nlink.Project.Structure.Nodes.NonlinearLinks.Find("Conection"))
'Cria-se a ligação não linear denominada por KnlL
Set RnlL =rob_nlink. Project.Structure.Nodes.NonlinearLinks.Create("Conection")
'Define-se o tipo de curva no formato linear, parabólico, outro
RnlL.SetCurveType I_NLCT_CUSTOM, I_NLSAT_ANY
'Define-se que a ligação não linear é do tipo momento-rotação
RnlL.ModelType = RobotOM.IRobotNonlinearLinkModelType.I_NLMT_MOMENT_ROTATION
'Definição da parte negativa da curva de ligação
Set RnnLP = RnlL.GetParams(I_NLSAT_NEGATIVE)
'A definição da curva é feita com 13 pontos
'A definição da curve é feita com 6 segmentos
segments = 26
SegmentsN = 13
'Definição de um vetor para valores de momento com double
Dim momento(26) As Double
'Definição de um vetor para valores de rotação com double
Dim rota(26) As Double
'Definição dos pares de valores momento/rotação
For i = 1 To segments
momento(i) = Folha2.Cells(i, 6) * 1000
rota(i) = Folha2.Cells(i, 5)
Next
Dim q As Integer
q = 1
'Ciclo de 1 a 6 que define os valores da parte negativa da ligação
For n = SegmentsN To 1 Step -1
'Atribui os parâmetros da ligação aos segmentos da curva
Set RnnLS = RnnLP.New
'Define-se que não existem constantes
RnnLS.Constant = False
```

```
'Segmentos da curva em que expression corresponde ao momento (ordenada)
RnnLS.Expression = -1 * momento(n)
'Segmentos da curva em que .OriginPint corresponde às rotações (abcissa)
RnnLS.OriginPoint = -1 * rota(n)
'Os segmentos definidos são atribuídos aos parâmetros da ligação
RnnLP.Set q, RnnLS
q = q + 1
Next
'O objeto de ligação não linear recebe os parâmetros relativos à parte negativa
RnlL.SetParams RnnLP, I_NLSAT_NEGATIVE
'As linhas de código abaixo definem a parte positiva do troço
Set RnnLP = RnlL.GetParams(I_NLSAT_POSITIVE)
For n = 14 To segments
Set RnnLS = RnnLP.New
RnnLS.Constant = False
RnnLS.Expression = momento(n)
RnnLS.OriginPoint = rota(n)
RnnLP.Set n - 13, RnnLS
Next
'O objeto de ligação não linear recebe os parâmetros relativos à parte positiva
RnlL.SetParams RnnLP, I_NLSAT_POSITIVE
'rob_nlink.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_NONLINEAR_HINGE, "conection"
Set release_label = rob_nlink.Project.Structure.Labels.Create(I_LT_BAR_RELEASE,
"CR_Release")
'Definição do objeto release_bar que recebe os dados da libertação
Set release_bar = release_label.Data
'Indicação de que a ligação no nó de extremidade inicial é articulada
release_bar.StartNode.RY = I_BERV_STD
'Indicação do tipo de articulação do nó de extremidade final é articulada
'Esta definição tem que ser previamente definido antes dos tipos de ligações mais avançados
'como é o caso da ligação elástica
release_bar.EndNode.RY = I_BERV_STD
'Definição de um tipo de ligação elástica
'release_bar.EndNode.RY = I_BERV_NONLINEAR
'Definição do valor da rigidez da ligação
release_bar.EndNode.NonlinearModel.Set I_DOF_RY, "conection"
'release_bar.EndNode.UX = I_BERV_STD
'release_bar.EndNode.NonlinearModel.Set I_DOF_UX, "conection"
'Atribuição da libertação à barra 2
rob_nlink.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_RELEASE, "CR_Release"
'A libertação é armazenada no programa de cálculo
rob_nlink.Project.Structure.Labels.Store release_label
End Sub
```

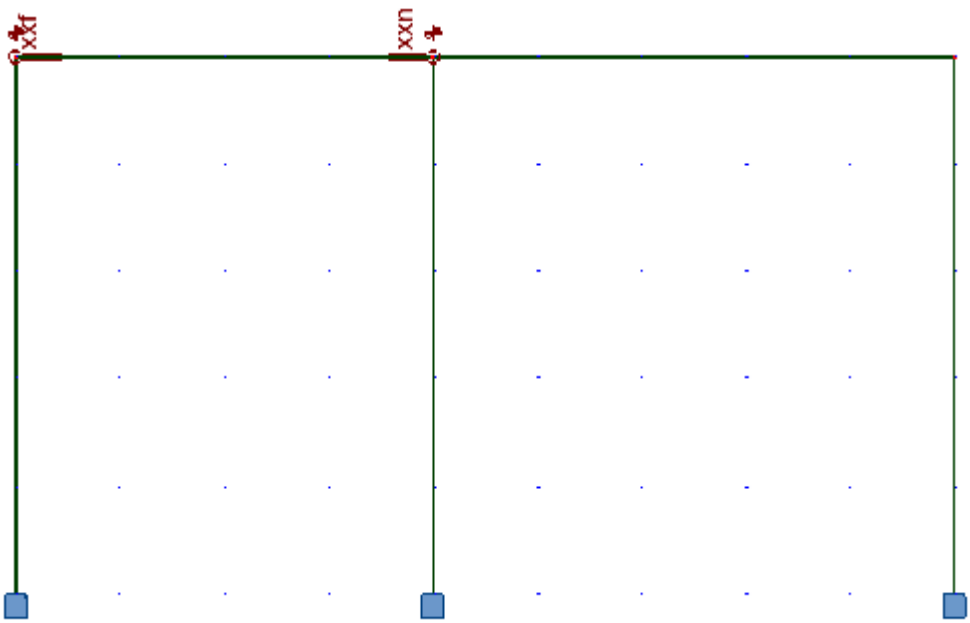


Figura 20: Representação da ligação não-linear.

3.5. Aplicação

Nesta secção é descrita a modelação e análise de um pórtico (Figura 21 e Figura 22), em que as ligações entre vigas e pilares têm comportamento não-linear, tal como explicado na secção 3.4. Abaixo constam a descrição das secções transversais das vigas e dos pilares, assim como das chapas de extremidades e dos parafusos.

-Viga IPE 300;
-Pilar HEB 300;
-Chapas de extremidade espessura igual a 10 mm;
-Três linhas de parafusos M20, classe 10.9.

A Figura 23 descreve a curva Momento-Rotação que foi considerada para as ligações entre todas as vigas e pilares.

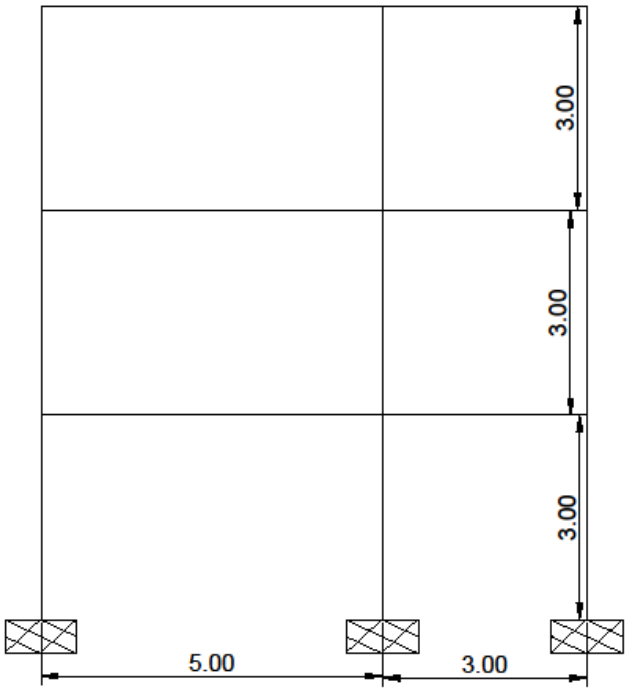


Figura 21: Representação do pórtico em análise.

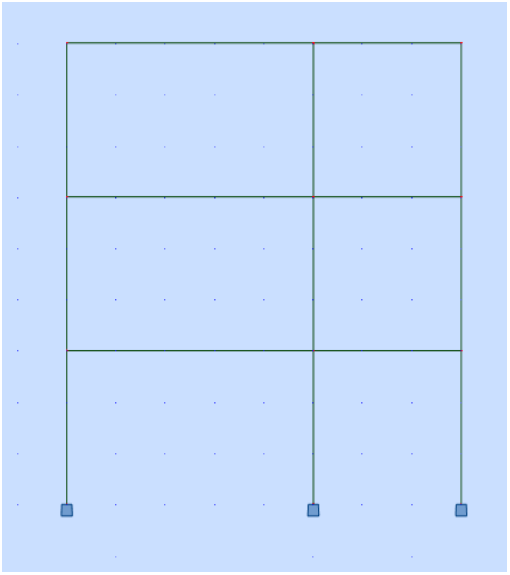


Figura 22: Modelação do pórtico em análise.

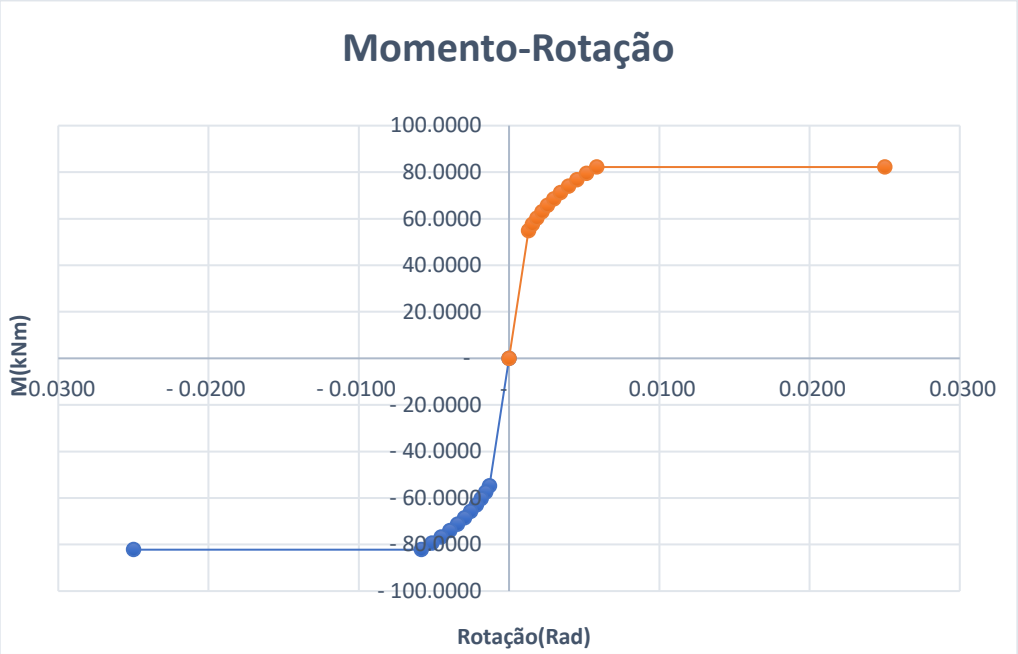


Figura 23: Gráfico Momento-Rotação.

Os blocos de código do anexo B - Aplicação do API do RSA, definem a formação da estrutura pretendida e a ligação não-linear, momento-rotação, atribuída à extremidade das vigas. As Figuras 24 a 26 mostram os resultados obtidos em termo de diagrama dos esforços.

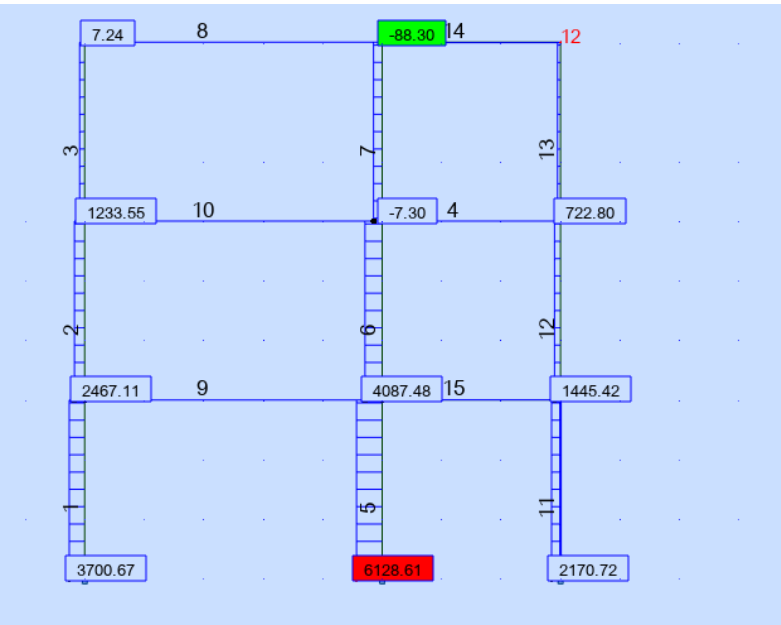


Figura 24: Representação do Esforço Axial com a estrutura e análise gerados a partir do código que consta no ponto 3 do Anexo B - Aplicação do API do RSA.

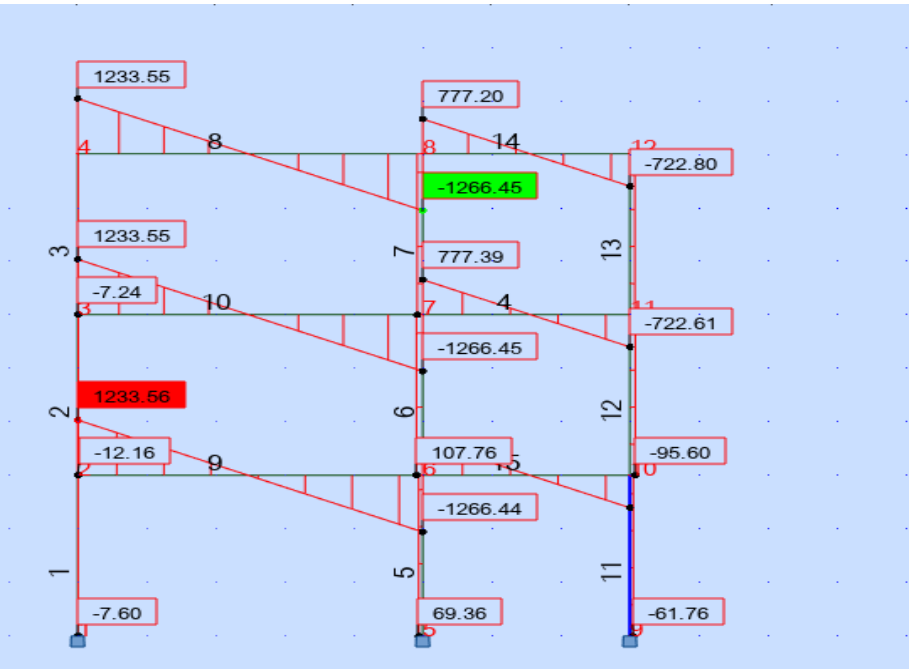


Figura 25: Representação do Esforço Transverso com a estrutura e análise gerados a partir do código que consta no ponto 3 do Anexo B - Aplicação do API do RSA.

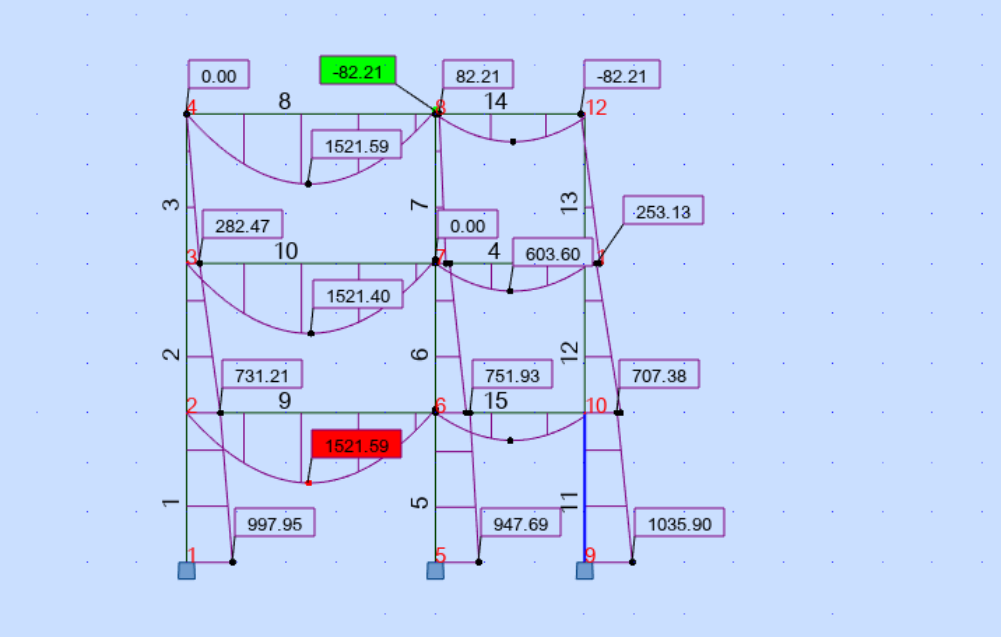


Figura 26: Representação do Momentos fletores (My) com a estrutura e análise gerados a partir do código que consta no ponto 3 do Anexo B - Aplicação do API do RSA.

3.6. Modelação e análise no tempo de ponte ferroviária

3.6.1. Enquadramento

Em pontes ferroviárias, especialmente para alta velocidade, torna-se necessário realizar uma análise dinâmica com variação posicional do comboio. No caso, considera-se o modelo de carga HSLM-A (Figura 26), conforme o Eurocódigo 1-Parte 2 [36]. Essa análise permite, entre outros aspetos, determinar os níveis de aceleração que podem ter implicações no nível de conforto dos passageiros.

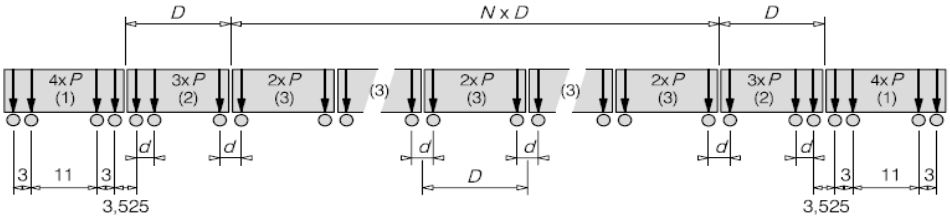


Figura 27: Modelo de carga HSLM-A [36]

A realização da análise implica considerar a variação posicional do comboio influenciada pelas dimensões e configuração do modelo de carga do comboio e da sua velocidade. Assim, ao longo do tempo da análise dinâmica é considerada a variação posicional do comboio, o que introduz complexidade na modelação do carregamento.
Na presente secção é descrita essa análise através de um conjunto sucessivos blocos de código que fazem o pré-processamento dos dados e criação do modelo.

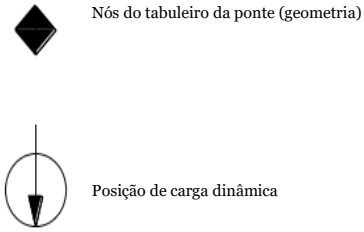
3.6.2. Descrição do Procedimento

A modelação do carregamento em função da variação posicional das cargas ao longo do tempo não é um processo direto, pelo que é preciso um pré-processamento conforme se explica abaixo.

A estrutura é modelada por nós que não traduzem todas as posições do comboio (Figura 27). Assim, torna-se necessário associar a cada nó dos elementos onde circulam os comboios funções de cargas temporais em função das posições do comboio. Para o efeito, e de modo simplificado, consideram-se funções lineares que determinam a influência das cargas do comboio em cada nó (Figura 28). Ou seja, cada nó é influenciado por cargas entre o mesmo e os dois nós adjacentes, frente e atrás, ou à esquerda e à direita (Figura 27). O código do Anexo C corresponde à definição das funções de carga associadas a cada

nó, em função do modelo de carga do comboio, da sua velocidade e da posição de cada nó do modelo estrutural.

Figura 28: Representa da posição do comboio numa ponte.



O modelo de comboio corresponde a 50 cargas pontuais, com posição relativa entre si definidas conforme as disposições do EC 1-2 [36]. Em função da posição relativa e da velocidade do comboio é possível determinar a posição de cada carga para um determinado instante de tempo, tendo como ponto de referência o primeiro nó do tabuleiro da ponte.

Nó em consideração

Outros nós

Cargas que influenciam o nó em consideração

Cargas que não influenciam o nó em consideração

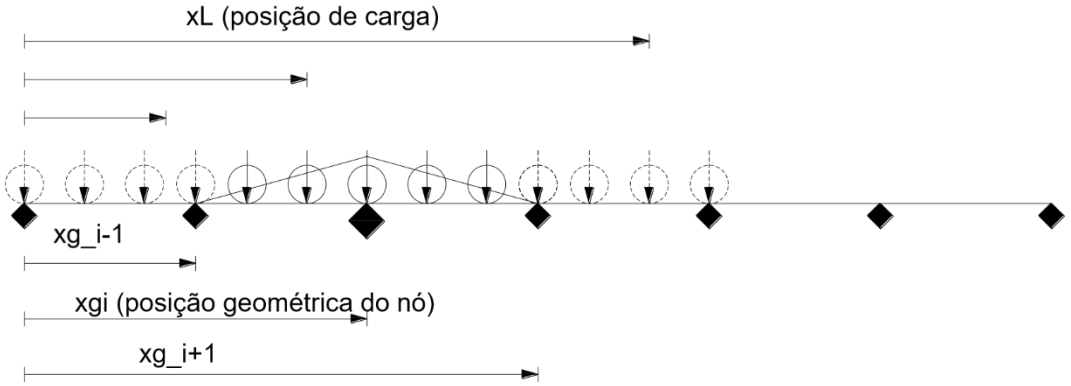


Figura 29: Definição da função forma com base no percurso do comboio em uma ponte.

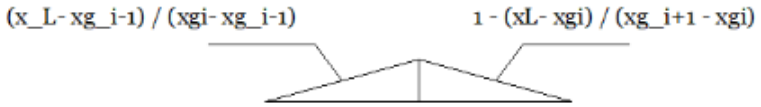


Figura 30: Definição simplificada da função de forma com base no percurso do comboio em uma ponte

O bloco de código do Anexo C determina para cada nó a função de carga temporal criando um conjunto de dados que posteriormente são considerados por uma aplicação criada no bloco de código do Anexo C, que define do RSA as funções temporais de carga dos nós. Note-se que para isso é preciso associar a cada nó do tabuleiro um caso de carga com valor unitário que o programa associa à função temporal e considera na análise.

Na secção seguinte apresenta-se um exemplo de aplicação para um caso de ponte ferroviária, em que a análise é feita com modelação através dos códigos apresentados no Anexo C.

3.6.3. Aplicação

Os códigos criados e que constam no Anexo C - Cálculo do viaduto em análise, foram aplicados para uma análise dinâmica de uma ponte ferroviária adaptada de [34]. O caso de estudo, da modelação e análise no tempo de ponte ferroviária, será o viaduto de Olzaileko, situado em País Basco [34], descrito nas Figuras 31 e 32. Para os materiais foi considerado Betão C50/60 e a velocidade máxima do comboio igual a 250 km/h.

- Alçado do viaduto

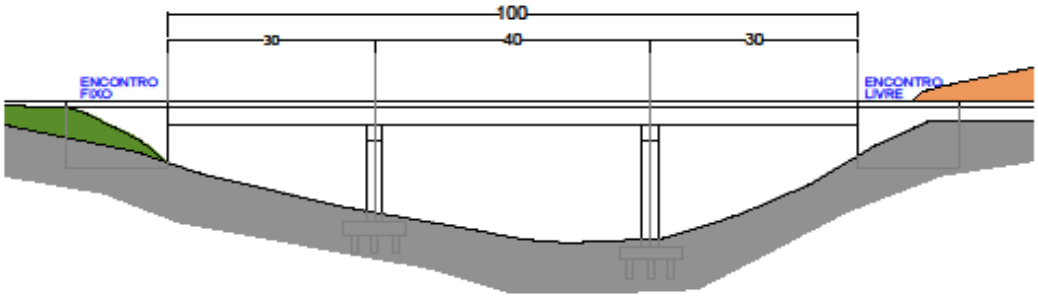


Figura 31: Alçado do viaduto adaptado [34].

- Secção do tabuleiro do viaduto

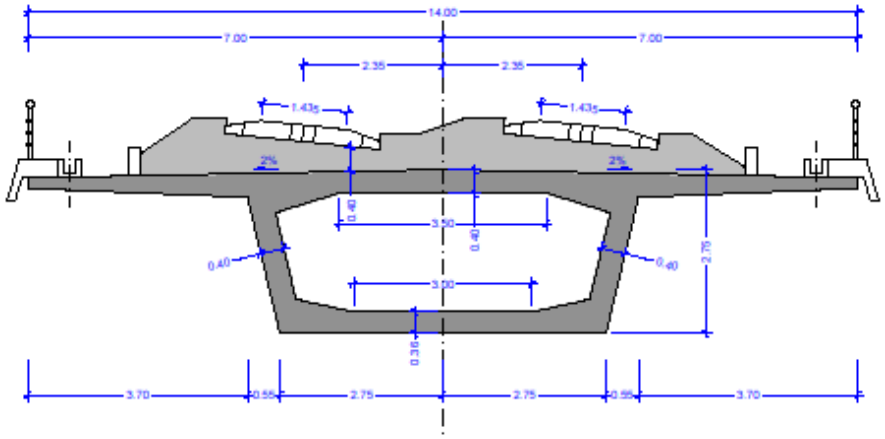


Figura 32: secção do viaduto adaptado [34].

A Figura 33, representa o viaduto em análise.

Figura 33: Modelo estrutural simplificado do tabuleiro.

Os códigos utilizados encontram-se no Anexo C - Cálculo do viaduto em análise.

A aplicação é descrita para a análise do comportamento de um nó, nomeadamente o nó 10 da Figura 33.

Os resultados da análise modal da estrutura surgem abaixo com representação dos modos de vibração e das frequências naturais.

A dinâmica foi feita recorrendo ao método de Newmark (aceleração) [45], com incrementos de tempo de 0.001, para um tempo total de análise de 13 segundos, com frequência de armazenamento de resultados de 1. A definição dos parâmetros de análise no RSA foi feita a partir da aplicação Excel, cujo código consta no Anexo C.

As Figuras 34 e 35 descrevem a função de carga ao longo do tempo num nó do tabuleiro, associado à passagem do comboio. A função descrita nas Figuras 34 e 35 resulta na variação observada em um nó localizado no início e mais distante da entrada da ponteEstas funções são definidas no RSA através da aplicação descrita no código do Anexo C - Cálculo do viaduto em análise.

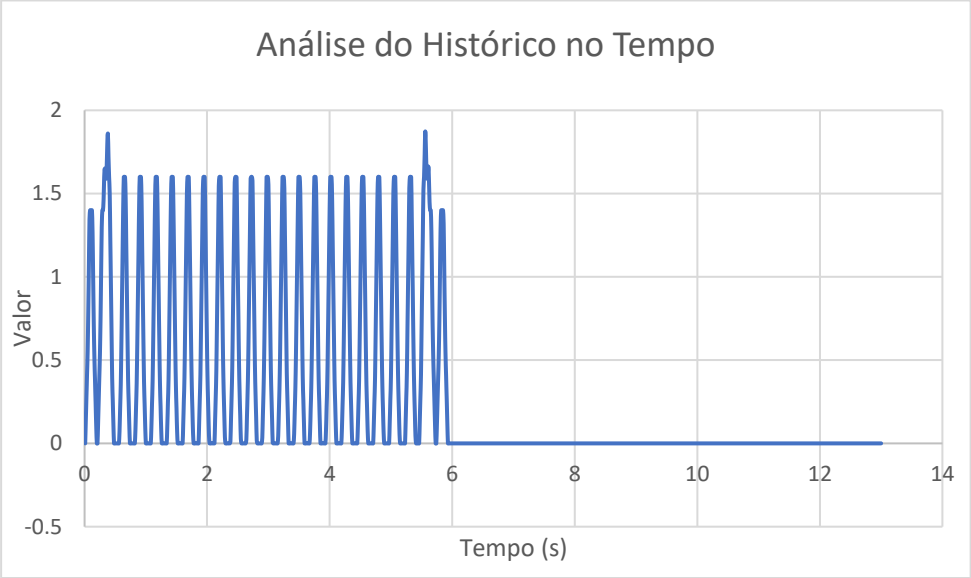


Figura 34: Gráfico que representa a variação do comboio ao longo do viaduto de um nó localizado no início da Tabuleiro da Ponte.

Figura 35: Gráfico que representa a variação do comboio ao longo do viaduto de um nó localizado no final da Tabuleiro da Ponte.

A análise dinâmica efetuada para a estrutura considerando um comboio correspondente ao modelo HSMA do EC1-parte 2, a uma velocidade de 250 km/h. permite obter a variação da aceleração dos nós do tabuleiro ao longo do tempo. A Figura 36 mostra a aceleração obtida para o nó 10 do tabuleiro. No anexo C constam outros resultados obtidos na análise.

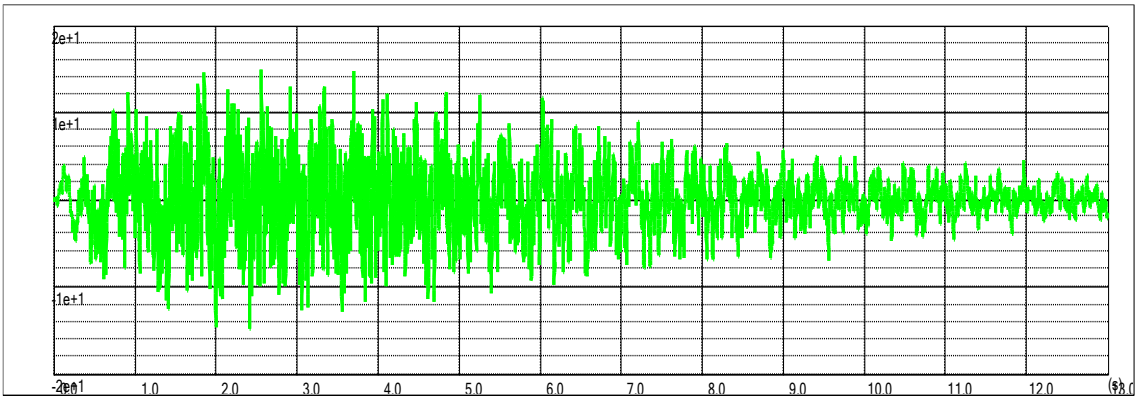


Figura 36:Gráfico da Aceleração do comboio no intervalo de tempo.

Capítulo 4

Análise estrutural com OpensSes e visualização de resultados com Paraview

4.1. Introdução

Nesta secção apresenta-se a criação de modos de visualização estrutural e de resultados associada à utilização de OpenSees. Esses modos de visualização são particularmente úteis tendo em conta que o OpenSees não tem interface gráfica. Para os objetivos pretendidos recorreu-se a estruturas de dados VTK (Visuaization Toolkit) [42]. Paralelamente, são utilizadas ferramentas de acesso livre que permitem a modelação geométrica e a geração de malhas de elementos finitos.

A modelação geométrica é feita utilizando o FreeCAD [39], com exportação dos dados em formato .STEP [39]. A geração de malhas é feita recorrendo ao Gmsh [40] que permite criar diferentes tipos de malhas de elementos finitos e a exportação das mesmas em diferentes formatos nomeadamente, formato .vtk. Assim é possível utilizar o mesmo formato na modelação estrutural e posteriormente na visualização de resultados.

Como ferramenta de visualização do formato .vtk, utiliza-se o ParaView [38], que permite visualizar e pós-processar os resultados de análise estrutural em termos de mapas, configurações deformadas, campos vetoriais e linhas de fluxo. Por outro lado, permite aceder a informações como identificadores e coordenadas dos nós, e identificadores e nós dos elementos finitos.

4.2. Descrição do formato VTK

Conforme [41], o formato.vtk é um sistema de código aberto para computação gráfica 3D, visualização científica e plotagem 2D. A sigla "VTK" [42] tem origem na expressão em inglês "Visualization ToolKit", que se refere a um conjunto de ferramentas de visualização de dados em 3D amplamente utilizadas em diversas áreas, incluindo a análise estrutural.

Na presente secção descreve-se o formato .vtk recorrendo a exemplos simples. A utilização mais complexa, nomeadamente em problemas de análise estrutural é descrita nos blocos que constam no Anexo D - Análise estrutural com OpenSees e visualização de resultados com Paraview, em que os formato .vtk são escritos de forma automática com organização e estruturação dos resultados da análise. Nos problemas descritos adotou-se a seguinte sequência:

- Criação da geometria utilizando FreeCad;
- Exportação da geometria para o Gmsh para criação da malha;
- Exportação da malha em formato. vtk inserida no ficheiro Excel;
- Análise estrutural;
- Escrita de resultados em formato vtk para leitura em ParaView.

A estrutura.vtk permite a visualização de diferentes dados. Abaixo descrevem-se e exemplificam-se as diferentes possibilidades em vtk.

- Pontos estruturados: representa um conjunto de pontos tendo por base a indicação do número de pontos, o ponto de origem e o espaçamento. Este tipo de representação pode ser mais limitado que uma representação não estruturada, mas facilita o código.

DATASET STRUCTURED_POINTS
DIMENSIONS n x ny nz
ORIGIN x y z
SPACING s x sy sz

- Uma das possibilidades mais práticas de representar dados em formato. vtk corresponde a grelhas não estruturadas, numa sequência de pontos e células, similares a elementos finitos planos. Nessa grelha não estruturada, a partir dos pontos é possível definir linhas, polilinhas, vértices e outros elementos geométricos.

Na escrita de ficheiros.vtk é necessário especificar o tipo de dados usando o código "dataset", seguido de um dos tipos de estrutura de dados:

- Rectilinear_Grid;
- Polydata;
- Unstructured_Grid.

No caso do formato Polydata, representam-se pontos, vértices, linhas, polilinhas e elementos triangulares.

No caso dos pontos é necessário definir o número de pontos, o tipo de dados e as coordenadas de cada ponto, com definição de x,y e z. A designação de cada ponto é feita pela ordem de definição, começando por zero. Esse identificador de ordem será o identificador de cada ponto na definição de outros elementos, vértices, linhas ou polilinhas.

DATASET POLYDATA
POINTS n dataType
P0x p0y p0z
p1x p1y p1z
...
p(n-1) x p(n-1) y p(n-1) z

Na definição dos vértices, é indicado o número de vértices presentes e o tamanho do bloco de dados. Em seguida, para cada vértice, é indicado o número de pontos presentes em cada linha, seguido pelos pontos correspondentes que formam as linhas.

VERTICES n size num
Points0, i0, j0, k0, ...
numPoints1, i1, j1, k1, ...
numPointsn-1, in-1, jn-1, kn-1, ...

Na definição de linhas a estruturação de dados é similar, conforme se descreve abaixo. Tal como na definição de pontos, é necessário definir o número de linhas e o tamanho do bloco de dados.

LINES n size

numPoints0, i0, j0, k0, ...

numPoints1, i1, j1, k1, ...

numPointsn-1, in-1, jn-1, kn-1, ...

Além disso, é possível representar dados usando malhas compostas por diferentes tipos de elementos. Existem códigos específicos para cada tipo de elemento. O código "Grid" é usado para representar essa malha, que pode ser estruturada ou não estruturada, conforme o Quadro 80.

Quadro 80: Definição de códigos para a representação de malhas.

Tipo	Código
Vértice	1
Vértices de uma polilinha	2
Linha com dois pontos	3
Linha com vários pontos	4
Triângulo de três nós	5
Banda de elementos triangulares	6
Elemento fechado formado por um polígono	7
Elemento quadrangular de quatro nós	8
Elemento trapezoidal de 4 nós	9
Elemento tetraedro	10
Elemento cúbico	11
Elemento Hexaedro	12
Elemento volumétrico triangular	13
Elemento piramidal	14

4.3. Aplicações

Na presente secção constam diferentes aplicações com o formato vtk, com representações de vários tipos de elementos. Esses exemplos servem de base a aplicações relativas à análise de estruturas com uso do OpenSeesPy. O Quadro 81 descreve a representação de um retângulo a partir das coordenadas dos vértices.

Quadro 81: Representação de um retângulo com 4 pontos e 4 linhas

```
# vtk DataFile Version 2.0
untitled
ASCII
DATASET POLYDATA
POINTS 4 float
0.0 0.0 0.0
1 0 0
1 1 0
0 1 0
LINES 4 12
2 0 1
2 1 2
2 2 3
2 3 0
```

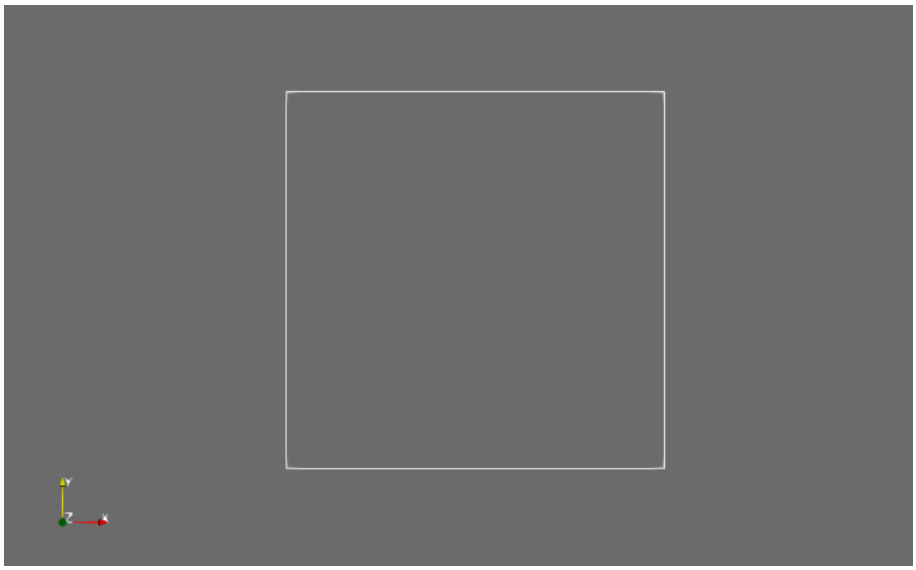


Figura 37: Representação de um retângulo com 4 pontos e 4 linhas.

O Quadro 82 mostra a representação de pontos, linhas e elementos trapezoidais de 4 nós. O código para as linhas é o código 3 e o código para o trapézio é o 9.

Quadro 82: Criação de malhas por 4 elementos.

```
# vtk DataFile Version 2.0
polinh
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 9 float
0.0 0.0 0.0
0.5 0 0
1 0 0
1 0.5 0
1 1 0
0.5 1 0
0 1 0
0 0.5 0
0.5 0.5 0
CELLS 16 56
2 0 1
2 1 2
2 2 3
2 3 4
2 4 5
2 5 6
2 6 7
2 7 0
2 1 8
2 5 8
2 3 8
2 7 8
4 0 7 8 1
4 1 2 3 8
4 8 3 4 5
4 7 8 5 6
CELL_TYPES 16
3
3
```

```
3
3
3
3
3
3
3
3
3
3
9
9
9
9
```

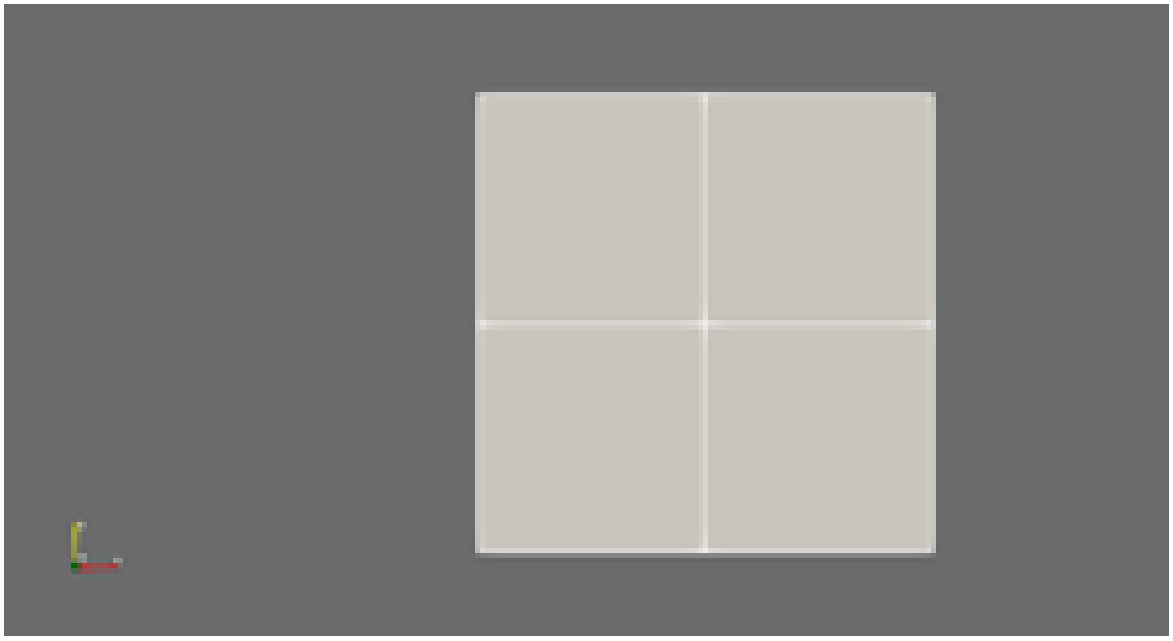


Figura 38: Representação da malha composta com 4 elementos.

A malha pode ser definida com elementos triangulares, que implica uma redefinição de cada elemento e da dimensão dos dados.

Quadro 83: Definição de uma malha de elementos triangulares.

```
# vtk DataFile Version 2.0
polinh
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 9 float
0.0 0.0 0.0
0.5 0 0
1 0 0
1 0.5 0
1 1 0
0.5 1 0
0 1 0
0 0.5 0
0.5 0.5 0
CELLS 19 65
2 0 1
2 1 2
2 2 3
2 3 4
2 4 5
2 5 6
2 6 7
2 1 8
2 5 8
2 3 8
2 7 8
3 0 1 7
3 1 7 8
3 1 3 8
3 1 2 3
3 8 4 3
3 8 5 4
3 7 5 6
3 8 7 5109
```

```
CELL_TYPES 19
3
3
3
3
3
3
3
3
3
3
3
5
5
5
5
5
5
5
5
```

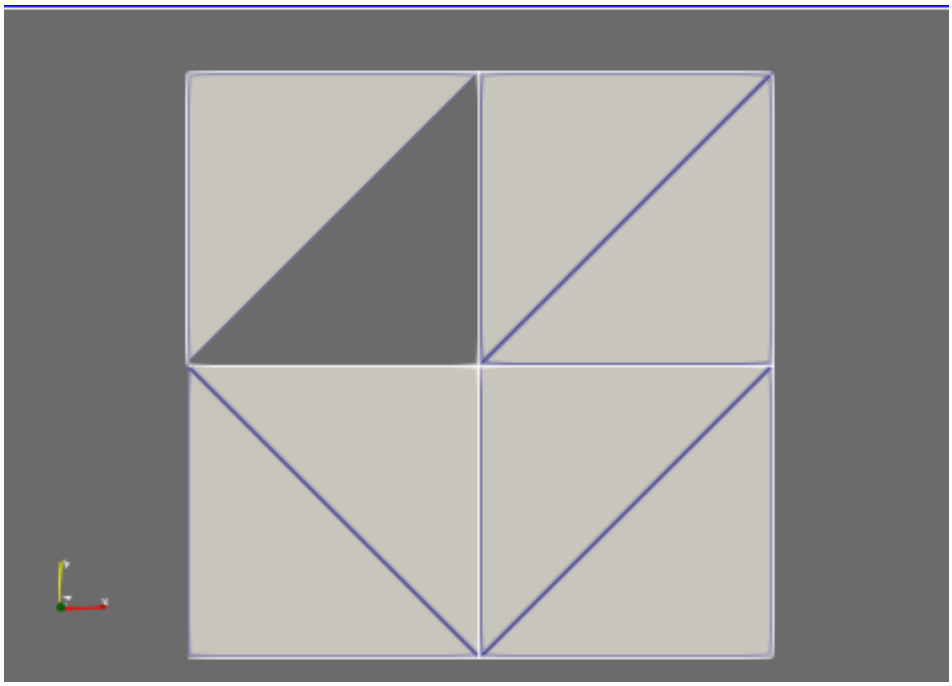


Figura 39: Representação da malha composta com elementos triangulares.

Aos pontos definidos é possível associar valores escalares que podem traduzir resultados de uma análise estrutural ou outra. O Quadro 84 mostra a representação de valores para cada um dos novos pontos definidos, o que permite a representação na forma de mapa.

Quadro 84: Criação de representação de valores em forma de mapa, associados ao valor escalar.

```
# vtk DataFile Version 2.0
polinh
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 9 float
0.0 0.0 0.0
0.5 0 0
1 0 0
1 0.5 0
1 1 0
0.5 1 0
0 1 0
0 0.5 0
0.5 0.5 0
CELLS 19 65
2 0 1
2 1 2
2 2 3
2 3 4
2 4 5
2 5 6
2 6 7
2 1 8
2 5 8
2 3 8
2 7 8
3 0 1 7
3 1 7 8
3 1 3 8
3 1 2 3
3 8 4 3
3 8 5 4
3 7 5 6
```

```
3 8 7 5
CELL_TYPES 19
3
3
3
3
3
3
3
3
3
3
3
5
5
5
5
5
5
5
5
POINT_DATA 9
SCALARS scalars float 1
LOOKUP_TABLE default
10
15
5
-2
6
5
6
5
20
```

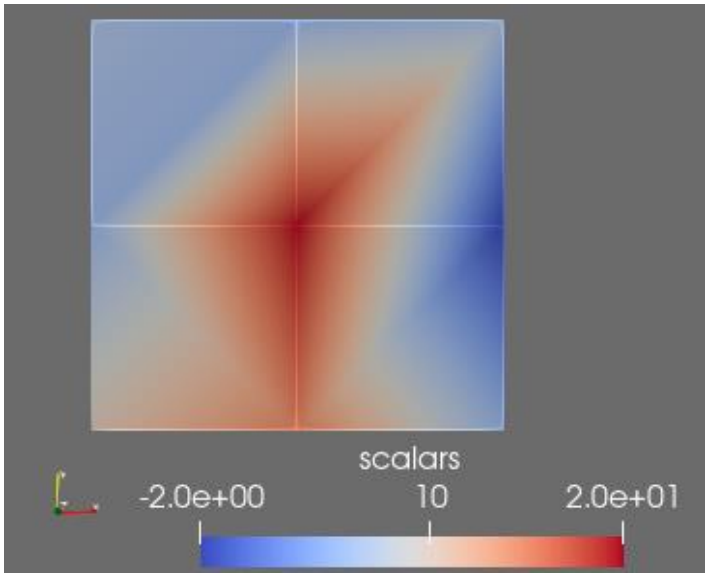


Figura 40: Criação de representação de valores em forma de mapa, associados ao valor escalar.

Para além de escalaras associados a cada ponto, é possível associar vetores. A diferença em relação aos escalares está na definição do tipo de dado e na dimensão da estrutura de dados. Os vetores são definidos de forma sucessiva para cada um dos pontos, em concordância com a sequência de definição dos pontos. Assim, não se torna necessário associar um identificador do ponto a que está associado o vetor.

Quadro 85: Criação de representação de valores em forma de mapa, associados ao valor vetorial.

```
# vtk DataFile Version 2.0
polinh
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 9 float
0.0 0.0 0.0
0.5 0 0
1 0 0
1 0.5 0
1 1 0
0.5 1 0
0 1 0
0 0.5 0
0.5 0.5 0
CELLS 19 65
2 0 1
```

2 1 2
2 2 3
2 3 4
2 4 5
2 5 6
2 6 7
2 1 8
2 5 8
2 3 8
2 7 8
3 0 1 7
3 1 7 8
3 1 3 8
3 1 2 3
3 8 4 3
3 8 5 4
3 7 5 6
3 8 7 5
CELL_TYPES 19
3
3
3
3
3
3
3
3
3
3
3
5
5
5
5
5
5
5
5

```
POINT_DATA 9
SCALARS scalars float 1
LOOKUP_TABLE default
10
15
5
-2
6
5
6
5
20
VECTORS vectors float
0.1 0.1 0
0.1 0.1 0
0.2 0.1 0
0 0.1 0
0.5 0.5 0
0.5 0.5 0
0.2 0.2 0
0.2 0.2 0
0.1 0.1 0
```

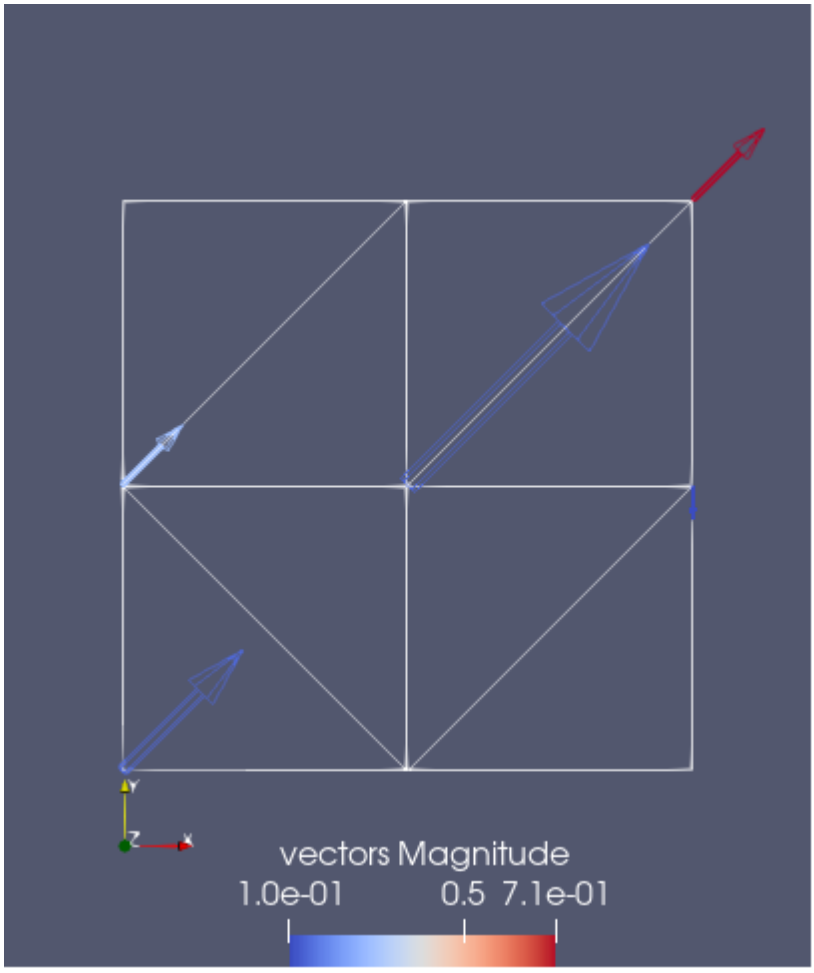


Figura 41: Criação de representação de valores em forma de mapa, associados ao valor vetorial.

A análise de dados pode implicar diferentes representações agrupadas de forma sucessiva. O código do Quadro 85 exemplifica essa representação.

Quadro 86: Criação de representação de valores em forma de mapa, associados ao valor escalar e vetorial

```
# vtk DataFile Version 2.0
polinh
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 9 float
0.0 0.0 0.0
0.5 0 0
1 0 0
1 0.5 0
1 1 0
0.5 1 0
```

```
0 1 0
0 0.5 0
0.5 0.5 0
CELLS 19 65
2 0 1
2 1 2
2 2 3
2 3 4
2 4 5
2 5 6
2 6 7
2 1 8
2 5 8
2 3 8
2 7 8
3 0 1 7
3 1 7 8
3 1 3 8
3 1 2 3
3 8 4 3
3 8 5 4
3 7 5 6
3 8 7 5
CELL_TYPES 19
3
3
3
3
3
3
3
3
3
3
3
3
5
5
5
```

```
5
5
5
5
5
POINT_DATA 9
SCALARS scalars_1 float 1
LOOKUP_TABLE default
10
15
5
-2
6
5
6
5
20
SCALARS scalars_2 float 1
LOOKUP_TABLE default
100
15
5
-200
6
5
6
500
20
VECTORS Vector_1 float
0.1 0.1 0
0.1 0.1 0
0.2 0.1 0
0 0.1 0
0.5 0.5 0
0.5 0.5 0
0.2 0.2 0
0.2 0.2 0
0.1 0.1 0
```

```
VECTORS Vector_2 float
0.1 0.1 0
0.1 0.1 0
0.2 0.1 0
0 0.1 0
0.5 0.5 0
0.5 0.5 0
0.2 0.2 0
0.2 0.2 0
0.1 0.1 0
```

4.3.1. Aplicações em OpenSeesPy

O exemplo de aplicação de OpenSeesPy correspondem a um problema em estado plano de tensão, criados com as ferramentas descritas acima. A modelação estrutural é feita a partir de dados colocados em folhas de cálculo do Microsoft Excel, contendo coordenadas dos nós, nós dos elementos, dados dos materiais, tipos de elementos, apoios, elementos de interface e códigos para definição do modelo .vtk.

Os problemas apresentados correspondem a uma parede, uma viga e dois problemas de análise de contacto.

4.3.1.1. Parede sujeita a cargas concentradas

O primeiro problema de aplicação do OpenSeesPy com visualização de resultados corresponde a uma parede com duas cargas concentradas e dois apoios colocados nos cantos inferiores esquerdo e direito. No Anexo D constam os códigos que realizam a análise estrutural e o pós-processamento com criação de um ficheiro vtk. Os dados da estrutura desde geometria, elementos finitos e apoios são obtidos a partir de uma folha de cálculo, correspondendo a uma forma de gerir e manipular a informação da estrutura. Os resultados da análise são armazenados num ficheiro .xls que pode ser utilizado para pós-processamento de resultados e que é acedido pela aplicação que gera o formato .vtk que é visualizado em Paraview, contendo a configuração deformada, mapas de tensões e campos vetoriais. As Figuras 43 a 45 descrevem a geometria da malha, os mapas de tensões e a configuração deformada.

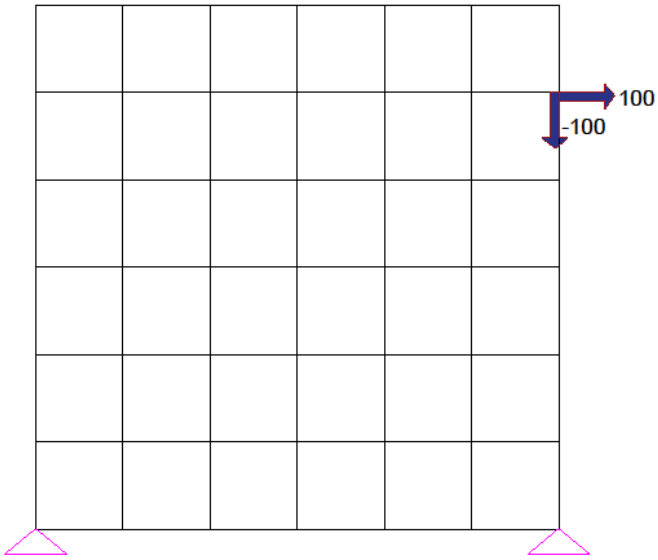


Figura 42: Descrição do modelo da estrutura com cargas e apoios.

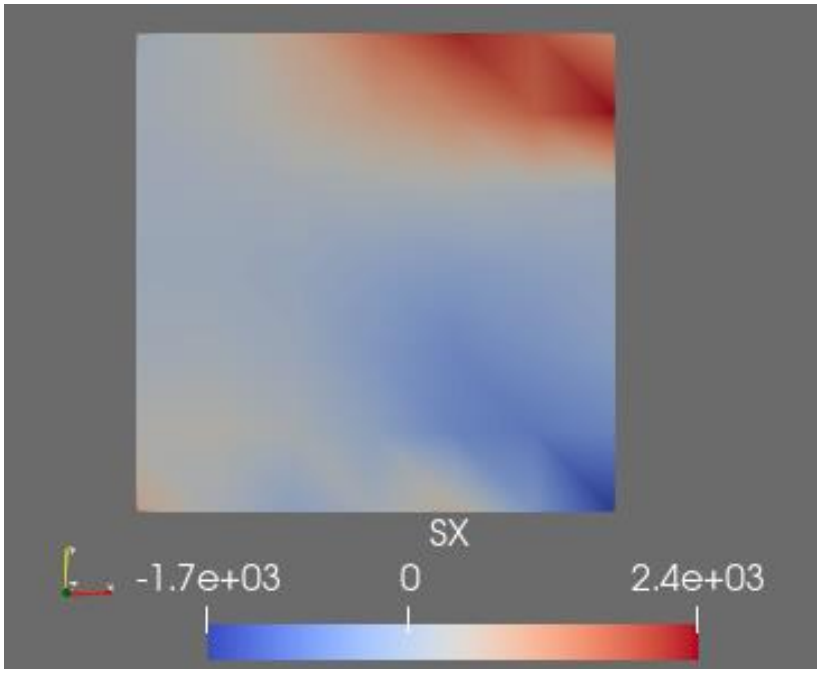


Figura 43: Representação do mapa de tensões normais da estrutura tipo parede na direção x.

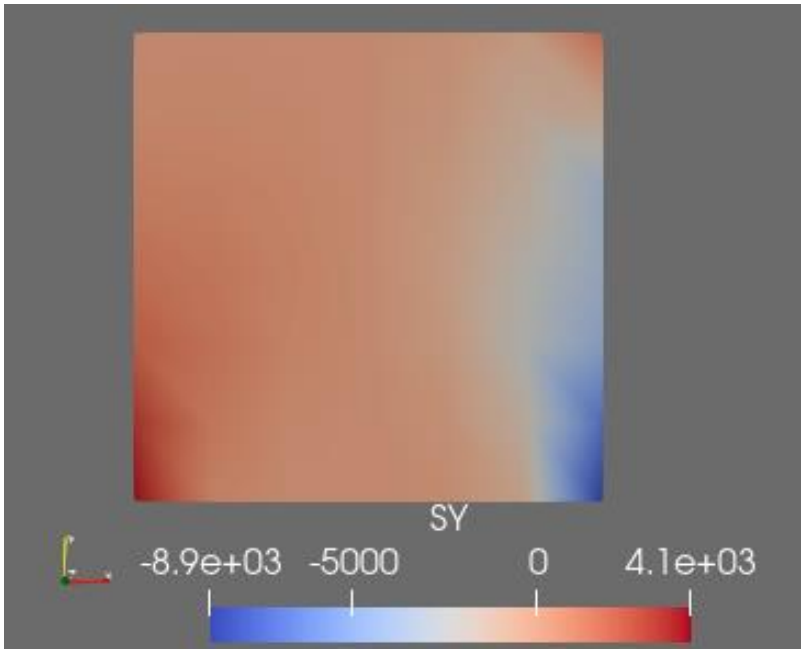


Figura 44: Representação do mapa de tensões normais da estrutura tipo parede na direção y.

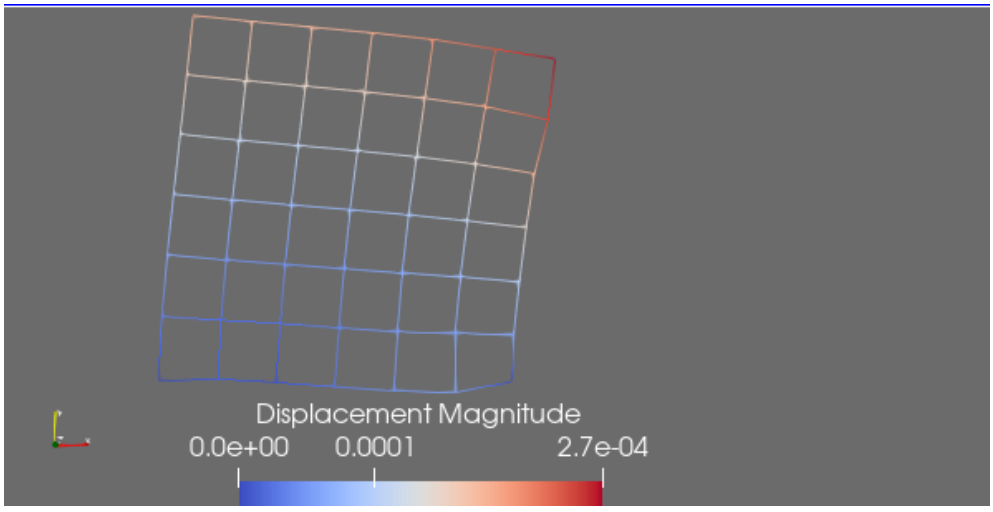


Figura 45: Representação da configuração deformada da malha de elementos finitos da estrutura tipo parede.

4.3.1.2. Viga sujeita a cargas concentradas

O problema em análise corresponde a uma viga com as dimensões, material e propriedades descritos no Anexo D, nos blocos de códigos 4 e 5. O carregamento é aplicado na parte superior da viga, nomeadamente, aos nós 41 e 42 da estrutura e dois apoios nos cantos inferiores (Figura 46). Os blocos de código 4 e 5 do Anexo D, são similares aos blocos de códigos 1 e 2 do mesmo anexo, têm como objetivo criar uma viga, ou seja, um modelo estrutural bidimensional composto por elementos triangulares. O bloco de código 3, permite a visualização de resultados e da modelação estrutural. As Figuras 47 a 49 apresentam a geometria da malha em Paraview, recorrendo ao formato . vtk. As Figuras 47 a 49 mostram os mapas de tensões e a configuração deformada para a viga em análise. No programa de visualização Paraview é possível ajustar as escalas dos mapas de tensões, alterar a visualização assim como a escala da configuração deformada.

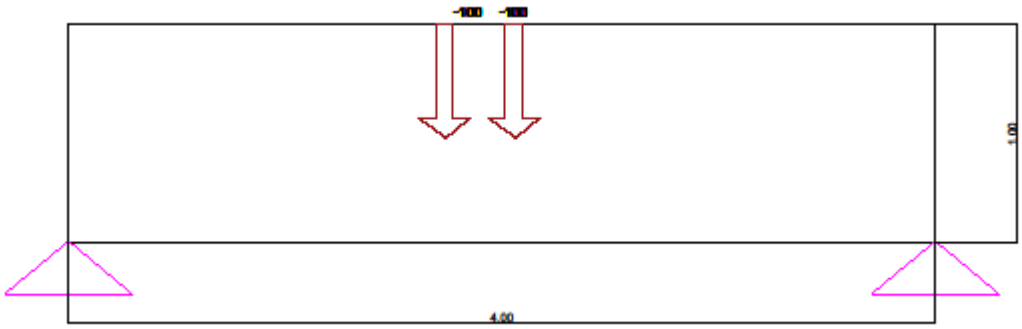


Figura 46: Descrição do modelo da malha triangular com cargas e apoios.

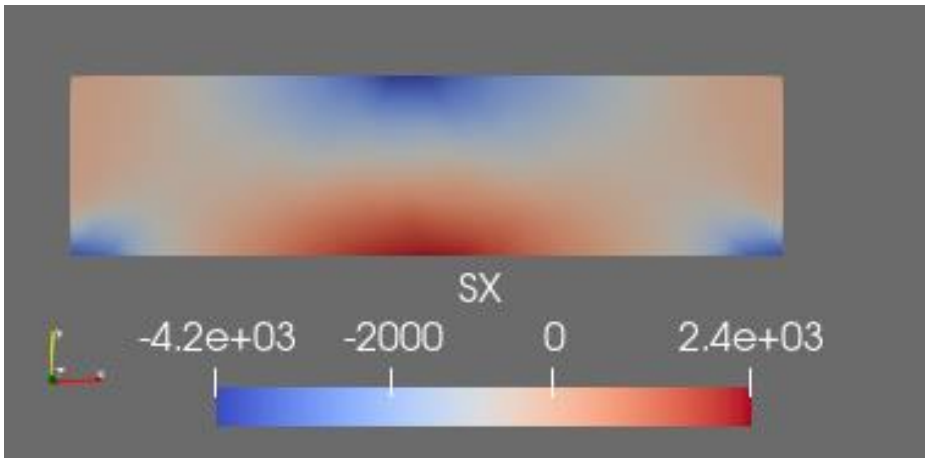


Figura 47: Representação do mapa de tensões normais do modelo com malha triangular na direção x.

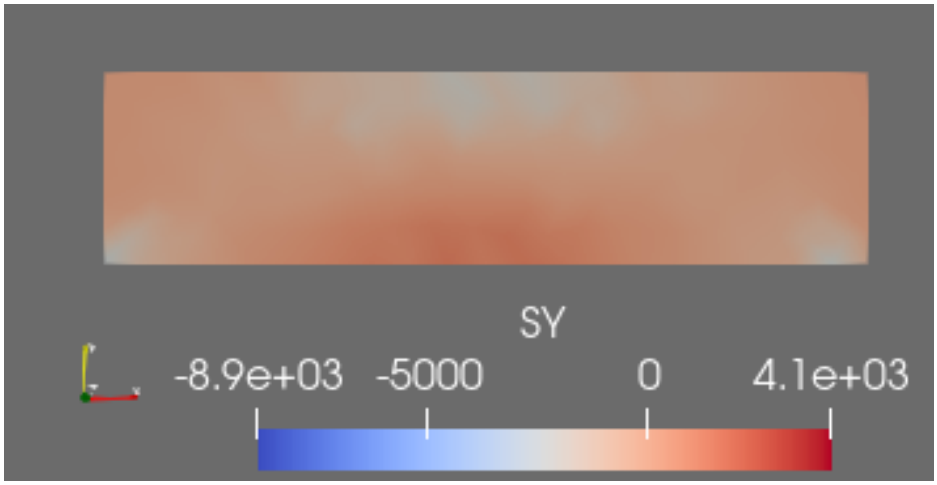


Figura 48: Representação do mapa de tensões normais do modelo com malha triangular na direção y.

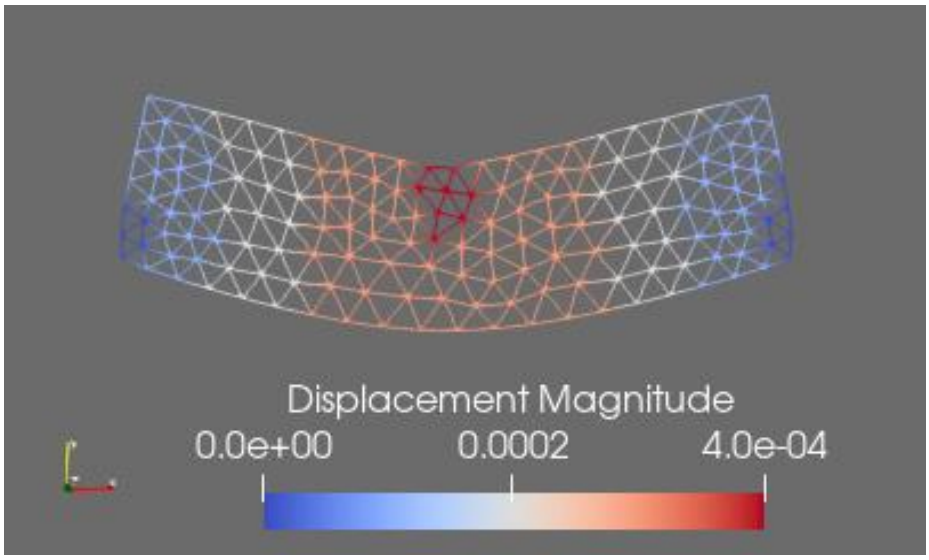


Figura 49: Representação da configuração deformada de elementos finitos do modelo com malha triangular.

Assim como o exemplo anterior, no problema em análise pretende-se em definir uma viga de modelo estrutural bidimensional composto por elementos quadrilaterais. Os blocos de código 7 e 8 do Anexo D, descrevem as dimensões, material e propriedades da estrutura. O carregamento é aplicado na parte superior da viga, nomeadamente, aos nós 54 e 56 da estrutura e os apoios estão colocados nos dois cantos inferiores (Figura 50). As figuras 51 a 53 apresentam a geometria da malha em ParaView, recorrendo ao formato ".vtk". As Figuras 51 a 53 mostram os mapas de tensões e a configuração deformada para a viga, a partir de formato .vtk visualizados no Paraview.

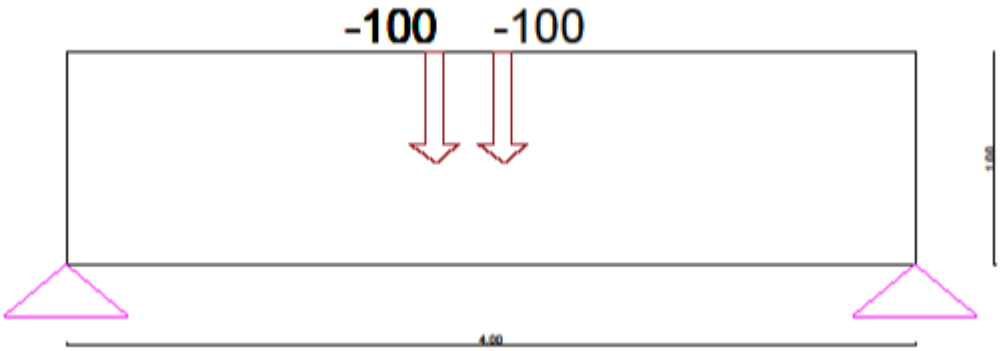


Figura 50: Descrição do modelo com cargas e apoios.

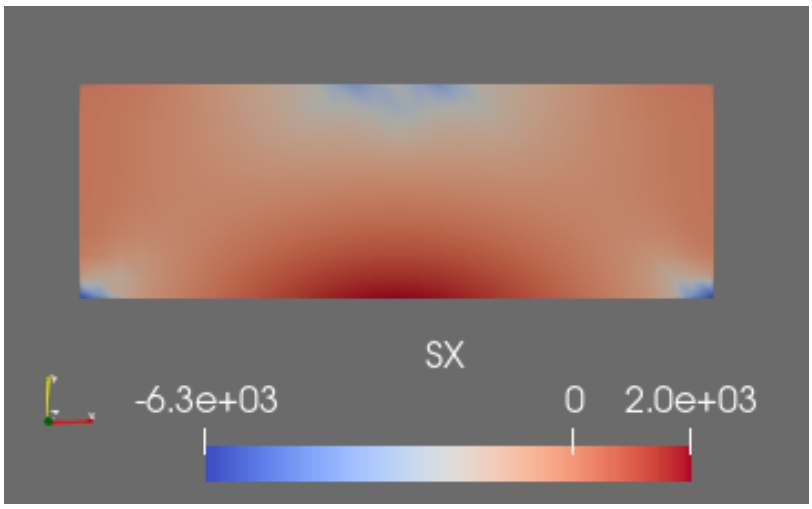


Figura 51: Representação do mapa de tensões normais do modelo na direção x.

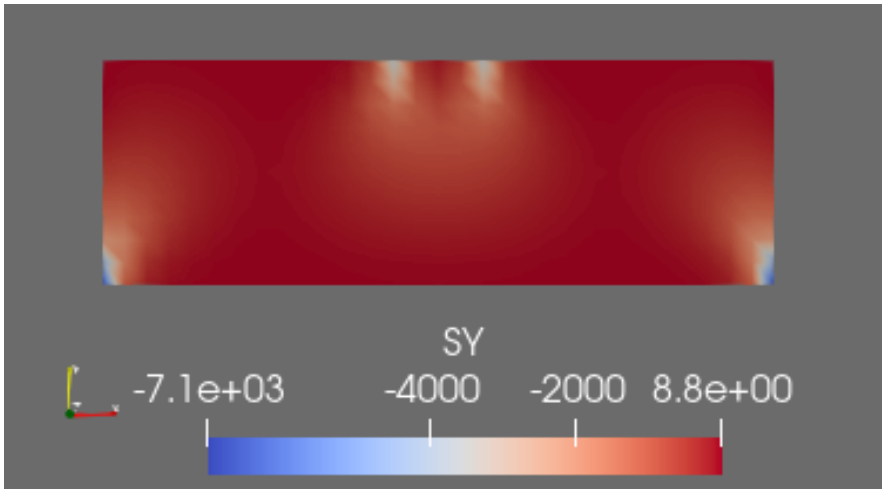


Figura 52: Representação do mapa de tensões normais do modelo na direção y.

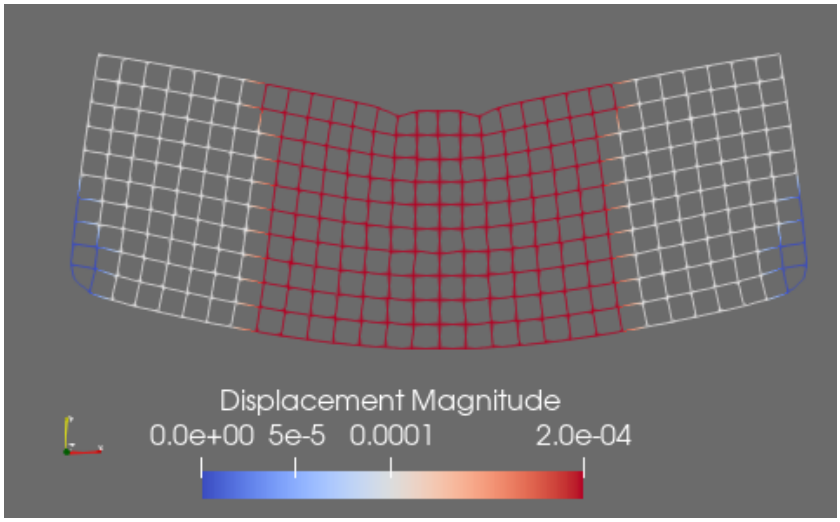


Figura 53: Representação da configuração deformada da malha de elementos finitos

Capítulo 5

Conclusões Finais

Na presente dissertação, foi exemplificada a utilização de programas de cálculo estrutural por meio de aplicações externas. Essa exemplificação foi realizada por meio de diferentes exemplos que, de forma genérica, apresentam os passos essenciais e permitem a compreensão dos conceitos.

Adicionalmente, foi demonstrada a importância de se entender conceitos como a tecnologia API e as noções básicas de programação orientada a objetos, tanto em Visual Basic quanto em Python.

No que respeita às linguagens de programação, parecem ser claras as vantagens de utilização de Python, pela sua versatilidade e pelo vasto conjunto de bibliotecas que permitem uma grande variedade de aplicações.

Os exemplos apresentados com a API do RSA mostraram as vantagens da sua utilização, na medida em que tornam desnecessárias tarefas muito repetitivas. Adicionalmente, embora não tenha sido apresentada de forma concreta, foi demonstrada a modelação da geometria estrutural, que pode ser utilizada para problemas complexos, como estruturas tipo casca ou arcos.

Na segunda parte da dissertação, é exemplificada a modelação e análise estrutural por meio de aplicações externas, recorrendo exclusivamente a ferramentas de acesso livre, tanto para a modelação geométrica quanto para a análise estrutural e o pós-processamento. Tornou-se possível demonstrar que, recorrendo a conhecimentos elementares de programação, é possível uma utilização eficiente e viável para a resolução de problemas de diferentes níveis de complexidade.

O conjunto de aplicações apresentado e explicado mostra que é justificável o esforço para melhorar os conhecimentos de programação dos estudantes e profissionais de engenharia civil, tendo em vista que esses conhecimentos melhoram as capacidades de utilização do ponto de vista do usuário.

O uso futuro deste tema pode incluir a integração de programas de cálculo estrutural em modelagem estrutural e software de design, permitindo que os usuários obtenham resultados precisos enquanto minimizam o tempo de trabalho. Isso pode aumentar a eficiência e a produtividade em diversas áreas, como engenharia civil, arquitetura e construção, entre outras. Além disso, a utilização de programas de cálculo estrutural por meio de aplicativos externos pode permitir o desenvolvimento de novas soluções e tecnologias para análise avançada de estruturas complexas, contribuindo para o avanço e inovação em diversas áreas da engenharia e da ciência.

É necessário ter em conta que esta é uma área em constante desenvolvimento, pelo que qualquer trabalho de desenvolvimento futuro deverá ter em conta todas as alterações efetuadas às linguagens de programação e software.

Referências

[1] Take.blog, acedido em 03 de março de 2022, em: https://www.take.net/blog/tecnologia/api-conceito-e-exemplos/.

[2] Maplink Blog, acedido 12 de março de 2022, em: https://maplink.global/blog/o-que-e-api/.

[3] Kevan, J. M., & Ryan, P. R. (2016). Experience API: Flexible, Decentralized and Activity-Centric Data Collection. *Technology, Knowledge and Learning*, *21*(1), 143–149.

[4] TechTudo, acedido 12 de março de 2022, em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml.

[5] ComputerWeekly, acedido 13 de março de 2022, em: https://www.computerweekly.com/br/definicoe/Interface-de-programacao-de-aplicacao-API.

[6] Tulach, J. (2008). *Practical API Design: Confessions of a Java Framework Architect*. Apress.

[7] *GDC Data Portal User's Guide TCGA User*. (sem data). UserManual.wiki. Obtido 13 de março de 2022.

[8] Rockcontent, acedido 15 de março 2022, em: https://rockcontent.com/br/blog/linguagem-de-programacao/.

[9] Pereira, Pedro de França, Lourenço, Pedro Guerra, & Bergamaschi, Marcelo Pereira. (2018, março 7). *Conceitos Iniciais sobre a Ferramenta Computacional Blender*.

[10] Pressman, R. S., & Maxim, B. R. (2021). *Engenharia de software—9.ed*. McGraw Hill Brasil.

[11] DevMedia, acedido 15 de março de 2022, em: https://www.devmedia.com.br/orientacao-a-objetos/.

[12] EXCELCUTE, acedido 21 de novembro de 2022, em: https://excelcute.com/ejemplo-modulo-clase-vba/.

[13] Excelsmart, acedido 21 de dezembro de 2022, em: https://excelsmart.com.br/modulo-de-classe-no-vba/.

[14] Tutorialspoint, acedido 11 de julho de 2022, em: https://www.tutorialspoint.com/vb.net/vb.net_classes_objects.htm.

[15] PERRY, G. M. (1999). Aprenda em 21 dias Visual Basic 6. Gulf Professional Publishing.

[17] Del Sole, A. (2015). Visual Basic 2015 Unleashed. Sams Publishing.

[18] Newsome, B. (2015c). Beginning Visual Basic 2015. John Wiley & Sons.

[19] Microsoft, acedido em 09 de abril de 2023, em: https://www.microsoft.com/pt-pt/microsoft-365/excel.

[20] Downey, Allen B. (2008). Think Python: How to Think Like a Computer Scientist. Green Tea Press.

[21] Stroher, P. (2009). Aprenda Computação com Python 3.0.

[22] Anaconda Navegador, acedido em 03 de agosto 2022, em: https://www.anaconda.com/products/distribution.

[23] Welcome to Python.org, acedido em 03 de agosto 2022, em: https://www.python.org/.

[24] Apostila Python, acedido em 09 de agosto 2022 , em: https://www.alura.com.br/apostila-python-orientacao-a-objetos/o-que-e-python#python.

[25] Autodesk. (2022). Robot Open Standard-Version 8.1.

[26] Autodesk, acedido 21 de dezembro de 2022, em: https://www.autodesk.com/education/edu-software/overview.

[27] Autodesk, acedido 21 de 12 de 2022, em: https://www.autodesk.com/products/robot-structural-analysis/overview.

[28] Autodesk Revit. (2015). Integrating Autodesk Revit,Revit Structure, and RobotStructural Analysis Professional. Autodesk University.

[29] OpenSeesPy, acedido em 19 de dezembro de 2022, em: https://openseespydoc.readthedocs.io/en/latest/.

[30] OpenSees, caedido em 19 de dezembro de 2022, em: https://opensees.berkeley.edu/.

[31] Mazzoni, S., McKenna, F., Scott, M. H, & Fenves, G. L. (2006). OpenSees command language manual. Pacific Earthquake Engineering Research (PEER) Center.

[32] Zhu, M., McKenna, F., & Scott, M. H. (2018). OpenSeesPy: Python library for the OpenSees finite element framework. SoftwareX, 7, 6–11.

[33] Cesdb, acedido em 23 de dezembro de 2022, em: https://www.cesdb.com/acop-arcelormittal-connection-program.html.

[34] Adão, A., Ortega, M., & Mato, M. (2010). Três viadutos em betão pré-esforçado em linhas ferroviárias de alta velocidade atravessando zonas montanhosas.

[35] EC1. prEN1991-1: Eurocode 1: Actions on structures - Part 1-1: General actions - Densities, self-weight, imposed loads for buildings. CEN,Brussels, final draft, 2002.

[36] EC1. prEN1991-2: Eurocode 1: Action on Structures – Part 2: Traffic Loads on Bridges. CEN,Brussels, final draft, 2002.

[37] EC3. prEN1993-1: Eurocode 3: Design of steel structures - Part 1-8: Design of joints.CNE.Brussels, final draft, 2005.

[38] Paraview, acedido em 01 de abril de 2023, em: https://www.paraview.org/about/.

[39] FreeCad, acedido em 01 de abril de 2023, em: https://www.freecad.org/.

[40] Gmsh, acedido em 01 de abril de 2023, em:
https://gmsh.info/.

[41] VTK, acedido em 01 de abril de 2023, em:
https://vtk.org/about/#history.

[42] Schroeder, Will; Martin, Ken; Lorensen, Bill (2006), The Visualization Toolkit (4ª ed.), Kitware, ISBN 978-1-930934-19-1.

[43] Opsvis, acedido em 05 de abril de 2023, em:
https://opsvis.readthedocs.io/en/latest/ex_quads_4x4.html.

[44] Scribd, acedido em 12 de abril de 2023, em:
https://www.scribd.com/document/398367212/Robot-API.

[45] Autodesk RSA Profisional 2022, acedido em 21 de 12 de 2022, em:
https://help.autodesk.com/view/RSAPRO/2022/ENU/?guid=GUID-D61A82B0-3B4B-49E4-A7F8-F302E129CE4B.

[46] FreeCad, acedido em 10 de abril de 2023, em:
https://wiki.freecad.org/Manual:Introduction.

[47] Numpy, acedido em 24 de abril de 2023, em:
https://numpy.org/.

[48] Matplotlib, acedido em 24 de abril de 2023, em:
https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html.

Anexos

Anexo A - Robot Open Standard

Quadro A 1 - Definições das interfaces e identificadores utilizados no Robot Object Model.

IRobotApplication: A interface que representa todo o aplicativo é o objeto principal do Objeto Robot Modelo.
IRobotNode: interface que representa um nó de uma estrutura.
IRobotValuesArray: Interface representando uma tabela unidimensional de números reais, ou seja, interface que matrizes.
IRobotNamesArray: A interface que descreve uma tabela unidimensional de nomes representados por caracteres cordas.
IRobotNumbersArray: A interface que representa uma tabela de números inteiros. Os elementos da tabela são indexados de 1 a contar.
IRobotObjectsArray: A interface que fornece acesso à tabela de objetos.
IRobotNumbersCollection: Coleção de números (inteiros).
IRobotCodeRegistrar: A fim de simplificar o processo de registo de códigos externos estendendo o programa Robot.
IRobotPointsArray: Tabela de pontos no espaço 3D.
IRobotPreferences: Interface que descreve as configurações atuais para todo o aplicativo.
IRobotProjectType: interface que descreve o conjunto de valores constantes
ou identificadores (Tal interface representa o tipo de enumeração, Identificadores referentes a diferentes tipos de projetos disponíveis no Robot são os seus membros. A fim de usar um membro de tal interface, basta para fornecer o seu nome)
Robot Object Model permite gerenciar todo um projeto (tarefa) salvo em um formato RTD arquivo. Um conjunto de identificadores é definido para se referir aos tipos de estrutura reconhecidos pelo Robot.
I_PT_FRAME_2D : int = 1
I_PT_TRUSS_2D : int = 2
I_PT_GRILLAGE : = 3
I_PT_FRAME_3D : = 4
I_PT_TRUSS_3D : = 5
I_PT_PLATE : = 6
I_PT_SHELL : = 7
I_PT_AXISYMMETRIC : = 8
I_PT_VOLUMETRIC : = 9
I_PT_CONCRETE_BEAM : = 10
I_PT_CONCRETE_COLUMN : = 11
I_PT_FOUNDATION : = 12
I_PT_PARAMETRIZED : = 13
I_PT_STEEL_CONNECTION : = 14
I_PT_SECTION : = 15
I_PT_PLANE_STRESS : = 16
I_PT_PLANE_DEFORMATION : = 17
I_PT_CONCRETE_DEEP_BEAM : = 18
IRobotProjectPreferences: Interface que descreve as configurações para o projeto atual (trabalho).

IRobotMaterialDatabase: A interface que fornece acesso ao banco de dados de materiais.
IRobotCodeType: Conjunto de identificadores determinando os tipos de códigos que podem ser configurados no programa Robot.
Alguns casos:
I_CT_STEEL_STRUCTURES : = 0
the code for steel structures
I_CT_STEEL_CONNECTIONS : = 1
the code for steel connections
I_CT_TIMBER_STRUCTURES : = 2
the code for timber structures
I_CT_RC_REAL_REINF : = 3
the code for real reinforcement
I_CT_RC_THEORETICAL_REINF : = 4
the code for theoretical reinforcement
I_CT_FOUNDATIONS_DESIGN : = 5
code for spread footing design
I_CT_CODE_COMBINATIONS : = 9
code combinations
I_CT_SNOW_WIND_LOADS : = 6
code for snow/wind loads
I_CT_SEISMIC_LOADS : = 7
code for seismic loads
I_CT_MOVING_LOADS : = 8
IRobotSectionDatabase: A interface que fornece acesso ao banco de dados da seção.
IRobotUnitType
I_UT_STRUCTURE_DIMENSION : = 1
Available since version 3.5.
I_UT_SECTION_DIMENSION : = 2
Available since version 3.5.
I_UT_SECTION_PROPERTIES : = 3
Available since version 3.5.
I_UT_STEEL_CONECTIONS : = 4
Available since version 3.5.
I_UT_DIAMETER_RC_BASE : = 5
Available since version 3.5.
I_UT_REINFORCEMENT_AREAS : = 6
Available since version 3.5.
I_UT_FORCE : = 7
Available since version 3.5.
I_UT_MOMENT : = 8
Available since version 3.5.
I_UT_STRESS : = 9
Available since version 3.5.
I_UT_DISPLACEMENT : = 10
Available since version 3.5.
I_UT_ANGLE_ROTATION_DATA : = 11

Available since version 3.5.
I_UT_ANGLE_ROTATION_RESULT : = 12
Available since version 3.5.
I_UT_TEMPERATURE : = 13
I_UT_WEIGHT : = 14
Available since version 3.5.
I_UT_MASS : = 15
Available since version 3.5.
I_UT_DIMENSIONLESS_QUALITY : = 16
Available since version 3.5.
I_UT_RULER : = 17
Available since version 3.5.
IRobotUnitData: Interface de definição de unidade.
IRobotUnitEditionType: interface de tipos de unidades editáveis.
I_UMT_LENGTH : = 0
Available since version 3.5.
I_UMT_FORCE : = 1
Available since version 3.5.
I_UMT_MASS : = 2
IRobotProjectComponentType: Um conjunto de identificadores é definido para descrever diferentes tipos de componentes que podem ser salvos em um projeto.
I_PCT_BEAM : int = 3
concrete beam
I_PCT_COLUMN : int = 4
I_PCT_FOOT : int = 5
concrete footing
I_PCT_JOINT : int = 6
steel connection
I_PCT_DRAWING : int = 7
plotter drawing
I_PCT_BWALL : int = 8
concrete wall
I_PCT_RETAINING_WALL : = 10
Retaining wall
Available since version 3.
I_PCT_SLAB : = 9
Available since version 3.
I_PCT_CONTINUOUS_FOOT : = 11

Quadro A 2 - Definições de interfaces e conjuntos de valores relacionados à sapata de abertura e sua análise estrutural.

IRConcrFooting: Interface representando a sapata de abertura.
IRConcrFootingDimType: Um conjunto de valores que identificam as dimensões individuais da sapata de abertura.
I_CFDT_FOOT_DIM_A : = 1
Available since version 3.
I_CFDT_FOOT_DIM_B : = 2
Available since version 3.
I_CFDT_FOOT_DIM_H1 : = 3
Available since version 3.
I_CFDT_FOOT_DIM_H2 : = 4
Available since version 3.
I_CFDT_FOOT_DIM_H3 : = 5
Available since version 3.
I_CFDT_FOOT_DIM_H4 : = 6
Available since version 3.
I_CFDT_FOOT_DIM_EX : = 7
Available since version 3.
I_CFDT_FOOT_DIM_EY : = 8
Available since version 3.
I_CFDT_FOOT_DIM_AP : = 9
Available since version 3.
I_CFDT_FOOT_DIM_BP : = 10
Available since version 3.
I_CFDT_FOOT_DIM_L : = 11
I_CFDT_FOOT_DIM_COL_A : = 12
Available since version 3.
I_CFDT_FOOT_DIM_COL_B : = 13
IRConcrFootingShapeType: Um conjunto de valores que definem a forma da sapata.
IRConcrFootingType: Um conjunto de valores que definem o tipo de fundação.
I_CFT_FOOT_TYPE_FOOTING : = 1
column footing
Available since version 3.
I_CFT_FOOT_TYPE_WALL : = 2
IRConcrFootingLoads
IRConcrFootingResults

Quadro A 3 - Propriedades e operações da estrutura em IRobot, uma visão geral.

IRobotExternalFileFormat: Um conjunto de identificadores é definido para descrever diferentes formatos de arquivos externos aceitos pelo Robot.

Todos os tipos de dados usados para definir e modelar uma estrutura (nós, barras, casos de carga) são gerenciados pelo Robot Object Model e pelo Data Server. O servidor também fornece acesso aos resultados dos cálculos efetuados para as estruturas definidas.

IRobotStructure: representa uma estrutura inteira. Componentes de estrutura particulares são representados por componentes de interface apropriados.

Properties:

Labels : IRobotLabelServer
Nodes : IRobotNodeServer
Bars : IRobotBarServer
Cases : IRobotCaseServer
Results : IRobotResultServer
Selections : IRobotSelectionFactory
Objects : IRobotObjObjectServer
FiniteElems : IRobotFiniteElementServer
Groups : IRobotGroupServer
Edit : IRobotStructureEditTools
ResultsFreeze : bool
Storeys : IRobotStoreyMngr
GroupObjects : IRobotGroupObjectServer
QuantitySurvey : IRobotStructureQuantitySurvey
Type : IRobotProjectType

Operations:

CreateCache () : IRobotStructureCache
ApplyCache (_struct_cache : IRobotStructureCache) : IRobotStructureApplyInfo
ExportXml (_input_xml : string, _output_xml : string)
Clear ().
Merge (_data : IRobotStructureMergeData, _params : IUnknown)

Quadro A 4 - Definições de interfaces de especificações e atributos relacionados às barras no RSA.

IRobotBar

Cada barra em uma estrutura é representada no Modelo de Objeto por um objeto de dados do tipo: I_OT_BAR. Os dados e funcionalidades relacionados à barra estão disponíveis por meio da interface RobotBar de histórico de tempo.

Os seguintes atributos de barras complexas são definidos por meio da rotulagem

mecanismo:

Section I_LT_BAR_SECTION

Release I_LT_BAR_RELEASE

Material I_LT_BAR_MATERIAL

Offset I_LT_BAR_OFFSET

IRobotBarEnd: Pode-se indicar um começo e um fim de cada compasso. O objeto do tipo RobotBarEnd, que descreve uma das extremidades da barra, é criado para permitir a realização de operações que afetem apenas o início ou o fim de uma barra (e não toda a barra).

IRobotBarTensionCompression:

I_BTC_STANDARD : = 0

the bar carries both compressive and tensile stresses

Available since version 2.5.

I_BTC_TENSION_ONLY : = 1

I_BTC_COMPRESSION_ONLY : = 2

the bar carries only compressive stresses

IRobotBarEndBracketType

I_BEBT_PLATES : = 1

I_BEBT_SECTION : = 2

IRobotBarEndBracketDataValue:

I_BEBDV_LENGTH : = 1

I_BEBDV_HEIGHT : = 2

I_BEBDV_WIDTH : = 3

I_BEBDV_THICKNESS_1 : = 4

I_BEBDV_THICKNESS_2 : = 5

IRobotBarSectionShapeType:

I_BSST_UNKNOWN : = 0

I_BSST_CAE : = 1

I_BSST_CAEP : = 2

I_BSST_CAI : = 3

I_BSST_CAIP : = 4

I_BSST_DCEC : = 5

I_BSST_DCED : = 6

I_BSST_DCEP : = 7

I_BSST_DCIG : = 8

I_BSST_DCIP : = 9

I_BSST_HEA : = 10

I_BSST_HEAA : = 11

I_BSST_HEB : = 12

I_BSST_HEC : = 13

I_BSST_HEM : = 14

I_BSST_HER : = 15
I_BSST_HHEA : = 16
I_BSST_HHEB : = 17
I_BSST_HHEM : = 18
I_BSST_IIPE : = 19
I_BSST_IPE : = 20
I_BSST_IPEA : = 21
I_BSST_IPEO : = 22
I_BSST_IPER : = 23
I_BSST_IPEV : = 24
I_BSST_IPN : = 25
I_BSST_MHEA : = 26
I_BSST_MHEB : = 27
I_BSST_MHEM : = 28
I_BSST_MIPE : = 29
I_BSST_PRS : = 30
I_BSST_TCAR : = 31
I_BSST_TEAE : = 32
I_BSST_TEAI : = 33
I_BSST_THEX : = 34
I_BSST_TREC : = 35
I_BSST_TRON : = 36
I_BSST_UAP : = 37
I_BSST_UPN : = 38
I_BSST_UUAP : = 39
I_BSST_UUPN : = 40
I_BSST_FRTG : = 41
I_BSST_UPAF : = 42
I_BSST_BOX : = 91
I_BSST_RECT : = 92
I_BSST_TUBE : = 93
I_BSST_ISYM : = 94
I_BSST_INSYM : = 95
I_BSST_TUSER : = 96
I_BSST_CUSER : = 97
I_BSST_TBETC : = 98
I_BSST_WELD_CROSS : = 201
I_BSST_DRECT : = 99
Available since version 1.7.
I_BSST_COLD_SIGMA1 : = 1001
Available since version 1.7.
I_BSST_COLD_SIGMA2 : = 1002
Available since version 1.7.
I_BSST_COLD_ZED1 : = 1003
Available since version 1.7.
I_BSST_COLD_U : = 1004

Available since version 1.7.
I_BSST_COLD_CE1 : = 1005
I_BSST_COLD_ANGL : = 1006
Available since version 1.7.
I_BSST_COLD_OMEGA : = 1007
Available since version 1.7.
I_BSST_COLD_SO1 : = 1008
Available since version 1.7.
I_BSST_COLD_RIVE1 : = 1009
Available since version 1.7.
I_BSST_USER_BOX : = 91
Available since version 2.5.
I_BSST_USER_RECT : = 92
Available since version 2.5.
I_BSST_USER_TUBE : = 93
Available since version 2.5.
I_BSST_USER_I_BISYM : = 94
Available since version 2.5.
I_BSST_USER_I_MONOSYM : = 95
Available since version 2.5.
I_BSST_USER_T_SHAPE : = 96
Available since version 2.5.
I_BSST_USER_C_SHAPE : = 97
Available since version 2.5.
I_BSST_USER_CROSS : = 201
Available since version 2.5.
I_BSST_WOOD_RECT : = 41
timber rectangular section
Available since version 2.5.
I_BSST_WOOD_DRECT : = 99
timber double rectangular section
Available since version 2.5.
I_BSST_RECT_FILLED : = 43
steel rectangular solid section
Available since version 2.5.
I_BSST_CIRC_FILLED : = 44
steel round solid section
Available since version 2.5.
I_BSST_CONCR_COL_R : = -108
Available since version 2.5.
I_BSST_CONCR_COL_T : = -107
Available since version 2.5.
I_BSST_CONCR_COL_L : = -106
Available since version 2.5.
I_BSST_CONCR_COL_Z : = -105
Available since version 2.5.

I_BSST_CONCR_COL_P : = -104
Available since version 2.5.
I_BSST_CONCR_COL_C : = -103
Available since version 2.5.
I_BSST_CONCR_COL_CH : = -102
Available since version 2.5.
I_BSST_CONCR_COL_CQ : = -101
Available since version 2.5.
I_BSST_CONCR_BEAM_RECT : = -3
RC rectangular beam
Available since version 2.5.
I_BSST_CONCR_BEAM_T : = -2
Available since version 2.5.
I_BSST_CONCR_BEAM : = -1
RC beam of any shape
Available since version 2.5.
I_BSST_COMP_2C_FACE : = 1101
two C-sections face to face
Available since version 4.
I_BSST_COMP_2C_BACK : = 1102
two C-sections back to back
Available since version 4.
I_BSST_COMP_2I : = 1103
two I-sections
Available since version 4.
I_BSST_COMP_CI : = 1104
C-section and I-section
Available since version 4.
I_BSST_COMP_2LI : = 1105
two angles and one I-section
Available since version 4.
I_BSST_COMP_4L_FACE : = 1106
four angles face to face
Available since version 4.
I_BSST_COMP_4L_BACK : = 1107
four angles with legs back to back
Available since version 4.
I_BSST_COMP_2L_SHORT : = 1108
two angles with shorter legs back to back
Available since version 4.
I_BSST_COMP_2L_LONG : = 1109
two angles with longer legs back to back
Available since version 4.
I_BSST_COMP_2L_CROSS : = 1110
two angles - cross shape
Available since version 4.

I_BSST_USER_BOX_2 : = 102
I_BSST_CCL : = 45
semi-closed C-section
I_BSST_URND : = 46
rounded C-section – rotated
I_BSST_TRND : = 47
rectangular pipe
I_BSST_CUAP : = 48
C-sections set face to face, welded
I_BSST_WOOD_CIRC : = 100
round timber section - solid
I_BSST_USER_CIRC_FILLED : = 101
user steel section - round, solid
I_BSST_USER_POLYGONAL : = 103
pipe section - shaped like a regular polygon
I_BSST_COLD_C_PLUS : = 1010
cold-formed C-section - semi-closed, with bends
I_BSST_COLD_SIGMA_SL : = 1011
cold-formed Sigma SL section
I_BSST_COLD_SIGMA : = 1012
cold-formed Sigma section
I_BSST_COLD_Z : = 1013
cold-formed Z-section
I_BSST_COLD_L_LIPS : = 1014
cold-formed C-section - semi-closed
I_BSST_COLD_Z_ROT : = 1015
cold-formed Z-section in the local - central coordinate system
I_BSST_COMP_2L_FACE_SHORT : = 1111
two angles - shorter legs
I_BSST_COMP_2L_FACE_LONG : = 1112
two angles - longer legs
I_BSST_COMP_CI_BACK : = 1113
C-section and I-section set back to back
I_BSST_COMP_2C_FACE_WELD : = 1201
two C-sections set face to face, welded
I_BSST_COMP_2C_BACK_WELD : = 1202
two C-sections set back to back, welded
I_BSST_COMP_2I_WELD : = 1203
two welded I-sections
I_BSST_COMP_CI_WELD : = 1204
C-section and I-section set face to face - welded
I_BSST_COMP_2LI_WELD : = 1205
two angles and I-section - welded
I_BSST_COMP_4L_FACE_WELD : = 1206
four angles with legs set face to face - welded
I_BSST_COMP_4L_BACK_WELD : = 1207

four angles set in the form of a cross - welded

I_BSST_COMP_2L_SHORT_WELD : = 1208

two angles - shorter legs, welded

I_BSST_COMP_2L_LONG_WELD : = 1209

two angles - longer legs, welded

I_BSST_COMP_2L_CROSS_WELD : = 1210

two angles set in the form of a cross - welded

I_BSST_COMP_2L_FACE_SHORT_WELD : = 1211

I_BSST_COMP_2L_FACE_LONG_WELD : = 1212

I_BSST_COMP_CI_BACK_WELD : = 1213

IRobotBarSectionDataValue: Um conjunto de identificadores é definido para determinar parâmetros específicos (atributos) que descrevem uma seção de barra. Esse identificador é fornecido como o parâmetro de uma função que obtém ou define o valor do atributo de seção relevante.

I_BSDV_AX : = 0

I_BSDV_AY : = 1

I_BSDV_AZ : = 2

I_BSDV_IX : = 3

I_BSDV_IY : = 4

I_BSDV_IZ : = 5

I_BSDV_VY : = 6

I_BSDV_VPY : = 7

I_BSDV_VZ : = 8

I_BSDV_VPZ : = 9

I_BSDV_SURFACE : = 10

I_BSDV_WEIGHT : = 11

I_BSDV_D : = 12

section height - basic (maximum) vertical dimension of a section

I_BSDV_BF : = 13

section width - basic (maximum) horizontal dimension of a section

I_BSDV_TW : = 14

web thickness / vertical wall thickness

I_BSDV_TF : = 15

flange thickness / horizontal wall thickness

I_BSDV_RA : = 16

fillet radius

I_BSDV_RI : = 17

fillet radius

I_BSDV_S : = 18

spacing - distance between section elements (double angles)

I_BSDV_ZY : = 19

section plasticity modulus - bending around Y axis

I_BSDV_ZZ : = 20

section plasticity modulus - bending around Z axis

I_BSDV_WX : = 21

section modulus for calculation of torsional stresses

I_BSDV_WY : = 22

section modulus for calculation of limit shear stresses along Y axis
I_BSDV_WZ : = 23
section modulus for calculation of limit shear stresses along Z axis
I_BSDV_GAMMA : = 24
angle between principal and main coordinate system axes
I_BSDV_IOMEGA : = 25
first moment of area
I_BSDV_P1_LENGTH : = 26
length of plate 1 in the CROSS section type
I_BSDV_P1_THICKNESS : = 27
thickness of plate 1 in the CROSS section type
I_BSDV_P2_LENGTH : = 28
length of plate 2 in the CROSS section type
I_BSDV_P2_THICKNESS : = 29
thickness of plate 2 in the CROSS section type
I_BSDV_P3_LENGTH : = 30
length of plate 3 in the CROSS section type
I_BSDV_P3_THICKNESS : = 31
thickness of plate 3 in the CROSS section type
I_BSDV_P4_LENGTH : = 32
length of plate 4 in the CROSS section type
I_BSDV_P4_THICKNESS : = 33
thickness of plate 4 in the CROSS section type
I_BSDV_BF2 : = 34
second section width (compare I_BSDV_BF)
I_BSDV_TF2 : = 35
second flange thickness (compare I_BSDV_TF)
I_BSDV_DIM1 : = 36
first dimension
I_BSDV_DIM2 : = 37
second dimension
I_BSDV_DIM3 : = 38
third dimension
I_BSDV_ANGLE1 : = 39
additional angle
I_BSDV_ANGLE2 : = 40

Anexo B – Aplicação do API do RSA

Aplicações de nível básico

Quadro B 1 – Criação do nós e das barras da estrutura.

```
Sub ap_1()
'Define robot_ap1 como uma nova aplicação do programa Robot
Dim robot_ap1 As New RobotApplication
'Coordenadas dos nós do pórtico
AX = 0
AY = 0
BX = 0
BY = 5
CX = 4
CY = 0
DX = 4
DY = 5
EX = 9
Ey = 0
FX = 9
FY = 5
'Criação dos nós com os valores de coordenadas acima referidas
robot_ap1.Project.Structure.Nodes.Create 1, AX, 0, AY
robot_ap1.Project.Structure.Nodes.Create 2, BX, 0, BY
robot_ap1.Project.Structure.Nodes.Create 3, CX, 0, CY
robot_ap1.Project.Structure.Nodes.Create 4, DX, 0, DY
robot_ap1.Project.Structure.Nodes.Create 5, EX, 0, Ey
robot_ap1.Project.Structure.Nodes.Create 6, FX, 0, FY
'Criação de barras com os valores do nós acima referidos
robot_ap1.Project.Structure.Bars.Create 1, 1, 2
robot_ap1.Project.Structure.Bars.Create 2, 2, 4
robot_ap1.Project.Structure.Bars.Create 3, 3, 4
robot_ap1.Project.Structure.Bars.Create 4, 4, 6
robot_ap1.Project.Structure.Bars.Create 5, 5, 6
Set robot_ap1 = Nothing
End Sub
```

Quadro B 2 - Criação das propriedades mecânicas, dos apoios e do material das barras da estrutura.

```
Sub ap_2()
'Define robot_ap2 como uma nova aplicação do programa Robot
Dim robot_ap2 As New RobotApplication
'Definição de um novo projeto de pórtico plano
'robot_ap2.Project.New I_PT_FRAME_2D
'Definição de um servidor de atributos de barras
'Dim Labelbars As IRobotLabel
'Criação de um objeto da classe RobotBarSectionData com as propriedades das barras
'Dim bar_sec As RobotBarSectionData
'Definição de um servidor de atributos de material
'Dim LabelMaterial As RobotLabel
'Criação de um objeto da classe RobotMaterialData com os dados do material
'Dim mat_prop As RobotMaterialData
'Definição de um servidor para apoios
'Dim LabelApoios As RobotLabel
'Definição de um objeto da classe RobotNodeSupportData com os dados dos apoios
'Dim apoios_prop As RobotNodeSupportData
'Criação de um objeto correspondente a um caso de carga simples para carga permanente
'Dim CASE_Permanente As RobotSimpleCase
'Criação de uma variável que recebe os dados de cargas associados a um determinado caso de carga
'Dim LoadREC_perm As RobotLoadRecord
'Criação de um objeto correspondente a um caso de carga simples para carga variável
```

```
'Dim CASE_Live As RobotSimpleCase
'Dim LoadREC_live As RobotLoadRecord
'Definição das propriedades mecânicas das barras, área, inércia
Set Labelbars = robot_ap2.Project.Structure.Labels.Create(I_LT_BAR_SECTION, "section")
'Instrução que define a atribuição das propriedades das barras ao objeto bar_sec
Set bar_sec = Labelbars.Data
'Definição da área de secção da barra igual a 100 cm2
bar_sec.SetValue I_BSDV_AX, 0.01
'Definição dos momentos de inércia iguais a 5000 cm4
bar_sec.SetValue I_BSDV_IX, 0.00005
bar_sec.SetValue I_BSDV_IY, 0.00005
bar_sec.SetValue I_BSDV_IZ, 0.00005
bar_sec.SetValue I_BSDV_IY, 0.00005
bar_sec.SetValue I_BSDV_IZ, 0.00005
'O servidor Labelbars é guardado na aplicação em definição
robot_ap2.Project.Structure.Labels.Store Labelbars
'Atribuição da secção definida às barras que previamente foram criadas no modelo
robot_ap2.Project.Structure.Bars.Get(1).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(3).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(4).SetLabel I_LT_BAR_SECTION, "section"
robot_ap2.Project.Structure.Bars.Get(5).SetLabel I_LT_BAR_SECTION, "section"
'Definição do material a atribuir às barras
Set LabelMaterial = robot_ap2.Project.Structure.Labels.Create(I_LT_MATERIAL, "material")
'Instrução que define a atribuição das propriedades do material ao objeto mat_prop
Set mat_prop = LabelMaterial.Data
'Definição do tipo de material, no caso betão
mat_prop.Type = I_MT_CONCRETE
'Definição do módulo de elasticidade
'Para introduzir 30000 MPa no Robot, no código deve introduzir-se 30000000000
mat_prop.E = 30000000000#
'Definição do coeficiente de Poisson
mat_prop.NU = 1 / 6
'Definição do peso volumico
mat_prop.RO = 25000
'Definição do módulo de distorção
mat_prop.Kirchoff = mat_prop.E / (2 * (1 + mat_prop.NU))
'O servidor LabelsMaterial é guardado na aplicação em definição
robot_ap2.Project.Structure.Labels.Store LabelMaterial
'O material criado é atribuído às barras previamente modeladas
robot_ap2.Project.Structure.Bars.Get(1).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(2).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(3).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(4).SetLabel I_LT_MATERIAL, "material"
robot_ap2.Project.Structure.Bars.Get(5).SetLabel I_LT_MATERIAL, "material"
'Definição de um tipo de apoio
Set LabelApoios = robot_ap2.Project.Structure.Labels.Create(I_LT_SUPPORT, "apoio_def")
Set apoios_prop = LabelApoios.Data
'Os valores abaixo permitem definir um apoio encastrado
apoios_prop.UX = 1
apoios_prop.UY = 1
apoios_prop.UZ = 1
apoios_prop.RX = 1
apoios_prop.RY = 1
apoios_prop.RZ = 1
robot_ap2.Project.Structure.Labels.Store LabelApoios
'O apoio definido é atribuído aos nós 1,3 e 5
robot_ap2.Project.Structure.Nodes.Get(1).SetLabel I_LT_SUPPORT, "apoio_def"
robot_ap2.Project.Structure.Nodes.Get(3).SetLabel I_LT_SUPPORT, "apoio_def"
robot_ap2.Project.Structure.Nodes.Get(5).SetLabel I_LT_SUPPORT, "apoio_def"
'Definição de um caso de carga permanente a para análise linear com o nome Caso 1
Set CASE_Permanente = robot_ap2.Project.Structure.Cases.CreateSimple(1, "Caso 1",
I_CN_PERMANENT, I_CAT_STATIC_LINEAR)
'Definição no programa de cálculo da carga do tipo dead load que corresponde ao peso próprio da
estrutura
CASE_Permanente.Records.New I_LRT_DEAD
'Atribuição dos dados do caso de carga permanente ao objeto LoadREC_perm
Set LoadREC_perm = CASE_Permanente.Records.Get(1)
'Definição da direção e sentido da carga permente
LoadREC_perm.SetValue I_DRV_Z, -1
```

```
'Atribuição do caso de carga a todos os elementos da estrutura
LoadREC_perm.SetValue I_DRV_ENTIRE_STRUCTURE, True
'Definição de uma sobrecarga variável
Set CASE_Live = robot_ap2.Project.Structure.Cases.CreateSimple(2, "Live", I_CN_EXPLOATATION, I_CAT_STATIC_LINEAR)
'Definição do tipo de carga uniformente distribuída associado ao caso de carga e atribuído à variável uniform
Uniform = CASE_Live.Records.New(I_LRT_BAR_UNIFORM)
'Atribuição ao objeto LoadREC_live da propriedade de carga uniforme em função
'da qual serão definidos os dados correspondentes
Set LoadREC_live = CASE_Live.Records.Get(Uniform)
'Definição do valor da carga uniforme na direção X
LoadREC_live.SetValue I_URV_PX, 0
'Definição do valor da carga uniforme na direção Y
LoadREC_live.SetValue I_URV_PZ, -10000
'Atribuição da carga à barra 2
LoadREC_live.Objects.FromText ("2")
'Definição do tipo de carga concentrada associado ao caso de carga e atribuído à variável Concentrated
Concentrated = CASE_Live.Records.New(I_LRT_NODE_FORCE)
'Atribuição ao objeto LoadREC_live da propriedade de carga uniforme em função
'da qual serão definidos os dados correspondentes
Set LoadRECLIVE = CASE_Live.Records.Get(Concentrated)
'Definição do valor da carga concentratada na direção X
LoadRECLIVE.SetValue I_URV_PX, 10000
'Atribuição da carga ao nó 2
LoadRECLIVE.Objects.FromText ("2")
'Comando para execução da análise
robot_ap2.Project.CalcEngine.Calculate
End Sub
```

Quadro B3 - Pós-processamento de resultados

```
Sub Read_results()
Dim robot_results As New RobotApplication
'Definição de uma estrutura de dados para os nós da estrutura
'Dim grupo_nos As IRobotCollection
'Definição de uma estrutura de dados para as barras da estrutura
'Dim grupo_barras As IRobotCollection
'Definição de um objeto da classe RobotNode
'Dim node As IRobotNode
'Definição de um objeto da classe RobotBar
'Dim bar As IRobotBar
'Definição de variável para esforços nas barras
'Dim force_bar As RobotBarForceData
'Definição de variável para deslocamentos nas barras
'Dim no_displacement As RobotNodeDisplacementData
'Acesso a todos os nós de uma estrutura
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n2 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n3 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n4 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n5 As Double
'Definição de variável Double para receber o valor do deslocamento de um nó
Dim deslc_X_n6 As Double
'Definição de uma variável para receber o valor do momento MY numa barra
Dim momen_y_b1 As Double
Dim momen_y_b2 As Double
Dim momen_y_b3 As Double
Dim momen_y_b4 As Double
Dim momen_y_b5 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_x_b1 As Double
Dim esf_z_b1 As Double
'Definição de uma variável para receber o valor do esforço numa barra
```

```
Dim esf_Z_b2 As Double
Dim esf_x_b2 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b3 As Double
Dim esf_x_b3 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b4 As Double
Dim esf_x_b4 As Double
'Definição de uma variável para receber o valor do esforço numa barra
Dim esf_Z_b5 As Double
Dim esf_x_b5 As Double
'Acesso a todos os nós da estrutura atribuído à estrutura de dados grupo_nos
Set grupo_nos = robot_results.Project.Structure.Nodes.GetAll
'Atribuição ao objeto node das informações do nó 1
'Atribuição ao objeto node das informações do nó 2
'Atribuição ao objeto node das informações do nó 3
'Atribuição ao objeto node das informações do nó 4
'Atribuição ao objeto node das informações do nó 5
'Atribuição ao objeto node das informações do nó 6
'Para acesso de atributos de outros nós basta mudar o argumento do nó
'corresponde ao nó
Set node = grupo_nos.Get(1)
Set node = grupo_nos.Get(2)
Set node = grupo_nos.Get(3)
Set node = grupo_nos.Get(4)
Set node = grupo_nos.Get(5)
Set node = grupo_nos.Get(6)
'Atribuição à variável NN do número do nó
NN = node.Number
'Atribuição à variável X da abcissa do nó
X = node.X
Y = node.Y
Z = node.Z
'Acesso a todas as barras da estrutura
Set grupo_barras = robot_results.Project.Structure.Bars.GetAll
'Atribuição ao objeto bar das informações da barra 1
'Atribuição ao objeto bar das informações da barra 2
'Atribuição ao objeto bar das informações da barra 3
'Atribuição ao objeto bar das informações da barra 4
'Atribuição ao objeto bar das informações da barra 5
Set bar = grupo_barras.Get(1)
Set bar = grupo_barras.Get(2)
Set bar = grupo_barras.Get(3)
Set bar = grupo_barras.Get(4)
Set bar = grupo_barras.Get(5)
xk = bar.EndNode
'Atribuição à varial NA do número da barra 1
NA = bar.Number
LA = bar.Length
'Atribuição à varial NB do número da barra 2
NB = bar.Number
LB = bar.Length
'Atribuição à varial NC do número da barra 3
Nc = bar.Number
LC = bar.Length
'Atribuição à varial ND do número da barra 4
ND = bar.Number
LD = bar.Length
'Atribuição à varial NE do número da barra 5
NE = bar.Number
LE = bar.Length
'Acesso aos deslocamentos do nó 2 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 3 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 4 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 5 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos do nó 6 para o caso de carga 2
```

```
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'A variável deslc_X_n2 recebe o deslocamento horizontal do nó 2
deslc_X_n2 = no_displacement.UX
Folha1.Cells(1, 1) = deslc_X_n2
'Acesso aos deslocamentos do nó 3 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'A variável deslc_X_n3 recebe o deslocamento horizontal do nó 3
deslc_X_n3 = no_displacement.UX
Folha1.Cells(1, 2) = deslc_X_n3
'A variável deslc_X_n4 recebe o deslocamento horizontal do nó 4
deslc_X_n4 = no_displacement.UX
Folha1.Cells(1, 3) = deslc_X_n4
'A variável deslc_X_n5 recebe o deslocamento horizontal do nó 5
deslc_X_n5 = no_displacement.UX
Folha1.Cells(1, 4) = deslc_X_n5
'A variável deslc_X_n6 recebe o deslocamento horizontal do nó 6
deslc_X_n6 = no_displacement.UX
Folha1.Cells(1, 5) = deslc_X_n6
'Acesso aos deslocamentos da barra 1 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos da barra 2 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos da bara 3 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos da barra 4 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos da barra 5 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'Acesso aos deslocamentos da barra 1 para o caso de carga 2
Set no_displacement = robot_results.Project.Structure.Results.Nodes.Displacements.Value(2, 2)
'A variável deslc_x_b1 recebe o deslocamento horizontal da barra 2
deslc_X_b1 = no_displacement.UX
Folha1.Cells(2, 1) = deslc_X_b1
'A variável deslc_x_b2 recebe o deslocamento horizontal da barra 2
deslc_X_b2 = no_displacement.UX
Folha1.Cells(2, 2) = deslc_X_b2
'A variável deslc_x_b3 recebe o deslocamento horizontal da barra 2
deslc_X_b3 = no_displacement.UX
Folha1.Cells(2, 3) = deslc_X_b3
'A variável deslc_x_b4 recebe o deslocamento horizontal da barra 2
deslc_X_b4 = no_displacement.UX
Folha1.Cells(2, 4) = deslc_X_b4
'A variável deslc_x_b5 recebe o deslocamento horizontal da barra 2
deslc_X_b5 = no_displacement.UX
Folha1.Cells(2, 5) = deslc_X_b5
'Acesso aos esforços da barra 1, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(1, 2, 1)
'A variável momen_y_b1 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b1 = force_bar.MY
Folha1.Cells(3, 1) = momen_y_b1
'A variável esf_x_b1 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b1 = force_bar.FX
Folha1.Cells(4, 1) = esf_x_b1
'A variável esf_Z_b1 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_z_b1 = force_bar.FZ
Folha1.Cells(5, 1) = esf_z_b1
'Acesso aos esforços da barra 2, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(2, 2, 1)
'A variávelmomen_y_b2 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b2 = force_bar.MY
Folha1.Cells(3, 2) = momen_y_b2
'A variável esf_x_b2 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b2 = force_bar.FX
Folha1.Cells(4, 2) = esf_x_b2
'A variável esf_Z_b2 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b2 = force_bar.FZ
Folha1.Cells(5, 2) = esf_Z_b2
'Acesso aos esforços da barra 3, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(3, 2, 1)
'A variável momen_y_b3 recebe o momento fletor MY acedido através do objeto force_bar
```

```
momen_y_b3 = force_bar.MY
Folha1.Cells(3, 3) = momen_y_b3
'A variável esf_x_b3 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b3 = force_bar.FX
Folha1.Cells(4, 3) = esf_x_b3
'A variável esf_Z_b3 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b3 = force_bar.FZ
Folha1.Cells(5, 3) = esf_Z_b3
'Acesso aos esforços da barra 4, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(4, 2, 1)
'A variável momen_y_b4 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b4 = force_bar.MY
Folha1.Cells(3, 4) = momen_y_b4
'A variável esf_x_b4 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b4 = force_bar.FX
Folha1.Cells(4, 4) = esf_x_b4
'A variável esf_Z_b4 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b4 = force_bar.FZ
Folha1.Cells(5, 4) = esf_Z_b4
'Acesso aos esforços da barra 5, para o caso de carga 2, na secção 1
Set force_bar = robot_results.Project.Structure.Results.Bars.Forces.Value(5, 2, 1)
'A variável momen_y_b5 recebe o momento fletor MY acedido através do objeto force_bar
momen_y_b5 = force_bar.MY
Folha1.Cells(3, 5) = momen_y_b5
'A variável esf_x_b5 recebe o esforço axial FX acedido através do objeto force_bar
esf_x_b5 = force_bar.FX
Folha1.Cells(4, 5) = esf_x_b5
'A variável esf_Z_b5 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_Z_b5 = force_bar.FZ
Folha1.Cells(5, 5) = esf_Z_b5
End Sub
```

Aplicações de nível intermédio

Quadro B 4 - Criação dos nós, das barras da estrutura e da ligação articulada.

```
Sub ap_1()
'Define robot_ap1 como uma nova aplicação do programa Robot
Dim robot_ap1 As New RobotApplication
num_nos = Folha2.Cells(1, 12)
num_barras = Folha2.Cells(2, 12)
'Criação dos nós com os valores de coordenadas acima referidas
For ii = 7 To num_nos + 6
 robot_ap1.Project.Structure.Nodes.Create Folha2.Cells(ii, 7), Folha2.Cells(ii, 8), 0, Folha2.Cells(ii, 9)
Next
'Criação de barras com os valores do nós acima referidos
For ii = 7 To num_barras + 6
   robot_ap1.Project.Structure.Bars.Create Folha2.Cells(ii, 11), Folha2.Cells(ii, 12), Folha2.Cells(ii, 13)
Next
'Definiçaão das propriedades de uma libertação de barra com o nome CR_Release
Set release_label = robot_ap1.Project.Structure.Labels.Create(I_LT_BAR_RELEASE, "CR_Release")
'Definição do objeto release_bar que recebe os dados da libertação
Set release_bar = release_label.Data
'Indicação de que a rotação do nó inicial da barra é contínua
release_bar.EndNode.RY = I_BERV_NOME
'Indicação de que a rotação do no final da barra é livre
release_bar.StartNode.RY = I_BERV_STD
'Atribuição da libertação à barra 2
robot_ap1.Project.Structure.Bars.Get(2).SetLabel I_LT_BAR_RELEASE, "CR_Release"
'A libertação é armazenada no programa de cálculo
robot_ap1.Project.Structure.Labels.Store release_label
Set robot_ap1 = Nothing
End Sub
```

Quadro B 5 - Modelação dos nós e das barras.

```
Sub ap_1()
'Define robot_ap1 como uma nova aplicação do programa Robot
Dim robot_ap1 As New RobotApplication
num_nos = Folha2.Cells(1, 12)
num_barras = Folha2.Cells(2, 12)
'Criação dos nós com os valores de coordenadas acima referidas
For ii = 15 To 26
 robot_ap1.Project.Structure.Nodes.Create Folha2.Cells(ii, 7), Folha2.Cells(ii, 8), 0, Folha2.Cells(ii, 9)
Next
'Criação de barras com os valores do nós acima referidos
For ii = 15 To 29
 robot_ap1.Project.Structure.Bars.Create Folha2.Cells(ii, 11), Folha2.Cells(ii, 12), Folha2.Cells(ii, 13)
Next
Set robot_ap1 = Nothing
End Sub
```

Quadro B 6 - Definição de tipo projeto e servidores de propriedades e de objetos.

```
Sub ap_2()
'Define robot_ap2 como uma nova aplicação do programa Robot
Dim robot_ap2 As New RobotApplication
'Definição de um novo projeto de pórtico plano
'robot_ap2.Project.New I_PT_FRAME_2D
'Definição de um servidor de atributos de barras
'Dim Labelbars As IRobotLabel
'Criação de um objeto da classe RobotBarSectionData com as propriedades das barras
'Dim bar_sec As RobotBarSectionData
'Definição de um servidor de atributos de material
'Dim LabelMaterial As RobotLabel
'Criação de um objeto da classe RobotMaterialData com os dados do material
'Dim mat_prop As RobotMaterialData
'Definição de um servidor para apoios
'Dim LabelApoios As RobotLabel
'Definição de um objeto da classe RobotNodeSupportData com os dados dos apoios
'Dim apoios_prop As RobotNodeSupportData
'Definição das propriedades mecânicas das barras, área, inércia
'Instrução que define a atribuição das propriedades das barras ao objeto bar_sec
'Definição da área de secção da barra igual a 100 cm2
num_barras = Folha2.Cells(2, 12)
num_nos = Folha2.Cells(1, 12)
num_seccoes = Folha2.Cells(1, 15)
```

```
num_materiais = Folha2.Cells(1, 19)
aa = 15
For I = 1 To num_seccoes + 26
   Set Labelbars = robot_ap2.Project.Structure.Labels.Create(I_LT_BAR_SECTION, Folha2.Cells(aa, 14))
   Set bar_sec = Labelbars.Data
   bar_sec.SetValue I_BSDV_AX, Folha2.Cells(aa, 16)

   'Definição dos momentos de inércia iguais a 5000 cm4
   'O servidor Labelbars é guardado na aplicação em definição

   bar_sec.SetValue I_BSDV_IX, Folha2.Cells(aa, 17)
   robot_ap2.Project.Structure.Labels.Store Labelbars

   aa = aa + 1

Next

'Atribuição da secção definida às barras que previamente foram criadas no modelo

aa = 15
For ii = 1 To num_materiais + 26

   'Definição do material a atribuir às barras
   Set LabelMaterial = robot_ap2.Project.Structure.Labels.Create(I_LT_MATERIAL, Folha2.Cells(aa, 15))
   'Instrução que define a atribuição das propriedades do material ao objeto mat_prop
   Set mat_prop = LabelMaterial.Data
   'Definição do tipo de material, no caso betão
   mat_prop.Type = I_MT_CONCRETE
   'Definição do módulo de elasticidade
   'Para introduzir 30000 MPa no Robot, no código deve introduzir-se 30000000000
   mat_prop.E = Folha2.Cells(aa, 18)
   'Definição do coeficiente de Poisson
   mat_prop.NU = Folha2.Cells(aa, 19)
   'Definição do peso volumico
   mat_prop.RO = Folha2.Cells(aa, 20)
   'Definição do módulo de distorção
   mat_prop.Kirchoff = mat_prop.E / (2 * (1 + mat_prop.NU))
   'O servidor LabelsMaterial é guardado na aplicação em definição
   robot_ap2.Project.Structure.Labels.Store LabelMaterial
   'O material criado é atribuído às barras previamente modeladas
   aa = aa + 1
Next

For ii = 15 To 29
```

```
 barra = Folha2.Cells(ii, 11)
 robot_ap2.Project.Structure.Bars.Get(barra).SetLabel I_LT_MATERIAL, Folha2.Cells(ii, 15)
 robot_ap2.Project.Structure.Bars.Get(barra).SetLabel I_LT_BAR_SECTION, Folha2.Cells(ii, 14)
Next

aa = 3
For ii = 1 To 3

   'Definição de um tipo de apoio
   Set LabelApoios = robot_ap2.Project.Structure.Labels.Create(I_LT_SUPPORT, Folha4.Cells(aa, 7))
   Set apoios_prop = LabelApoios.Data
    aa = aa + 1
Next
   'Os valores abaixo permitem definir um apoio encastrado
   aa = 8
For ii = 1 To 3
   apoios_prop.UX = Folha4.Cells(aa, 1)
   apoios_prop.UY = Folha4.Cells(aa, 2)
   apoios_prop.UZ = Folha4.Cells(aa, 3)
   apoios_prop.RX = Folha4.Cells(aa, 4)
   apoios_prop.RY = Folha4.Cells(aa, 5)
   apoios_prop.RZ = Folha4.Cells(aa, 6)
   robot_ap2.Project.Structure.Labels.Store LabelApoios
   aa = aa + 1
Next

   aa = 9
'O apoio definido é atribuído aos nós
For ii = 1 To Folha4.Cells(1, 7)
 robot_ap2.Project.Structure.Nodes.Get(Folha4.Cells(aa, 10)).SetLabel I_LT_SUPPORT,
Folha4.Cells(aa, 11)
 aa = aa + 1
Next
End Sub
```

Quadro B 7 - Definição de casos de carga e atribuição de cargas a barras.

```
Sub ap_carga()
'Define robot_carga como uma nova aplicação do programa Robot
Dim robot_carga As New RobotApplication
'Criação de um objeto correspondente a um caso de carga simples para carga permanente
'Dim CASE_Permanente As RobotSimpleCase
'Criação de uma variável que recebe os dados de cargas associados a um determinado caso de carga
'Dim LoadREC_perm As RobotLoadRecord
'Criação de um objeto correspondente a um caso de carga simples para carga variável
'Dim CASE_Live As RobotSimpleCase
'Dim LoadREC_live As RobotLoadRecord
'Definição de um caso de carga permanente a para análise linear com o nome Caso 1
Set CASE_Permanente = robot_carga.Project.Structure.Cases.CreateSimple(1, "Caso 1",
I_CN_PERMANENT, I_CAT_STATIC_LINEAR)
'Definição no programa de cálculo da carga do tipo dead load que corresponde ao peso próprio da
estrutura
CASE_Permanente.Records.New I_LRT_DEAD
'Atribuição dos dados do caso de carga permanente ao objeto LoadREC_perm
Set LoadREC_perm = CASE_Permanente.Records.Get(1)
'Definição da direção e sentido da carga permente
LoadREC_perm.SetValue I_DRV_Z, -1
'Atribuição do caso de carga a todos os elementos da estrutura
LoadREC_perm.SetValue I_DRV_ENTIRE_STRUCTURE, True
'Definição de uma sobrecarga variável
Set CASE_Live = robot_carga.Project.Structure.Cases.CreateSimple(2, "Live", I_CN_EXPLOATATION,
I_CAT_STATIC_LINEAR)
'Definição do tipo de carga uniformente distribuída associado ao caso de carga e atribuído à variável
uniform
Uniform = CASE_Live.Records.New(I_LRT_BAR_UNIFORM)
'Atribuição ao objeto LoadREC_live da propriedade de carga uniforme em função
'da qual serão definidos os dados correspondentes
Set LoadREC_live = CASE_Live.Records.Get(Uniform)
'Definição do valor da carga uniforme na direção X
LoadREC_live.SetValue I_URV_PX, 0
'Definição do valor da carga uniforme na direção Y
LoadREC_live.SetValue I_URV_PZ, -500000
'Atribuição da carga às barras
aa = 9
For ii = 1 To 6
   LoadREC_live.Objects.FromText (Folha4.Cells(aa, 12))
aa = aa + 1
Next
'Comando para execução da análise
robot_carga.Project.CalcEngine.Calculate
End Sub
```

Quadro B 8 - Obtenção dos resultados da análise.

```
Sub Read_result()
Dim robot_result As New RobotApplication
'Definição de uma estrutura de dados para os nós da estrutura
'Dim grupo_nos As IRobotCollection
'Definição de uma estrutura de dados para as barras da estrutura
'Dim grupo_barras As IRobotCollection
'Definição de um objeto da classe RobotNode
'Dim node As IRobotNode
'Definição de um objeto da classe RobotBar
'Dim bar As IRobotBar
'Definição de variável para esforços nas barras
'Dim force_bar As RobotBarForceData
'Definição de variável para deslocamentos nas barras
'Dim no_displacement As RobotNodeDisplacementData
'Acesso a todos os nós de uma estrutura
'Definição de variável Double para receber o valor do deslocamento de um nó
'Definição de uma variável para receber o valor do momento MY numa barra
'Definição de uma variável para receber o valor do esforço numa barra
num_barras = Folha2.Cells(2, 12)
num_nos = Folha2.Cells(1, 12)
Dim deslc_X As Double
Dim deslc_Y As Double
Dim momen_y As Double
Dim esf_x As Double
Dim esf_z As Double
For ii = 15 To 26
   deslc_X = Folha2.Cells(ii, 7)
   deslc_Y = Folha2.Cells(ii, 7)
Next
For ii = 15 To 29
   momen_y = Folha2.Cells(ii, 11)
   esf_x = Folha2.Cells(ii, 11)
   esf_z = Folha2.Cells(ii, 11)
Next
'Acesso a todos os nós da estrutura, atribuído à estrutura de dados grupo_nos
Set grupo_nos = robot_result.Project.Structure.Nodes.GetAll
'Atribuição ao objeto node das informações dos nós
'Para acesso de atributos de outros nós basta mudar o argumento do nó
'corresponde ao nó
For ii = 15 To 26
   Set node = grupo_nos.Get(Folha2.Cells(ii, 7))
Next
'Atribuição à variável NN do número do nó
NN = node.Number
'Atribuição à variável X da abcissa do nó
```

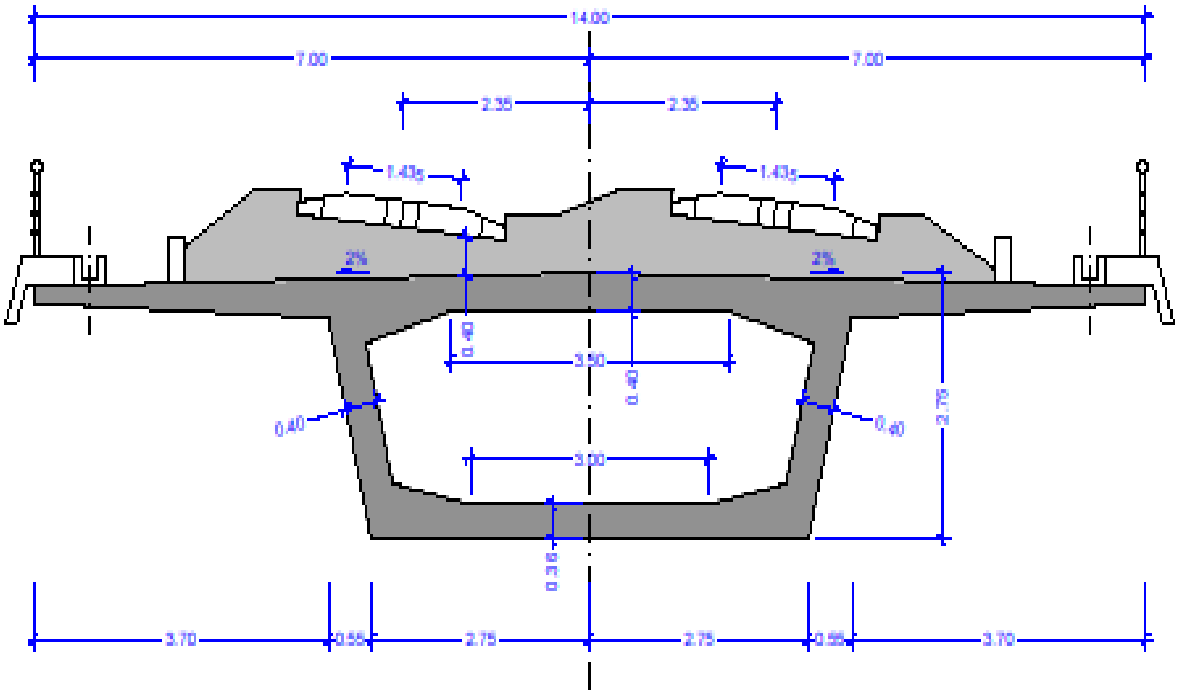
```
X = node.X
Y = node.Y
Z = node.Z
'Acesso a todas as barras da estrutura
Set grupo_barras = robot_result.Project.Structure.Bars.GetAll
'Atribuição ao objeto bar das informações da barra
For ii = 15 To num_barras - 1
   Set bar = grupo_barras.Get(Folha2.Cells(ii, 11))
   xk = bar.EndNode
Next
'Atribuição à varial NA do número da barra
'NA = bar.Number
'LA = bar.Length
'Acesso aos deslocamentos dos nós para o caso de carga
aa = 1
For ii = 15 To 26
   Set
no_displacement=robot_result.Project.Structure.Results.Nodes.Displacements.Value(Folha2.Cells(ii,
7), 2)
   'A variável deslc_X recebe o deslocamento horizontal dos nós
   deslc_X = no_displacement.UX
   Folha3.Cells(aa, 1) = deslc_X
   aa = aa + 1
Next

aa = 1
For ii = 15 To 26
   Set no_displacement =
robot_result.Project.Structure.Results.Nodes.Displacements.Value(Folha2.Cells(ii, 7), 2)
   'A variável deslc_Y recebe o deslocamento vertical dos nós
deslc_Y = no_displacement.UY
Folha3.Cells(aa, 2) = deslc_Y
   aa = aa + 1
Next
'Acesso aos esforços das barras, para o caso de carga 2, na secção 1
aa = 1
For ii = 15 To 29
  Set force_bar = robot_result.Project.Structure.Results.Bars.Forces.Value(Folha2.Cells(ii, 11), 2, 1)
  'A variável momen_y_b1 recebe o momento fletor MY acedido através do objeto force_bar
momen_y = force_bar.MY
Folha3.Cells(aa, 3) = momen_y
'A variável esf_x_b1 recebe o esforço axial FX acedido através do objeto force_bar
esf_x = force_bar.FX
Folha3.Cells(aa, 4) = esf_x
'A variável esf_Z_b1 recebe o esforço transverso Fz acedido através do objeto force_bar
esf_z = force_bar.FZ
```

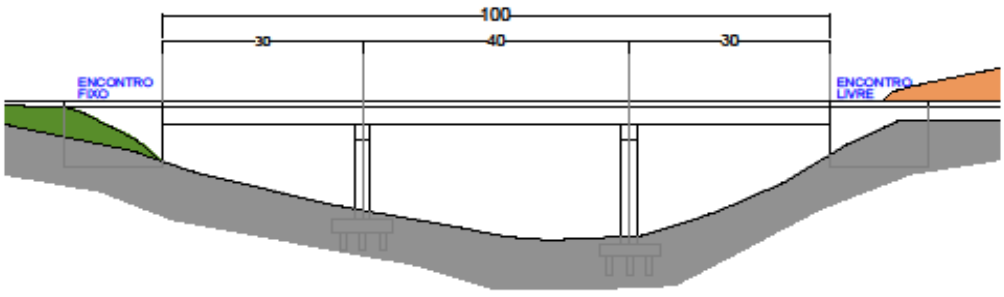
```
Folha3.Cells(aa, 5) = esf_z
aa = aa + 1
  Next
End Sub
```

Anexo C - Análise dinâmica de ponte ferroviária

Corte do Tabuleiro do viaduto



Alçado do Viaduto



Quadro C 1 - Modelação dos nós e das barras.

```
Sub ap_1()
'Define robot_ap1 como uma nova aplicação do programa Robot
Dim robot_ap1 As New RobotApplication
num_nos = Folha1.Cells(1, 12)
num_barras = Folha1.Cells(2, 12)
'Criação dos nós com os valores de coordenadas conforme a folha 1 do Excel
For ii = 7 To num_nos + 6
 robot_ap1.Project.Structure.Nodes.Create Folha1.Cells(ii, 7), Folha1.Cells(ii, 8), 0, Folha1.Cells(ii, 9)
Next
'Criação de barras com os valores do nós acima referidos
For ii = 7 To num_barras + 6
 robot_ap1.Project.Structure.Bars.Create Folha1.Cells(ii, 11), Folha1.Cells(ii, 12), Folha1.Cells(ii, 13)
Next
Set robot_ap1 = Nothing
End Sub
```

Quadro C 2 - Definição dos valores das funções de carga no tempo.

```
Sub preproc1()
Dim n_t_cases As Integer
Dim tempo As Integer
Dim aux As Double
Dim no_inf As Integer
Dim no_sup As Integer
Dim a As Integer
Dim B As Integer
Dim c As Integer
Dim bb As Double
Dim ti As Double
Dim e As Integer
Dim i As Integer
Dim j As Integer
Dim t As Double
Dim dt As Double
Dim ti_load As Double
Dim veloc As Double
Dim num_cargas As Integer
Dim num_nodes As Integer
Dim x_load As Double
Dim x_node_i As Double
Dim x_node_i_1 As Double
Dim x_node_i1 As Double
Dim k As Integer
Dim d As Integer
```

```
Dim ti_1 As Double
Dim ti1 As Double
Dim Pnode_i As Double
Dim Pnode_i_e As Double
Dim Pnode_i_d As Double
Dim x As Integer
Dim tt As Double
Dim w As Integer
Dim v As Integer
Dim caso As Integer
num_cargas = 50
tempo = 600
dt = 1 / 100
veloc = 250000 / 3600
num_nodes = 21
a = 1
B = 1
no_inf = 2
no_sup = 5
For i = 1 To num_cargas
   t = 0
   For j = 1 To tempo
   'posição de cada carga em cada instante de tempo
      Folha3.Cells(j, a) = Folha2.Cells(B, 2) + veloc * t
      t = t + dt
   Next
   B = B + 1
   a = a + 1
Next
a = 3
B = 4
c = 1
d = 1
ti = 0
For i = 1 To 1
   x_node_i = 0 ' Folha1.Cells(1, 1)
   x_node_i_1 = 0 ' Folha1.Cells(c - 1, 1)
   x_node_i1 = 5 'Folha1.Cells(2, 1)
     For k = 1 To num_cargas
      For j = 1 To tempo
         x_load = Folha3.Cells(j, k)
         If x_node_i_1 <= x_load Then
            If x_node_i >= x_load Then
               Pnode_i = (x_load - x_node_i_1) / (x_node_i1 - x_node_i)
               Folha5.Cells(j, k) = Pnode_i
            End If
```

```
        End If
      Next
    Next
    ti = 0
    For j = 1 To tempo
      aux = 0
      For e = 1 To num_cargas
        aux = aux + Folha5.Cells(j, e)
        Folha4.Cells(j, 1) = aux
        Folha5.Cells(j, e) = 0
      Next
       ti = ti + dt
      Folha4.Cells(j, 2) = ti
    Next
Next
c = 2
For i = 2 To num_nodes - 1
    x_node_i = Folha1.Cells(c, 1)
    x_node_i_1 = Folha1.Cells(c - 1, 1)
    x_node_i1 = Folha1.Cells(c + 1, 1)
      aux = 0
    For k = 1 To num_cargas
      For j = 1 To tempo
        x_load = Folha3.Cells(j, k)
        If x_node_i_1 <= x_load Then
          If x_node_i >= x_load Then
            Pnode_i = (x_load - x_node_i_1) / (x_node_i - x_node_i_1)
            Folha5.Cells(j, k) = Pnode_i
          End If
        End If
        If x_node_i <= x_load Then
          If x_node_i1 >= x_load Then
            Pnode_i = 1 - (x_load - x_node_i) / (x_node_i1 - x_node_i)
            Folha5.Cells(j, k) = Pnode_i
          End If
        End If
      Next
    Next
     ti = 0
    For j = 1 To tempo
      aux = 0
      For e = 1 To num_cargas
        aux = aux + Folha5.Cells(j, e)
        Folha4.Cells(j, a) = aux
        Folha5.Cells(j, e) = 0
      Next
```

```
    ti = ti + dt
    Folha4.Cells(j, B) = ti
  Next
  c = c + 1
  a = a + 2
  B = B + 2
Next
a = 1
B = 2

End Sub
```

Quadro C 3 - Definição da função de carga no tempo associado.

```
Sub Robot_model_1()
Dim RB As New RobotOM.RobotApplication
'Bom a funcionar'
Dim n_t_cases As Integer
Dim sacc As RobotOM.RobotSimpleCase
Set sacc = RB.Project.Structure.Cases.CreateSimple(RB.Project.Structure.Cases.FreeNumber, "TESTT", I_CN_EXPLOATATION, I_CAT_TIME_HISTORY)
Dim thfl As RobotOM.RobotTimeHistoryFunctionList
Set thfl = RB.Project.Structure.Cases.TimeHistoryFunctions
Dim path As String
Dim pathh As String
path = "D:\Dissertação\Sabino Dumbo_Dissertação_Parte prática\Cálculo Visual Basic_Excel\Visual Basic_Versão 2\Exercício 1_Modelação e análise no tempo de ponte ferroviária\Exemplos 2_"
Dim thpc As RobotOM.RobotTimeHistoryPointsCollection
Dim p As RobotOM.RobotTimeHistoryAnalysisParams
Dim mp As RobotOM.RobotTimeHistoryNewmarkAccelParams
Set p = sacc.GetAnalysisParams
p.Method = RobotOM.IRobotTimeHistoryAnalysisMethod.I_THAM_NEWMARK_ACCELERATION
Set thpc = thfl.Create()
Dim tempo As Integer
Dim aux As Double
Dim no_inf As Integer
Dim no_sup As Integer
Dim a As Integer
```

```
Dim B As Integer
Dim bb As Double
Dim ti As Double
Dim e As Integer
Dim i As Integer
Dim j As Integer
Dim t As Double
Dim dt As Double
Dim ti_load As Double
Dim veloc As Double
Dim num_cargas As Integer
Dim num_nodes As Integer
Dim x_load As Double
Dim x_node_i As Double
Dim x_node_i_1 As Double
Dim x_node_i1 As Double
Dim k As Integer
Dim d As Integer
Dim ti_1 As Double
Dim ti1 As Double
Dim Pnode_i As Double
Dim Pnode_i_e As Double
Dim Pnode_i_d As Double
Dim x As Integer
Dim tt As Double
Dim w As Integer
Dim v As Integer
Dim caso As Integer
Set p = sacc.GetAnalysisParams
p.Method =
RobotOM.IRobotTimeHistoryAnalysisMethod.I_THAM_NEWMARK_ACCELERATION
Set thpk = thfl.Create()
caso = 2
num_cargas = 50
tempo = 600
dt = 1 / 100
veloc = 250000 / 3600
```

```
num_nodes = 21
a = 1
B = 1
no_inf = 2
no_sup = 5
a = 3
B = 4
c = 1
d = 1
ti = 0
a = 1
B = 2
For i = 1 To num_nodes
   For j = 1 To tempo
      thpc.Add Folha4.Cells(j, B), Folha4.Cells(j, a) * 170
   Next
   a = a + 2
   B = B + 2
   pathh = i
   thfl.Store pathh, thpc
   p.Set caso, pathh, 1, 0
   caso = caso + 1
   thpc.SaveToFile path
   thfl.SaveToFile path
   thpc.Clear
Next
p.TimeStep = 0.002
p.Division = 1
p.End = 13
Set mp = p.MethodParams
mp.Alpha = 0.4
mp.Beta = 0.22
mp.Nonlinearity = True
mp.NonlinearParams.PDelta = True
mp.NonlinearParams.MatrixUpdateAfterEachIteration = True
mp.NonlinearParams.MatrixUpdateAfterEachSubdivision = True
mp.NonlinearParams.MaximumIterationNumberForOneIncrement = 4000
```

```
sacc.SetAnalysisParams p
Set sacc = Nothing
Set p = Nothing
Set mp = Nothing
auxx = 1
End Sub
```

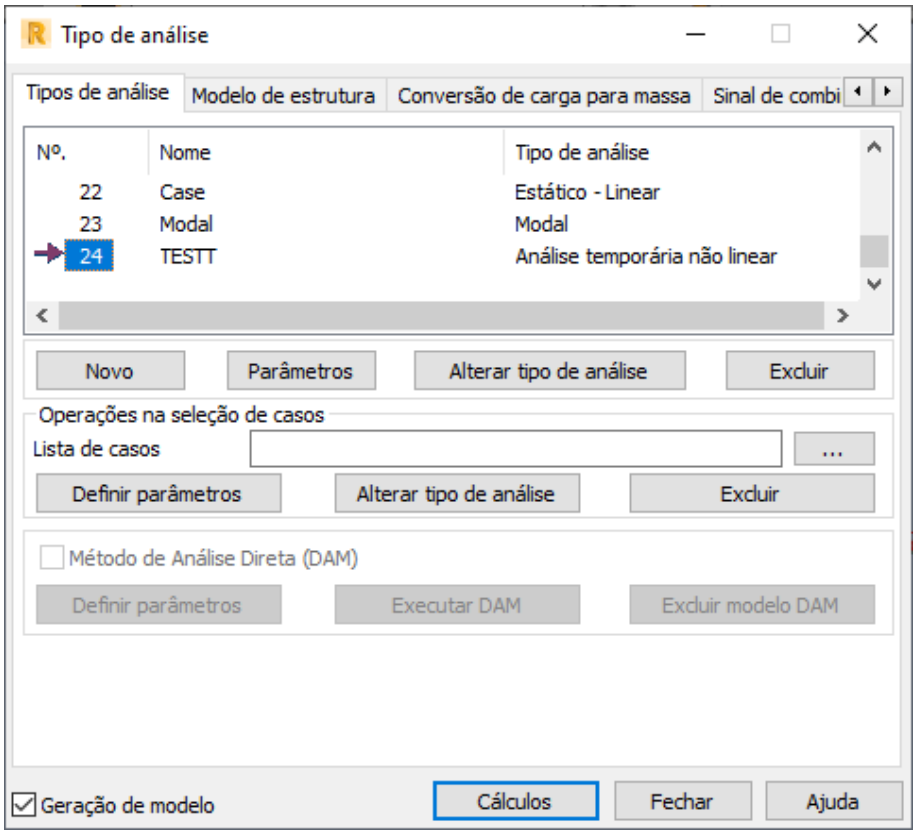
Quadro C 4 - Atribuição da carga dinâmica.

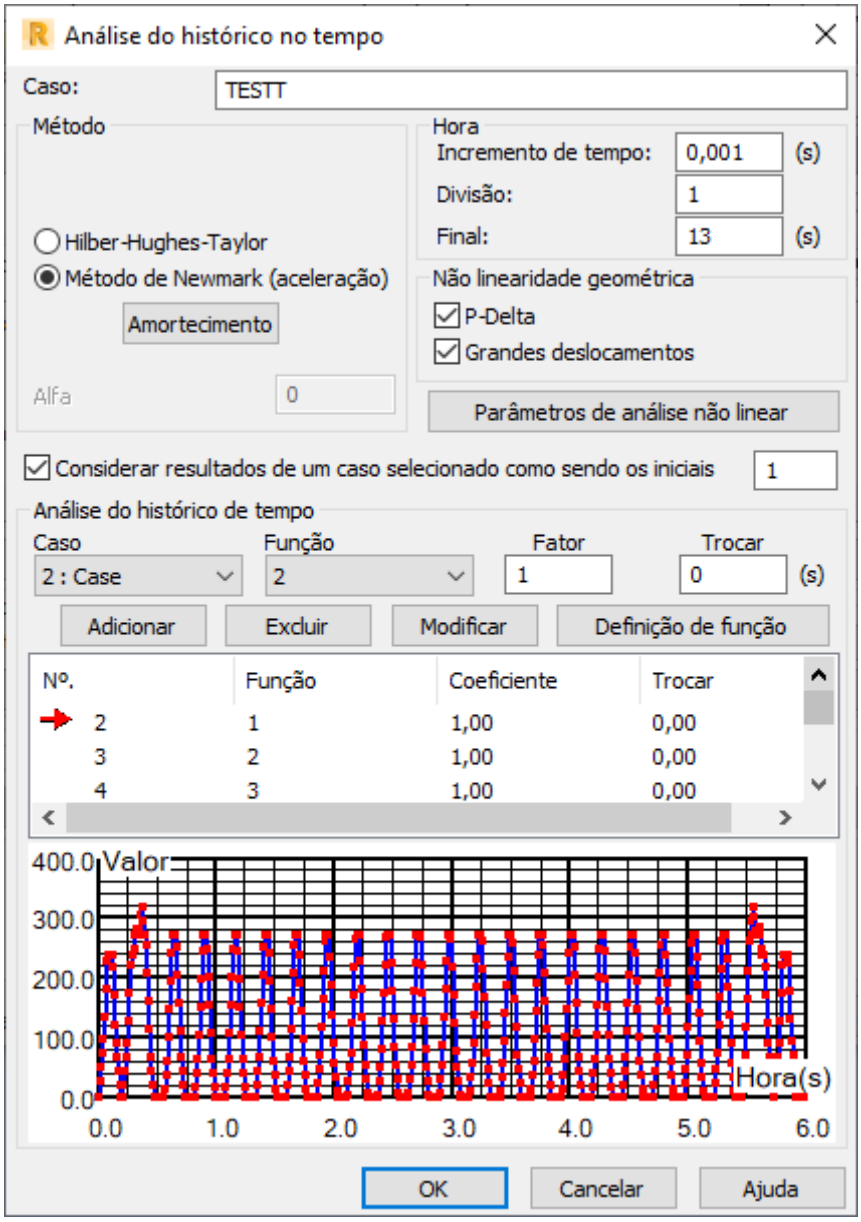
```
Sub WWRITERa()
Dim num_nodes As Integer
Dim i As Integer
'Criação de reações de apoio a partir de valores de uma tabela em folha de excel
Set robappa = New RobotApplication
'Cria a variável cas que corresponde a um simples caso de carga
'Dim cas As IRobotSimpleCase
'nod_col vai corresponder a todos os nós do modelo
'Dim nod_col As IRobotCollection
'à variável nod_col são atribuídos todos os nós do modelo
Set nod_col = robappa.Project.Structure.Nodes.GetAll()
Dim mystring As String
Dim B As Integer
'a variável recc incorpora todos os parâmetros de carga nome do caso de carga, valor de carga
'Dim recc As IRobotLoadRecord
B = 1
'Neste exemplo pretende-se criar um caso de carga para cada nó, com uma carga nodal vertical aplicada
'O ciclo seguinte corre de 1 a todos os nós
num_nodes = 21
For i = 1 To num_nodes 'nod_col.Count
  'cas recebe as caracteríticas do caso de carga, nome, tipo de carga (sobrecarga ou permenente) estática linear
  'e tipo de aplicação da carga, distribuída ou nodel
  'Dim nod As IRobotNode
  Set nod = nod_col.Get(i)
   'If nod.HasLabel(I_LT_SUPPORT) Then
  Set cas =
```

```
robappa.Project.Structure.Cases.CreateSimple(robappa.Project.Structure.Cases.FreeNumber,
"Case", I_CN_EXPLOATATION, I_CAT_STATIC_LINEAR)
   cas.Records.New I_LRT_NODE_FORCE
   Set recc = cas.Records.Get(1)
   'Conversão da variável A em string associado à variável myString
   mystring = B
   'É atribuída a variável myString ao objecto nó, sendo o nó que recebe a carga aplicada
   recc.Objects.FromText mystring
   'Definição da carga, neste caso carga vertival nodal,
   'cujo valor vai buscar à folha de excel
   recc.SetValue I_NFRV_FZ, -1000
   B = B + 1
   'End If
Next
Set recc = Nothing
Set cas = Nothing
Set robappa = Nothing
Set nod = Nothing
End Sub
```

Obtenção dos resultados da análise

Tabelas da definição da análise dinâmica




Análise do histórico no tempo
Caso:
TESTT
Método
Hilber-Hughes-Taylor
Método de Newmark (aceleração)
Amortecimento
Alfa
0
Hora
Incremento de tempo:
0,001
(s)
Divisão:
1
Final:
13
(s)
Não linearidade geométrica
P-Delta
Grandes deslocamentos
Parâmetros de análise não linear
Considerar resultados de um caso selecionado como sendo os iniciais
1
Análise do histórico de tempo
Caso
2 : Case
Função
2
Fator
1
Trocar
0
(s)
Adicionar
Excluir
Modificar
Definição de função
Nº.
Função
Coeficiente
Trocar
2
1
1,00
0,00
3
2
1,00
0,00
4
3
1,00
0,00
400.0
300.0
200.0
100.0
0.0
Valor
Hora(s)
0.0
1.0
2.0
3.0
4.0
5.0
6.0
OK
Cancelar
Ajuda

Gráficos dos resultados

Gráfico: Aceleração (AZ)

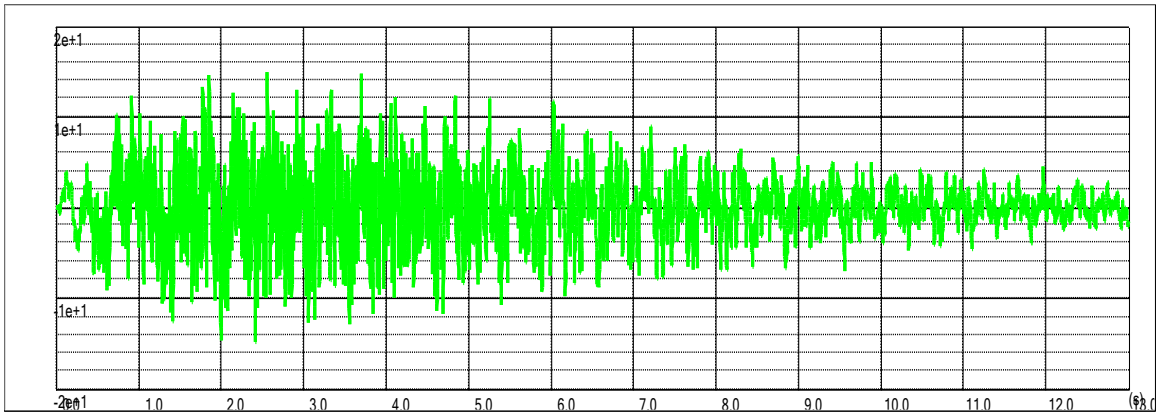


Gráfico: Aceleração (AX)

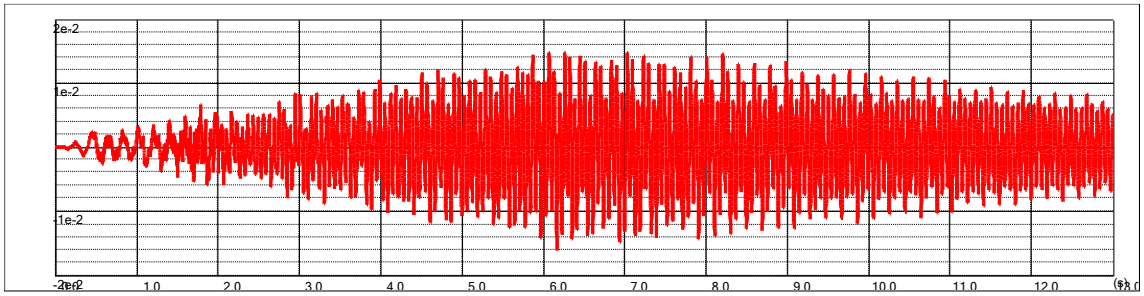


Gráfico: Deslocamento (UZ)

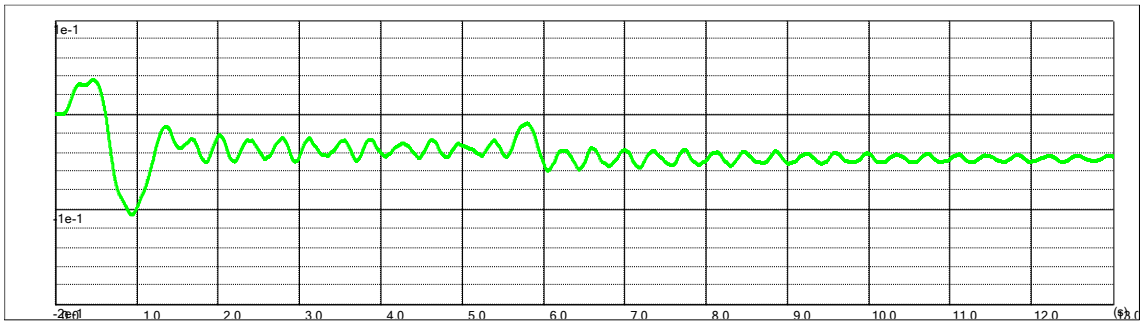
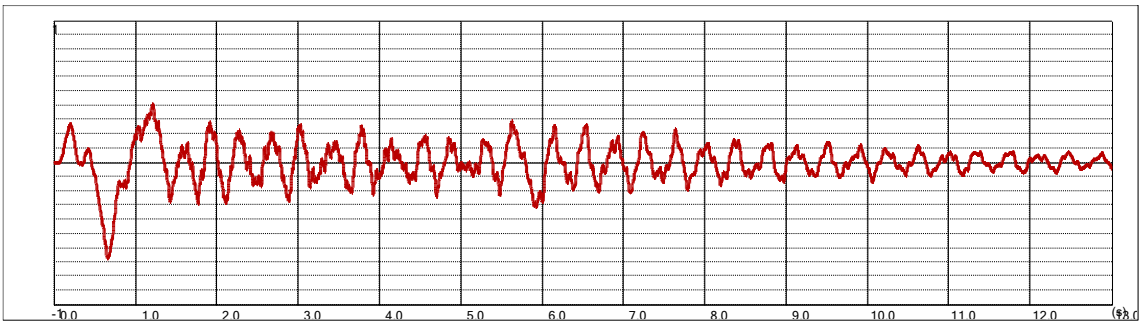


Gráfico: Velocidade (VZ)



Anexo D - Análise estrutural com OpenSees e visualização de resultados com Paraview

Exemplos de aplicação

Quadro D 1 – Bloco de código 1 (parede sujeita a cargas concentradas).

```
# Converted to openseespy by: Anurag Upadhyay, University of Utah.
# Units: N and m to follow the originally published code.
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
import matplotlib.pyplot as plt
import os
wipe()
#workbook=xlsxwriter.Workbook('Result_Elementos.xlsx')
#workbook1=xlsxwriter.Workbook('Result_Nos.xlsx')
#Elements=workbook.add_worksheet('Elements')
#Nodes_l=workbook1.add_worksheet('Nodes')
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo opensees_do ponto 4.2.2. Exemplo 3/Malha_t3.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo opensees_do ponto 4.2.2. Exemplo 3/Malha_t3.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
print(num_nos)
num_elementos=int(sheet3.cell_value(0,13))
print(num_elementos)
num_nos_ap=int(sheet5.cell_value(0,8))
print(num_nos_ap)
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
mat_tag=1
type_=str(sheet3.cell_value(0,10))
print(type_)
for i in range(0,num_nos):
```

```
node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,
2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad',     i,     int(sheet3.cell_value(i,4)),     int(sheet3.cell_value(i,3)),
int(sheet3.cell_value(i,2)),                                      int(sheet3.cell_value(i,1)),
float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31',     i,     int(sheet3.cell_value(i,1)),     int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)),
mat_tag)
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
load(11, 100., -100.)
system('BandGeneral')
# Create the constraint handler, the transformation method
constraints('Transformation')
# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer('RCM')
# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test('NormDispIncr', 1.0e-12, 100, 1)
# Create the solution algorithm, a Newton-Raphson algorithm
algorithm('Linear')
# Create the integration scheme, the LoadControl scheme using steps of 0.1
integrator('LoadControl', 1)
# Create the analysis object
analysis('Static')
analyze(1)
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
   #A=recorder('Element','-file','G.txt','-eleRange',0,num_elementos,'stress')
   #B=recorder('Element','-file','G1.txt','-ele',1,'stress')
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      StressQUAD[i]=eleResponse(i,'stress')
      Nodes[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=Nodes[a][0]
      Elements.write(b,1,Nodes[a][0])
      Elements.write(b,2,StressQUAD[a][0])
      Elements.write(b,3,StressQUAD[a][1])
      Elements.write(b,4,StressQUAD[a][2])
      Elements.write(b,5,tipo_elemento)
      Elements.write(b,0,a)
      Elements.write(b+1,1,Nodes[a][1])
      Elements.write(b+1,2,StressQUAD[a][3])
```

```
        Elements.write(b+1,3,StressQUAD[a][4])
        Elements.write(b+1,4,StressQUAD[a][5])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,Nodes[a][2])
        Elements.write(b+2,2,StressQUAD[a][6])
        Elements.write(b+2,3,StressQUAD[a][7])
        Elements.write(b+2,4,StressQUAD[a][9])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        Elements.write(b+3,1,Nodes[a][3])
        Elements.write(b+3,0,a)
        Elements.write(b+3,2,StressQUAD[a][9])
        Elements.write(b+3,3,StressQUAD[a][10])
        Elements.write(b+3,4,StressQUAD[a][11])
        Elements.write(b+3,5,tipo_elemento)
        b=b+4
    if tipo_elemento==5:
        StressTRI[i]=eleResponse(i,'stress')
        print(StressTRI[0])
        NodesTRI[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=NodesTRI[a][0]
        Elements.write(b,1,NodesTRI[a][0])
        Elements.write(b,2,StressTRI[a][0])
        Elements.write(b,3,StressTRI[a][0])
        Elements.write(b,4,StressTRI[a][0])
        Elements.write(b,3,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,NodesTRI[a][1])
        Elements.write(b+1,2,StressTRI[a][0])
        Elements.write(b+1,3,StressTRI[a][0])
        Elements.write(b+1,4,StressTRI[a][0])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
          a=a+1
Elements.write(0,6,"Numero_linhas")
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
    NDSP[i]=nodeDisp(i)
    Nodes_l.write(i+1,0,i)
    Nodes_l.write(i+1,1,NDSP[i][0])
    Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
print(NDSP)
wb_elem.save('Resultados.xls')
```

Quadro D 2 - Bloco de código 2 (parede sujeita a cargas concentradas).

```
import xlrd
import openpyxl
from decimal import Decimal
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
loc = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo
opensees_do ponto 4.2.2. Exemplo 3/Malha_t3.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
wb_stress=Workbook()
no_atress=wb_stress.add_sheet('No_stress')
loc1 = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo
opensees_do ponto 4.2.2. Exemplo 3/Resultados.xls")
wb_r=xlrd.open_workbook(loc1)
elem_result=wb_r.sheet_by_index(0)
node_result=wb_r.sheet_by_index(1)
num_elem_dim=int(elem_result.cell_value(0,7))
num_node_dim=int(node_result.cell_value(0,4))
sx_node=np.zeros((num_node_dim,5),dtype=float)
num_mnode=np.zeros((num_node_dim,1),dtype=int)
counter_i=1
for i in range(0,num_node_dim):
   aux_x=0
   aux_y=0
   aux_xy=0
   counter=1
   num_node=int(node_result.cell_value(counter_i,0))
   num_mnode[i][0]=num_node
   for j in range(1,num_elem_dim):
      num_elem=int(elem_result.cell_value(j,1))
      if num_elem==num_node:
         sx_elem=float(elem_result.cell_value(j,2))
         aux_x=aux_x+sx_elem
         sy_elem=float(elem_result.cell_value(j,3))
         aux_y=aux_y+sy_elem
         sxy_elem=float(elem_result.cell_value(j,4))
         aux_xy=aux_xy+sxy_elem
         counter=counter+1
   num_mnode[i][0]=num_node
   sx_node[i][0]=aux_x/(counter-1)
   sx_node[i][1]=aux_y/(counter-1)
   sx_node[i][2]=aux_xy/(counter-1)
   sx_node[i][3]=float(node_result.cell_value(counter_i,1))
   sx_node[i][4]=float(node_result.cell_value(counter_i,2))
   counter_i=counter_i+1
    print(sx_node[10][0])
f=open("malha_t3.vtk",'w')
f.write("# vtk DataFile Version 2.0\n")
```

```
f.write("untitled, Created by Gmsh 4.11.0\n")
f.write("ASCII\n")
#f.write("DATASET POLYDATA\n")
f.write("DATASET UNSTRUCTURED_GRID\n")
points=int(sheet1.cell_value(0,5))
size_vertice=2*points
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
size_elementos=int(sheet3.cell_value(0,0))+1
for i in range(1,num_elementos):
   size_elementos=size_elementos+int(sheet3.cell_value(i,0))+1
print(size_elementos)
f.write("POINTS %d float\n"%points)
for i in range(0,points):
   x=float(sheet1.cell_value(i,1))
   y=float(sheet1.cell_value(i,2))
   if abs(x)<0.000001:
      x=0
   if abs(y)<0.000001:
      y=0
  # print(x)
   #f.write("%5.4f %5.4f 0\n"%x%y)
   f.write("{} {} 0\n".format(x,y))
f.write("\n".format(x,y))

#f.write("VERTICES {} {}\n".format(points,size_vertice))
#for i in range(0,points):
#    x=1
#    y=int(float(sheet2.cell_value(i,0))-1)
#    print(x)
   #f.write("%5.4f %5.4f 0\n"%x%y)
#    f.write("{} {}\n".format(x,y))
f.write("CELLS {} {}\n".format(num_elementos,size_elementos))
for i in range(0,num_elementos):
   type_elem=sheet3.cell_value(i,14)
   if type_elem==5:
      x=int(sheet3.cell_value(i,15))
      y=int(sheet3.cell_value(i,1))
      z=int(sheet3.cell_value(i,2))
      t=int(sheet3.cell_value(i,3))
      f.write("{} {} {} {}\n".format(x,z,y,t))
  if type_elem==8:
      x=int(sheet3.cell_value(i,15))
      y=int(sheet3.cell_value(i,1))
      z=int(sheet3.cell_value(i,2))
      t=int(sheet3.cell_value(i,3))
      w=int(sheet3.cell_value(i,4))
      f.write("{} {} {} {} {}\n".format(x,z,y,t,w))
f.write("\n")
f.write("CELL_TYPES %d\n"%num_elementos)
for i in range(0,num_elementos):
   x=int(sheet3.cell_value(i,14))
   f.write("%d\n"%x)
f.write("\n")
f.write("POINT_DATA %d\n"%points)
f.write("FIELD FieldData 4\n")
f.write("Displacement 3 %d float\n"%points)
```

```
for i in range(0,points):
   x=float(sx_node[i][3])
   y=float(sx_node[i][4])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
   ids="%20"
   pontos=int(points)
f.write("Displacement{}Magnitude 1 {} float\n".format(ids,pontos))
for i in range(0,points):
   x=float(sx_node[i][3])
   y=float(sx_node[i][4])
   magnitude=np.sqrt(x*x+y*y)
   f.write("%6.4f\n"%magnitude)
   f.write("Stress_H 3 %d float\n"%points)
for i in range(0,points):
   x=float(sx_node[i][0])
   y=float(sx_node[i][2])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
   f.write("Stress_V 3 %d float\n"%points)
for i in range(0,points):
   x=float(sx_node[i][2])
   y=float(sx_node[i][1])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
   type_data='SX'#str(sheet1.cell_value(0,6))
#.write("POINT_DATA %d\n"%points)
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][0]#float(sheet1.cell_value(i,7))
   f.write("%6.4f\n"%x)
type_data='SY'#str(sheet1.cell_value(0,8))
#f.write("POINT_DATA %d\n"%points)
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][1]#float(sheet1.cell_value(i,9))
   f.write("%6.4f\n"%x)
type_data='SXY'#str(sheet1.cell_value(0,10))
#f.write("POINT_DATA %d\n"%points)
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][2]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
type_data='ux'#str(sheet1.cell_value(0,10))
#f.write("POINT_DATA %d\n"%points)sx_no
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][3]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
type_data='uy'#str(sheet1.cell_value(0,10))
#f.write("POINT_DATA %d\n"%points)
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
```

```
for i in range(0,points):
   x=sx_node[i][4]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
f.write("VECTORS vector float\n")
for i in range(0,points):
   x=sx_node[i][3]
   y=sx_node[i][4]
   z=0
   f.write("{} {} {}\n".format(x,y,z))
f.write("\n")
f.close()
```

Quadro D 3 - Bloco de código 3 (parede sujeita a cargas concentradas).

```
# Converted to openseespy by: Anurag Upadhyay, University of Utah.
# Units: N and m to follow the originally published code.
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
import matplotlib.pyplot as plt
import os
wipe()
#workbook=xlsxwriter.Workbook('Result_Elementos.xlsx')
#workbook1=xlsxwriter.Workbook('Result_Nos.xlsx')
#Elements=workbook.add_worksheet('Elements')
#Nodes_l=workbook1.add_worksheet('Nodes')
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo
opensees_do ponto 4.2.2. Exemplo 3/Malha_t3.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/Meus cálculos openseespy/Cálculo
opensees_do ponto 4.2.2. Exemplo 3")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
print(num_nos)
num_elementos=int(sheet3.cell_value(0,13))
print(num_elementos)
num_nos_ap=int(sheet5.cell_value(0,8))
print(num_nos_ap)
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
```

```
mat_tag=1
type_=str(sheet3.cell_value(0,10))
print(type_)
for i in range(0,num_nos):

node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,
2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad', i, int(sheet3.cell_value(i,4)), int(sheet3.cell_value(i,3)),
int(sheet3.cell_value(i,2)), int(sheet3.cell_value(i,1)),
float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31', i, int(sheet3.cell_value(i,1)), int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)),
mat_tag)
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
opsv.plot_model()
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
load(11, 100., -100.)
system('BandGeneral')
# Create the constraint handler, the transformation method
constraints('Transformation')
# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer('RCM')
# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test('NormDispIncr', 1.0e-12, 100, 1)
# Create the solution algorithm, a Newton-Raphson algorithm
algorithm('Linear')
# Create the integration scheme, the LoadControl scheme using steps of 0.1
integrator('LoadControl', 1)
# Create the analysis object
analysis('Static')
analyze(1)
opsv.plot_model()
plt.axis('equal')
# plt.figure()
opsv.plot_loads_2d()
# - plot deformation
opsv.plot_defo(unDefoFlag=1)
plt.axis('equal')
# get values at OpenSees nodes
sig_out = opsv.sig_out_per_node()

# !!! select from sig_out: e.g. vmises
# j, jstr = 0, 'sxx'
j, jstr = 1, 'syy'
# j, jstr = 2, 'sxy'
# j, jstr = 3, 'vmis'
# j, jstr = 4, 's1'
# j, jstr = 5, 's2'
# j, jstr = 6, 'alfa'
```

```
nds_val = sig_out[:, j]
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
   #A=recorder('Element','-file','G.txt','-eleRange',0,num_elementos,'stress')
   #B=recorder('Element','-file','G1.txt','-ele',1,'stress')
   tipo_elemento=int(sheet3.cell_value(i,14))
      if tipo_elemento==8:
      StressQUAD[i]=eleResponse(i,'stress')
      Nodes[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=Nodes[a][0]
      Elements.write(b,1,Nodes[a][0])
      Elements.write(b,2,StressQUAD[a][0])
      Elements.write(b,3,StressQUAD[a][1])
      Elements.write(b,4,StressQUAD[a][2])
      Elements.write(b,5,tipo_elemento)
      Elements.write(b,0,a)
      Elements.write(b+1,1,Nodes[a][1])
      Elements.write(b+1,2,StressQUAD[a][3])
      Elements.write(b+1,3,StressQUAD[a][4])
      Elements.write(b+1,4,StressQUAD[a][5])
      Elements.write(b+1,5,tipo_elemento)
      Elements.write(b+1,0,a)
      Elements.write(b+2,1,Nodes[a][2])
      Elements.write(b+2,2,StressQUAD[a][6])
      Elements.write(b+2,3,StressQUAD[a][7])
      Elements.write(b+2,4,StressQUAD[a][9])
      Elements.write(b+2,5,tipo_elemento)
      Elements.write(b+2,0,a)
      Elements.write(b+3,1,Nodes[a][3])
      Elements.write(b+3,0,a)
      Elements.write(b+3,2,StressQUAD[a][9])
      Elements.write(b+3,3,StressQUAD[a][10])
      Elements.write(b+3,4,StressQUAD[a][11])
      Elements.write(b+3,5,tipo_elemento)
      b=b+4
   if tipo_elemento==5:
      StressTRI[i]=eleResponse(i,'stress')
      print(StressTRI[0])
      NodesTRI[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=NodesTRI[a][0]
      Elements.write(b,1,NodesTRI[a][0])
      Elements.write(b,2,StressTRI[a][0])
      Elements.write(b,3,StressTRI[a][0])
      Elements.write(b,4,StressTRI[a][0])
      Elements.write(b,3,tipo_elemento)
      Elements.write(b,0,a)
      Elements.write(b+1,1,NodesTRI[a][1])
      Elements.write(b+1,2,StressTRI[a][0])
      Elements.write(b+1,3,StressTRI[a][0])
      Elements.write(b+1,4,StressTRI[a][0])
```

```
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
        a=a+1
Elements.write(0,6,"Numero_linhas")
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
   NDSP[i]=nodeDisp(i)
   Nodes_l.write(i+1,0,i)
   Nodes_l.write(i+1,1,NDSP[i][0])
   Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
print(NDSP)
wb_elem.save('Resultados.xls')
```

Quadro D 4 - Bloco de código 4 (malha triangular).

```
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
import matplotlib.pyplot as plt
import os
wipe()
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t4/Malha_t4.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t4")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
num_nos_ap=int(sheet5.cell_value(0,8))
```

```
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
mat_tag=1
type_=str(sheet3.cell_value(0,10))
for i in range(0,num_nos):

node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,
2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad',     i,     int(sheet3.cell_value(i,1)),     int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),                                    int(sheet3.cell_value(i,4)),
float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31',     i,     int(sheet3.cell_value(i,1)),     int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)),
mat_tag)
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
opsv.plot_model()
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
load(41, 0., -100.)
load(42, 0., -100.)
system('BandGeneral')
constraints('Transformation')
numberer('RCM')
test('NormDispIncr', 1.0e-12, 100, 1)
algorithm('Linear')
integrator('LoadControl', 1)
analysis('Static')
analyze(1)
opsv.plot_model()
plt.axis('equal')
opsv.plot_loads_2d()
opsv.plot_defo(unDefoFlag=1)
plt.axis('equal')
sig_out = opsv.sig_out_per_node()
j, jstr = 1, 'syy'
nds_val = sig_out[:, j]
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      StressQUAD[i]=eleResponse(i,'stress')
      Nodes[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=Nodes[a][0]
      Elements.write(b,1,Nodes[a][0])
      Elements.write(b,2,StressQUAD[a][0])
      Elements.write(b,3,StressQUAD[a][1])
```

```
        Elements.write(b,4,StressQUAD[a][2])
        Elements.write(b,5,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,Nodes[a][1])
        Elements.write(b+1,2,StressQUAD[a][3])
        Elements.write(b+1,3,StressQUAD[a][4])
        Elements.write(b+1,4,StressQUAD[a][5])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,Nodes[a][2])
        Elements.write(b+2,2,StressQUAD[a][6])
        Elements.write(b+2,3,StressQUAD[a][7])
        Elements.write(b+2,4,StressQUAD[a][9])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        Elements.write(b+3,1,Nodes[a][3])
        Elements.write(b+3,0,a)
        Elements.write(b+3,2,StressQUAD[a][9])
        Elements.write(b+3,3,StressQUAD[a][10])
        Elements.write(b+3,4,StressQUAD[a][11])
        Elements.write(b+3,5,tipo_elemento)
        b=b+4
    if tipo_elemento==5:
        StressTRI[i]=eleResponse(i,'stress')
        NodesTRI[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=NodesTRI[a][0]
        Elements.write(b,1,NodesTRI[a][0])
        Elements.write(b,2,StressTRI[a][0])
        Elements.write(b,3,StressTRI[a][0])
        Elements.write(b,4,StressTRI[a][0])
        Elements.write(b,3,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,NodesTRI[a][1])
        Elements.write(b+1,2,StressTRI[a][0])
        Elements.write(b+1,3,StressTRI[a][0])
        Elements.write(b+1,4,StressTRI[a][0])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
    a=a+1
Elements.write(0,6,"Numero_linhas")
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
    NDSP[i]=nodeDisp(i)
    Nodes_l.write(i+1,0,i)
    Nodes_l.write(i+1,1,NDSP[i][0])
    Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
```

```
wb_elem.save('Resultados.xls')
print("Análise_finalizada")
```

Quadro D 5 - Bloco de código 5 (malha triangular).

```
import xlrd
import openpyxl
from decimal import Decimal
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t4/Malha_t4.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
wb_stress=Workbook()
no_atress=wb_stress.add_sheet('No_stress')
loc1 = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t4/Resultados.xls")
wb_r=xlrd.open_workbook(loc1)
elem_result=wb_r.sheet_by_index(0)
node_result=wb_r.sheet_by_index(1)
num_elem_dim=int(elem_result.cell_value(0,7))
num_node_dim=int(node_result.cell_value(0,4))
sx_node=np.zeros((num_node_dim,5),dtype=float)
num_mnode=np.zeros((num_node_dim,1),dtype=int)
counter_i=1
for i in range(0,num_node_dim):
    aux_x=0
    aux_y=0
    aux_xy=0
    counter=1
    num_node=int(node_result.cell_value(counter_i,0))
    num_mnode[i][0]=num_node
    for j in range(1,num_elem_dim):
        num_elem=int(elem_result.cell_value(j,1))
        if num_elem==num_node:
            sx_elem=float(elem_result.cell_value(j,2))
            aux_x=aux_x+sx_elem
            sy_elem=float(elem_result.cell_value(j,3))
            aux_y=aux_y+sy_elem
            sxy_elem=float(elem_result.cell_value(j,4))
            aux_xy=aux_xy+sxy_elem
            counter=counter+1
    num_mnode[i][0]=num_node
    sx_node[i][0]=aux_x/(counter-1)
    sx_node[i][1]=aux_y/(counter-1)
    sx_node[i][2]=aux_xy/(counter-1)
    sx_node[i][3]=float(node_result.cell_value(counter_i,1))
    sx_node[i][4]=float(node_result.cell_value(counter_i,2))
```

```
    counter_i=counter_i+1
f=open("Resultados_malha_t4.vtk",'w')
f.write("# vtk DataFile Version 2.0\n")
f.write("untitled, Created by Gmsh 4.11.0\n")
f.write("ASCII\n")
f.write("DATASET UNSTRUCTURED_GRID\n")
points=int(sheet1.cell_value(0,5))
size_vertice=2*points
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
size_elementos=int(sheet3.cell_value(0,0))+1
for i in range(1,num_elementos):
    size_elementos=size_elementos+int(sheet3.cell_value(i,0))+1
f.write("POINTS %d float\n"%points)
for i in range(0,points):
    x=float(sheet1.cell_value(i,1))
    y=float(sheet1.cell_value(i,2))
    if abs(x)<0.000001:
        x=0
    if abs(y)<0.000001:
        y=0
    f.write("{} {} 0\n".format(x,y))
f.write("\n".format(x,y))
f.write("CELLS {} {}\n".format(num_elementos,size_elementos))
for i in range(0,num_elementos):
    type_elem=sheet3.cell_value(i,14)
    if type_elem==5:
        x=int(sheet3.cell_value(i,15))
        y=int(sheet3.cell_value(i,1))
        z=int(sheet3.cell_value(i,2))
        t=int(sheet3.cell_value(i,3))
        f.write("{} {} {} {}\n".format(x,z,y,t))
    if type_elem==8:
        x=int(sheet3.cell_value(i,15))
        y=int(sheet3.cell_value(i,1))
        z=int(sheet3.cell_value(i,2))
        t=int(sheet3.cell_value(i,3))
        w=int(sheet3.cell_value(i,4))
        f.write("{} {} {} {} {}\n".format(x,z,y,t,w))
f.write("\n")
f.write("CELL_TYPES %d\n"%num_elementos)
for i in range(0,num_elementos):
    x=int(sheet3.cell_value(i,14))
    f.write("%d\n"%x)
f.write("\n")
f.write("POINT_DATA %d\n"%points)
f.write("FIELD FieldData 4\n")
f.write("Displacement 3 %d float\n"%points)
for i in range(0,points):
    x=float(sx_node[i][3])
    y=float(sx_node[i][4])
    z=0
    f.write("{} {} {}\n".format(x,y,z))
ids="%20"
pontos=int(points)
f.write("Displacement{}Magnitude 1 {} float\n".format(ids,pontos))
for i in range(0,points):
```

```
    x=float(sx_node[i][3])
    y=float(sx_node[i][4])
    magnitude=np.sqrt(x*x+y*y)
    f.write("%6.4f\n"%magnitude)
f.write("Stress_H 3 %d float\n"%points)
for i in range(0,points):
    x=float(sx_node[i][0])
    y=float(sx_node[i][2])
    z=0
    f.write("{} {} {}\n".format(x,y,z))
f.write("Stress_V 3 %d float\n"%points)
for i in range(0,points):
    x=float(sx_node[i][2])
    y=float(sx_node[i][1])
    z=0
    f.write("{} {} {}\n".format(x,y,z))
type_data='SX'#str(sheet1.cell_value(0,6))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
    x=sx_node[i][0]#float(sheet1.cell_value(i,7))
    f.write("%6.4f\n"%x)
type_data='SY'#str(sheet1.cell_value(0,8))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
    x=sx_node[i][1]#float(sheet1.cell_value(i,9))
    f.write("%6.4f\n"%x)
type_data='SXY'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
    x=sx_node[i][2]#float(sheet1.cell_value(i,11))
    f.write("%6.4f\n"%x)
type_data='ux'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
    x=sx_node[i][3]#float(sheet1.cell_value(i,11))
    f.write("%6.4f\n"%x)
type_data='uy'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
    x=sx_node[i][4]#float(sheet1.cell_value(i,11))
    f.write("%6.4f\n"%x)
f.write("VECTORS vector float\n")
for i in range(0,points):
    x=sx_node[i][3]
    y=sx_node[i][4]
    z=0
    f.write("{} {} {}\n".format(x,y,z))
f.write("\n")
f.close()
print("Análise_finalizada")
```

Quadro D 6 - Bloco de código 6 (malha triangular).

```
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
import matplotlib.pyplot as plt
import os
wipe()
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t4/Malha_t4.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t4/Malha_t4.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
num_nos_ap=int(sheet5.cell_value(0,8))
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
mat_tag=1
type_=str(sheet3.cell_value(0,10))
for i in range(0,num_nos):

node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,
2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad', i, int(sheet3.cell_value(i,1)), int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)), int(sheet3.cell_value(i,4)),
float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31', i, int(sheet3.cell_value(i,1)), int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)),
mat_tag)
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
opsv.plot_model()
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
```

```
load(41, 0., -100.)
load(42, 0., -100.)
system('BandGeneral')
constraints('Transformation')
numberer('RCM')
test('NormDispIncr', 1.0e-12, 100, 1)
algorithm('Linear')
integrator('LoadControl', 1)
analysis('Static')
analyze(1)
opsv.plot_model()
plt.axis('equal')
opsv.plot_loads_2d()
opsv.plot_defo(unDefoFlag=1)
plt.axis('equal')
sig_out = opsv.sig_out_per_node()
j, jstr = 1, 'syy'
nds_val = sig_out[:, j]
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      StressQUAD[i]=eleResponse(i,'stress')
      Nodes[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=Nodes[a][0]
      Elements.write(b,1,Nodes[a][0])
      Elements.write(b,2,StressQUAD[a][0])
      Elements.write(b,3,StressQUAD[a][1])
      Elements.write(b,4,StressQUAD[a][2])
      Elements.write(b,5,tipo_elemento)
      Elements.write(b,0,a)
      Elements.write(b+1,1,Nodes[a][1])
      Elements.write(b+1,2,StressQUAD[a][3])
      Elements.write(b+1,3,StressQUAD[a][4])
      Elements.write(b+1,4,StressQUAD[a][5])
      Elements.write(b+1,5,tipo_elemento)
      Elements.write(b+1,0,a)
      Elements.write(b+2,1,Nodes[a][2])
      Elements.write(b+2,2,StressQUAD[a][6])
      Elements.write(b+2,3,StressQUAD[a][7])
      Elements.write(b+2,4,StressQUAD[a][9])
      Elements.write(b+2,5,tipo_elemento)
      Elements.write(b+2,0,a)
      Elements.write(b+3,1,Nodes[a][3])
      Elements.write(b+3,0,a)
      Elements.write(b+3,2,StressQUAD[a][9])
      Elements.write(b+3,3,StressQUAD[a][10])
      Elements.write(b+3,4,StressQUAD[a][11])
      Elements.write(b+3,5,tipo_elemento)
      b=b+4
   if tipo_elemento==5:
```

```
        StressTRI[i]=eleResponse(i,'stress')
        NodesTRI[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=NodesTRI[a][0]
        Elements.write(b,1,NodesTRI[a][0])
        Elements.write(b,2,StressTRI[a][0])
        Elements.write(b,3,StressTRI[a][0])
        Elements.write(b,4,StressTRI[a][0])
        Elements.write(b,3,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,NodesTRI[a][1])
        Elements.write(b+1,2,StressTRI[a][0])
        Elements.write(b+1,3,StressTRI[a][0])
        Elements.write(b+1,4,StressTRI[a][0])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
    a=a+1
Elements.write(0,6,"Numero_linhas")
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
    NDSP[i]=nodeDisp(i)
    Nodes_l.write(i+1,0,i)
    Nodes_l.write(i+1,1,NDSP[i][0])
    Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
wb_elem.save('Resultados.xls')
print("Análise_finalizada")
```

Quadro D 7 - Bloco de código 7 (malha quadrada).

```
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
```

```
import matplotlib.pyplot as plt
import os
wipe()
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t10/Malha_t10.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t10/Malha_t10.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
num_nos_ap=int(sheet5.cell_value(0,8))
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
mat_tag=1
type_=str(sheet3.cell_value(0,10))
for i in range(0,num_nos):

node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad', i, int(sheet3.cell_value(i,1)), int(sheet3.cell_value(i,2)), int(sheet3.cell_value(i,3)), int(sheet3.cell_value(i,4)), float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31', i, int(sheet3.cell_value(i,1)), int(sheet3.cell_value(i,2)), int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
```

```
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
opsv.plot_model()
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
load(54, 0., -100.)
load(56, 0., -100.)
system('BandGeneral')
constraints('Transformation')
numberer('RCM')
test('NormDispIncr', 1.0e-12, 100, 1)
algorithm('Linear')
integrator('LoadControl', 1)
analysis('Static')
analyze(1)
opsv.plot_model()
plt.axis('equal')
opsv.plot_loads_2d()
opsv.plot_defo(unDefoFlag=1)
plt.axis('equal')
sig_out = opsv.sig_out_per_node()
j, jstr = 1, 'syy'
nds_val = sig_out[:, j]
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      StressQUAD[i]=eleResponse(i,'stress')
      Nodes[i]=eleNodes(i)
      Elements.write(b,0,a)
      aux=Nodes[a][0]
      Elements.write(b,1,Nodes[a][0])
```

```
        Elements.write(b,2,StressQUAD[a][0])
        Elements.write(b,3,StressQUAD[a][1])
        Elements.write(b,4,StressQUAD[a][2])
        Elements.write(b,5,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,Nodes[a][1])
        Elements.write(b+1,2,StressQUAD[a][3])
        Elements.write(b+1,3,StressQUAD[a][4])
        Elements.write(b+1,4,StressQUAD[a][5])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,Nodes[a][2])
        Elements.write(b+2,2,StressQUAD[a][6])
        Elements.write(b+2,3,StressQUAD[a][7])
        Elements.write(b+2,4,StressQUAD[a][9])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        Elements.write(b+3,1,Nodes[a][3])
        Elements.write(b+3,0,a)
        Elements.write(b+3,2,StressQUAD[a][9])
        Elements.write(b+3,3,StressQUAD[a][10])
        Elements.write(b+3,4,StressQUAD[a][11])
        Elements.write(b+3,5,tipo_elemento)
        b=b+4
    if tipo_elemento==5:
        StressTRI[i]=eleResponse(i,'stress')
        NodesTRI[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=NodesTRI[a][0]
        Elements.write(b,1,NodesTRI[a][0])
        Elements.write(b,2,StressTRI[a][0])
        Elements.write(b,3,StressTRI[a][0])
        Elements.write(b,4,StressTRI[a][0])
        Elements.write(b,3,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,NodesTRI[a][1])
        Elements.write(b+1,2,StressTRI[a][0])
        Elements.write(b+1,3,StressTRI[a][0])
```

```
        Elements.write(b+1,4,StressTRI[a][0])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
    a=a+1
Elements.write(0,6,"Numero_linhas")
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
    NDSP[i]=nodeDisp(i)
    Nodes_l.write(i+1,0,i)
    Nodes_l.write(i+1,1,NDSP[i][0])
    Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
wb_elem.save('Resultados.xls')
print("Análise_finalizada")
```

Quadro D 8 - Bloco de código 8 (malha quadrada).

```
import xlrd
import openpyxl
from decimal import Decimal
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t10/Malha_t10.xls")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
wb_stress=Workbook()
no_atress=wb_stress.add_sheet('No_stress')
```

```
loc1 = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t10/Resultados.xls")
wb_r=xlrd.open_workbook(loc1)
elem_result=wb_r.sheet_by_index(0)
node_result=wb_r.sheet_by_index(1)
num_elem_dim=int(elem_result.cell_value(0,7))
num_node_dim=int(node_result.cell_value(0,4))
sx_node=np.zeros((num_node_dim,5),dtype=float)
num_mnode=np.zeros((num_node_dim,1),dtype=int)
counter_i=1
for i in range(0,num_node_dim):
   aux_x=0
   aux_y=0
   aux_xy=0
   counter=1
   num_node=int(node_result.cell_value(counter_i,0))
   num_mnode[i][0]=num_node
   for j in range(1,num_elem_dim):
      num_elem=int(elem_result.cell_value(j,1))
      if num_elem==num_node:
         sx_elem=float(elem_result.cell_value(j,2))
         aux_x=aux_x+sx_elem
         sy_elem=float(elem_result.cell_value(j,3))
         aux_y=aux_y+sy_elem
         sxy_elem=float(elem_result.cell_value(j,4))
         aux_xy=aux_xy+sxy_elem
         counter=counter+1
   num_mnode[i][0]=num_node
   sx_node[i][0]=aux_x/(counter-1)
   sx_node[i][1]=aux_y/(counter-1)
   sx_node[i][2]=aux_xy/(counter-1)
   sx_node[i][3]=float(node_result.cell_value(counter_i,1))
   sx_node[i][4]=float(node_result.cell_value(counter_i,2))
   counter_i=counter_i+1
f=open("Resultados_malha_t10.vtk",'w')
f.write("# vtk DataFile Version 2.0\n")
f.write("untitled, Created by Gmsh 4.11.0\n")
f.write("ASCII\n")
f.write("DATASET UNSTRUCTURED_GRID\n")
points=int(sheet1.cell_value(0,5))
size_vertice=2*points
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
size_elementos=int(sheet3.cell_value(0,0))+1
for i in range(1,num_elementos):
   size_elementos=size_elementos+int(sheet3.cell_value(i,0))+1
f.write("POINTS %d float\n"%points)
for i in range(0,points):
   x=float(sheet1.cell_value(i,1))
   y=float(sheet1.cell_value(i,2))
   if abs(x)<0.000001:
      x=0
   if abs(y)<0.000001:
      y=0
   f.write("{} {} 0\n".format(x,y))
f.write("\n".format(x,y))
f.write("CELLS {} {}\n".format(num_elementos,size_elementos))
```

```
for i in range(0,num_elementos):
   type_elem=sheet3.cell_value(i,14)
   if type_elem==5:
      x=int(sheet3.cell_value(i,15))
      y=int(sheet3.cell_value(i,1))
      z=int(sheet3.cell_value(i,2))
      t=int(sheet3.cell_value(i,3))
      f.write("{} {} {} {}\n".format(x,z,y,t))
   if type_elem==8:
      x=int(sheet3.cell_value(i,15))
      y=int(sheet3.cell_value(i,1))
      z=int(sheet3.cell_value(i,2))
      t=int(sheet3.cell_value(i,3))
      w=int(sheet3.cell_value(i,4))
      f.write("{} {} {} {} {}\n".format(x,z,y,t,w))
f.write("\n")
f.write("CELL_TYPES %d\n"%num_elementos)
for i in range(0,num_elementos):
   x=int(sheet3.cell_value(i,14))
   f.write("%d\n"%x)
f.write("\n")
f.write("POINT_DATA %d\n"%points)
f.write("FIELD FieldData 4\n")
f.write("Displacement 3 %d float\n"%points)
for i in range(0,points):
   x=float(sx_node[i][3])
   y=float(sx_node[i][4])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
ids="%20"
pontos=int(points)
f.write("Displacement{}Magnitude 1 {} float\n".format(ids,pontos))
for i in range(0,points):
   x=float(sx_node[i][3])
   y=float(sx_node[i][4])
   magnitude=np.sqrt(x*x+y*y)
   f.write("%6.4f\n"%magnitude)
f.write("Stress_H 3 %d float\n"%points)
for i in range(0,points):
   x=float(sx_node[i][0])
   y=float(sx_node[i][2])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
f.write("Stress_V 3 %d float\n"%points)
for i in range(0,points):
   x=float(sx_node[i][2])
   y=float(sx_node[i][1])
   z=0
   f.write("{} {} {}\n".format(x,y,z))
type_data='SX'#str(sheet1.cell_value(0,6))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][0]#float(sheet1.cell_value(i,7))
   f.write("%6.4f\n"%x)
type_data='SY'#str(sheet1.cell_value(0,8))
f.write("SCALARS %s float 1\n"%type_data)
```

```
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][1]#float(sheet1.cell_value(i,9))
   f.write("%6.4f\n"%x)
type_data='SXY'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][2]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
type_data='ux'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][3]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
type_data='uy'#str(sheet1.cell_value(0,10))
f.write("SCALARS %s float 1\n"%type_data)
f.write("LOOKUP_TABLE default\n")
for i in range(0,points):
   x=sx_node[i][4]#float(sheet1.cell_value(i,11))
   f.write("%6.4f\n"%x)
f.write("VECTORS vector float\n")
for i in range(0,points):
   x=sx_node[i][3]
   y=sx_node[i][4]
   z=0
   f.write("{} {} {}\n".format(x,y,z))
f.write("\n")
f.close()
print("Análise_finalizada")
```

Quadro D 9 - Bloco de código 9 (malha quadrada).

```
from openseespy.opensees import *
import xlrd
import xlwt
from xlwt import Workbook
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import opsvis as opsv
import matplotlib.pyplot as plt
import os
wipe()
wb_elem=Workbook()
Elements=wb_elem.add_sheet('Elem',cell_overwrite_ok=True)
Nodes_l=wb_elem.add_sheet('Nodes_l',cell_overwrite_ok=True)
loc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a
dissertação/Malha_t10/Malha_t10.xls")
lovaloc = ("C:/Users/Utilizador/Desktop/OpenSees_Para a dissertação/Malha_t10")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
```

```
sheet1 = wb.sheet_by_index(1)
sheet2=wb.sheet_by_index(2)
sheet3=wb.sheet_by_index(3)
sheet4=wb.sheet_by_index(4)
sheet5=wb.sheet_by_index(5)
model('basic', '-ndm', 2, '-ndf', 2)
num_nos=int(sheet1.cell_value(0,5))
num_elementos=int(sheet3.cell_value(0,13))
num_nos_ap=int(sheet5.cell_value(0,8))
nDMaterial('ElasticIsotropic', 1, 30e6, 0)
mat_tag=1
type_=str(sheet3.cell_value(0,10))
for i in range(0,num_nos):

node(int(sheet1.cell_value(i,0)),float(sheet1.cell_value(i,1)),float(sheet1.cell_value(i,
2)))
for i in range(0,num_elementos):
   tipo_elemento=int(sheet3.cell_value(i,14))
   if tipo_elemento==8:
      element('quad',  i,  int(sheet3.cell_value(i,1)),  int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),                          int(sheet3.cell_value(i,4)),
float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)), mat_tag)
   if tipo_elemento==5:
      element('tri31',  i,  int(sheet3.cell_value(i,1)),  int(sheet3.cell_value(i,2)),
int(sheet3.cell_value(i,3)),float(sheet3.cell_value(i,9)),str(sheet3.cell_value(0,10)),
mat_tag)
for i in range(0,num_nos_ap):

fix(int(sheet5.cell_value(i,0)),int(sheet5.cell_value(i,1)),int(sheet5.cell_value(i,2)))
opsv.plot_model()
timeSeries('Linear', 1)
pattern('Plain', 1, 1)
load(54, 0., -100.)
load(56, 0., -100.)
system('BandGeneral')
constraints('Transformation')
numberer('RCM')
test('NormDispIncr', 1.0e-12, 100, 1)
algorithm('Linear')
integrator('LoadControl', 1)
analysis('Static')
analyze(1)
opsv.plot_model()
plt.axis('equal')
opsv.plot_loads_2d()
opsv.plot_defo(unDefoFlag=1)
plt.axis('equal')
sig_out = opsv.sig_out_per_node()
j, jstr = 1, 'syy'
nds_val = sig_out[:, j]
StressQUAD=np.zeros((num_elementos,12),dtype=float)
StressTRI=np.zeros((num_elementos,3),dtype=float)
Nodes=np.zeros((num_elementos,4),dtype=float)
NodesTRI=np.zeros((num_elementos,3),dtype=float)
a=0
b=1
for i in range(0,num_elementos):
```

```
    tipo_elemento=int(sheet3.cell_value(i,14))
    if tipo_elemento==8:
        StressQUAD[i]=eleResponse(i,'stress')
        Nodes[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=Nodes[a][0]
        Elements.write(b,1,Nodes[a][0])
        Elements.write(b,2,StressQUAD[a][0])
        Elements.write(b,3,StressQUAD[a][1])
        Elements.write(b,4,StressQUAD[a][2])
        Elements.write(b,5,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,Nodes[a][1])
        Elements.write(b+1,2,StressQUAD[a][3])
        Elements.write(b+1,3,StressQUAD[a][4])
        Elements.write(b+1,4,StressQUAD[a][5])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,Nodes[a][2])
        Elements.write(b+2,2,StressQUAD[a][6])
        Elements.write(b+2,3,StressQUAD[a][7])
        Elements.write(b+2,4,StressQUAD[a][9])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        Elements.write(b+3,1,Nodes[a][3])
        Elements.write(b+3,0,a)
        Elements.write(b+3,2,StressQUAD[a][9])
        Elements.write(b+3,3,StressQUAD[a][10])
        Elements.write(b+3,4,StressQUAD[a][11])
        Elements.write(b+3,5,tipo_elemento)
        b=b+4
    if tipo_elemento==5:
        StressTRI[i]=eleResponse(i,'stress')
        NodesTRI[i]=eleNodes(i)
        Elements.write(b,0,a)
        aux=NodesTRI[a][0]
        Elements.write(b,1,NodesTRI[a][0])
        Elements.write(b,2,StressTRI[a][0])
        Elements.write(b,3,StressTRI[a][0])
        Elements.write(b,4,StressTRI[a][0])
        Elements.write(b,3,tipo_elemento)
        Elements.write(b,0,a)
        Elements.write(b+1,1,NodesTRI[a][1])
        Elements.write(b+1,2,StressTRI[a][0])
        Elements.write(b+1,3,StressTRI[a][0])
        Elements.write(b+1,4,StressTRI[a][0])
        Elements.write(b+1,5,tipo_elemento)
        Elements.write(b+1,0,a)
        Elements.write(b+2,1,NodesTRI[a][2])
        Elements.write(b+2,2,StressTRI[a][0])
        Elements.write(b+2,3,StressTRI[a][0])
        Elements.write(b+2,4,StressTRI[a][0])
        Elements.write(b+2,5,tipo_elemento)
        Elements.write(b+2,0,a)
        b=b+3
    a=a+1
Elements.write(0,6,"Numero_linhas")
```

```
Elements.write(0,7,b-1)
NDSP=np.zeros((num_nos,2),dtype=float)
for i in range(0,num_nos):
   NDSP[i]=nodeDisp(i)
   Nodes_l.write(i+1,0,i)
   Nodes_l.write(i+1,1,NDSP[i][0])
   Nodes_l.write(i+1,2,NDSP[i][1])
Nodes_l.write(0,3,"Numero_nos")
Nodes_l.write(0,4,num_nos)
wb_elem.save('Resultados.xls')
print("Análise_finalizada")
```