
OPENCADD
MODEL - BASED DESIGN DRIVEN COMPANY
Modeling for life!

Ferramenta Gráfica para Modelagem e Análise em Engenharia Estrutural

Rafael Rangel (rafaelrangel@tecgraf.puc-rio.br)
Pedro Cortez (cortezpedro@tecgraf.puc-rio.br)

OPENCADD
MODEL - BASED DESIGN DRIVEN COMPANY
Modeling for life!

OPENCADD
MODEL - BASED DESIGN DRIVEN COMPANY
Modeling for life!

Engenharia estrutural
Desenvolvimento de um programa de análise
Arquitetura do código para processamento dos dados

Experiência na PUC-Rio no Desenvolvimento de Software Educacional em MATLAB

O Instituto Tecgraf/PUC-Rio, seguindo tradição e experiência no desenvolvimento de software educacional para engenharia, está ativo no uso do MATLAB para aumentar sua contribuição na área.

Exemplo: Software para análise estrutural
LESM – Linear Elements Structure Model

Outros departamentos e laboratórios também já iniciaram desenvolvimentos de programas educacionais.

O Centro Técnico Científico (CTC) da PUC-Rio está atento para essa nova realidade e realiza cursos para desenvolvimento de aplicativos gráficos no ambiente MATLAB.

Pontifícia Universidade Católica do Rio de Janeiro

30 ANOS TECGRAF PUC-RIO

Engenharia Estrutural

Ramo da engenharia civil dedicado ao projeto de estruturas para transporte, moradia, trabalho e lazer.

Objetivo do projeto estrutural

Documentar os detalhes para a execução de uma estrutura que atenda todas as necessidades para as quais ela foi planejada, visando a segurança e economia.

Principais etapas do projeto estrutural

- Concepção do modelo estrutural (modelagem)
- Cálculo dos esforços e deformações (análise)
- Dimensionamento e detalhamento das peças

Engenharia Estrutural

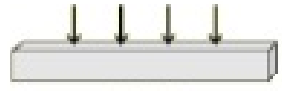
Modelo Estrutural

Idealização da estrutura real assumindo hipóteses simplificadoras sobre o seu comportamento que permitem descreve-la através de formulações matemáticas provenientes das teorias de mecânica dos sólidos.

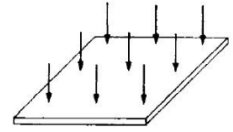
Hipóteses simplificadoras:

- Geometria do modelo
- Conexões internas e externas
- Propriedades físicas do material
- Carregamento aplicado

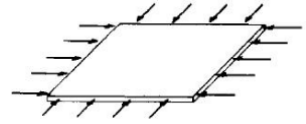
Exemplos de elementos estruturais:



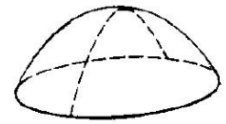
Viga



Placa



Chapa



Casca

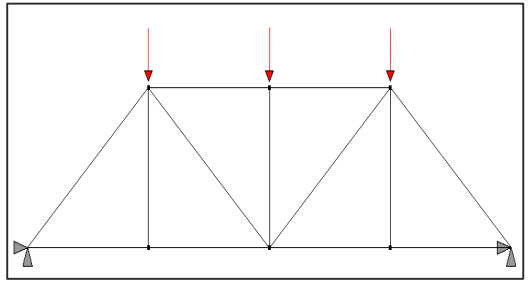
Engenharia Estrutural

Modelos Reticulados

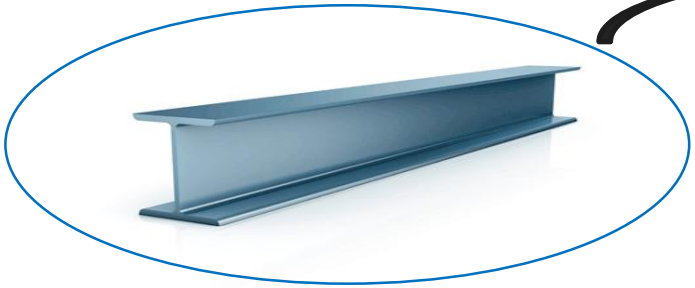
Tipo mais simples de modelo estrutural, formado por elementos estruturais que possuem um eixo claramente definido (barras, vigas, etc.).

Estrutura real

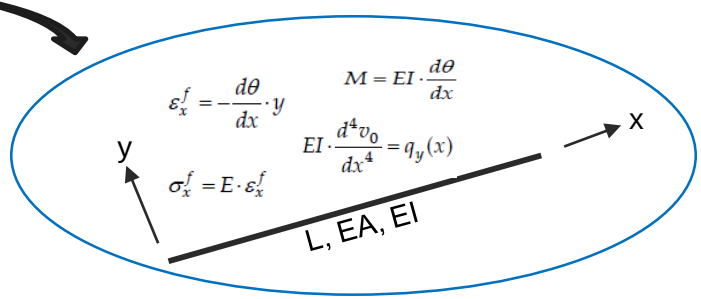
Hipóteses Simplificadoras



Modelo estrutural



Perfil de aço



Idealização matemática

Engenharia Estrutural

Análise Estrutural

Estudo dos efeitos das cargas no modelo estrutural para a determinação dos esforços e tensões e dos deslocamentos e deformações da estrutura que está sendo projetada.

Os resultados da análise são utilizados pelos projetistas para dimensionar as peças.

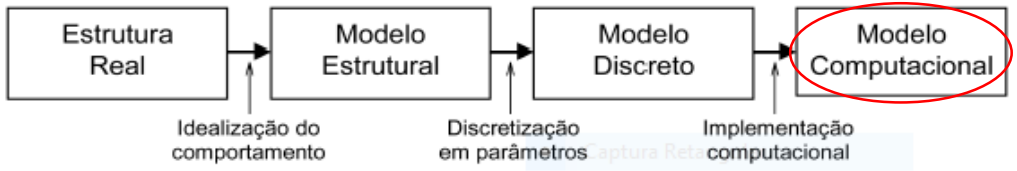
Modelo Discreto

As equações que descrevem o comportamento do modelo estrutural muitas vezes não possuem soluções analíticas ou são complexas de se obter.

Os métodos de análise consistem na discretização do modelo estrutural em um conjunto de parâmetros cujos valores são calculados para representar as soluções analíticas contínuas.

Ex.: Método das Forças, Método da Rigidez Direta, Método dos Elementos Finitos, etc.

Mecânica Computacional



Modelo Computacional

Implementação de um algoritmo para calcular os valores dos parâmetros discretos.

Busca-se técnicas de programação que visam a maior eficiência computacional para a implementação do método de análise e obtenção dos resultados.

Mecânica Computacional é a disciplina que utiliza métodos computacionais para estudar fenômenos governados pelos princípios da mecânica.

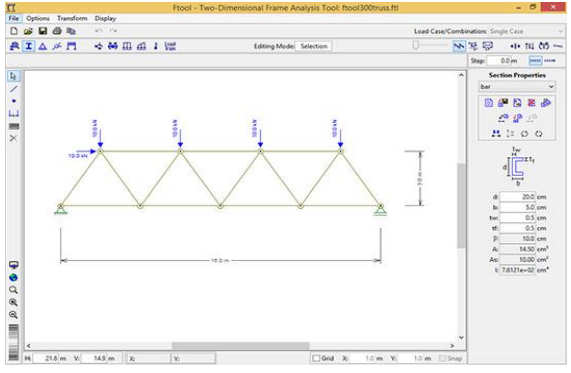
Princípios Mecânicos + Ciência da Computação

Mecânica Computacional

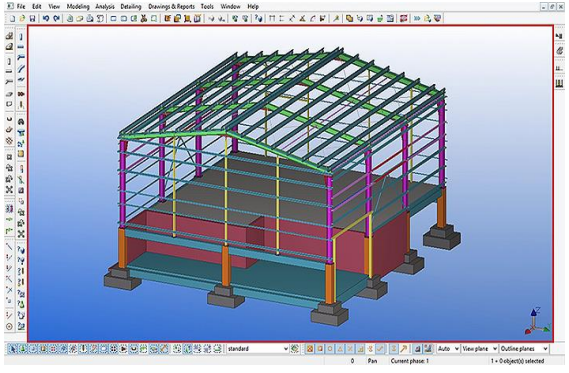
A importância da utilização de um software de análise estrutural:

- Estrutura de dados robusta
- Eficiência e rapidez nos cálculos
- Exibição gráfica do modelo e resultados

Com a criação de programas gráficos interativos, a análise estrutural passou a ser feita com uso de computador em praticamente todos os escritórios de cálculo estrutural.



SAP2000 ANSYS TQS SIMULIA ABAQUS

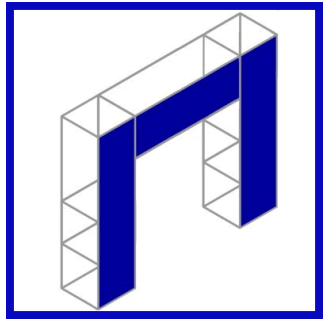


LESM – Linear Elements Structure Model

Motivação

Nascido através de projetos de monografia visando os seguintes objetivos:

- Código aberto e bem documentado para o ensino de análise matricial de estruturas
- Priorização da clareza e didática do código em relação à eficiência
- Desenvolvimento de uma interface gráfica para modelagem visualização dos resultados



https://web.tecgraf.puc-rio.br/lesm/

A primeira versão do programa acompanha o livro *Análise Matricial de Estruturas com Orientação a Objetos* do autor Luiz Fernando Martha, a ser lançado em 2018.

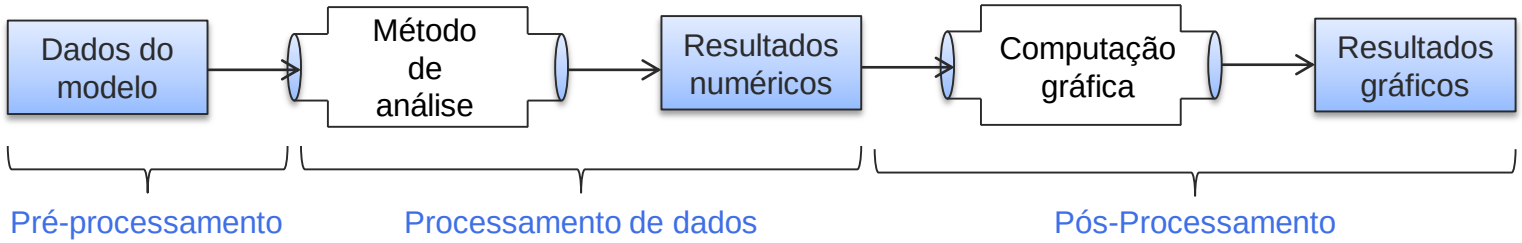
Características

- Modelos estruturais reticulados 2D e 3D (treliças, pórticos e grelhas)
- Análise estática linear-elástica
- Método da Rigidez Direta

Desenvolvimento de um Software de Análise Estrutural

Estágios de Processamento

- Pré-processamento: Fornecimento, captura e preparação dos dados do modelo à ser analisado
- Processamento: Cálculo dos resultados baseado nos dados fornecidos
- Pós-processamento: Tratamento e disponibilização dos resultados através de saídas gráficas



Diversos aspectos fundamentais no contexto de modelagem geométrica e computação gráfica estão envolvidos no desenvolvimento de um programa de computador para executar uma análise estrutural:

- Estruturas de dados e procedimentos para a criação do modelo geométrico
- Geração do modelo discreto
- Aplicação de atributos de análise (propriedades de materiais, carregamentos, condições de suporte etc.)
- Visualização dos resultados

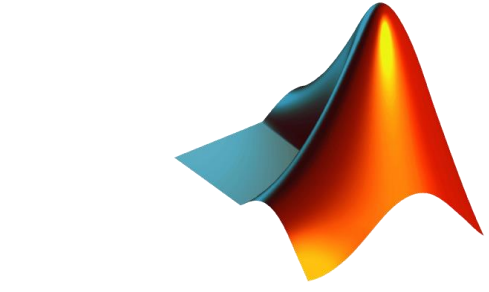
Uso do MATLAB para o Desenvolvimento de Aplicações

Ambiente de desenvolvimento integrado

O MATLAB é um ambiente de computação científica que engloba diversas funcionalidades na mesma plataforma.

No desenvolvimento do programa LESM, destacam-se:

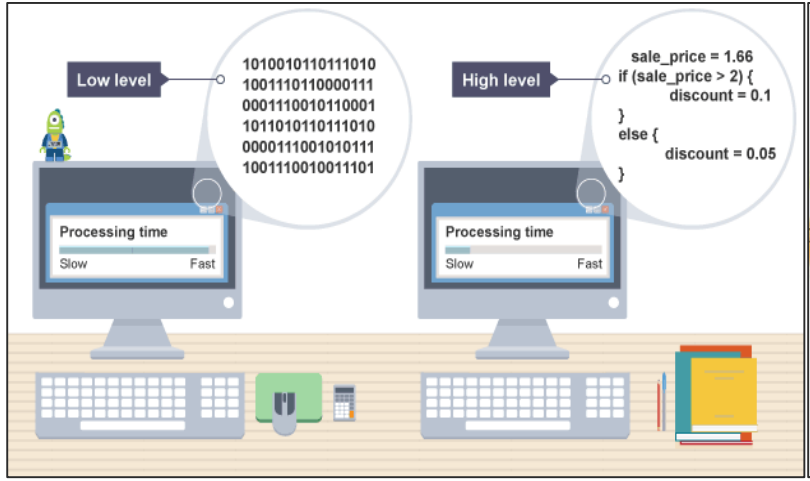
- Linguagem de programação de alto nível
- Programação Orientada a Objetos (POO)
- Álgebra simbólica
- Documentação
- Sistema gráfico
- Desenvolvimento de interface gráfica

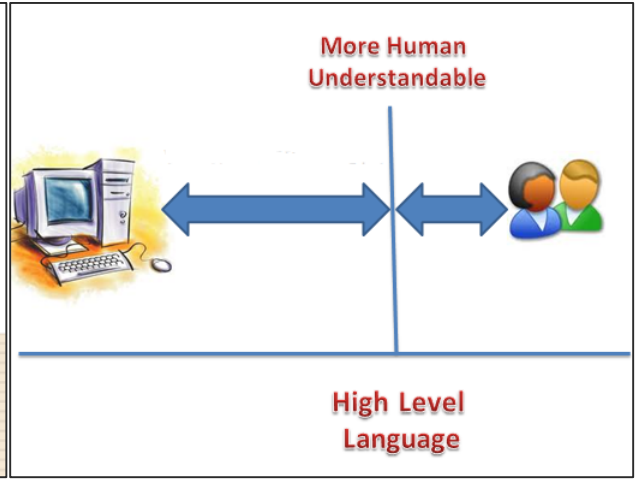


Linguagem de Programação de Alto Nível

- A linguagem de programação MATLAB é interpretada e considerada de 4ª geração
- Nível de abstração elevado, longe do código de máquina e próximo à linguagem humana
- Ampla biblioteca de funções pré definidas
- Editor de texto próprio com opções execução do código e *debug*
- Variáveis matriciais

Desvantagem: Maior tempo de processamento.



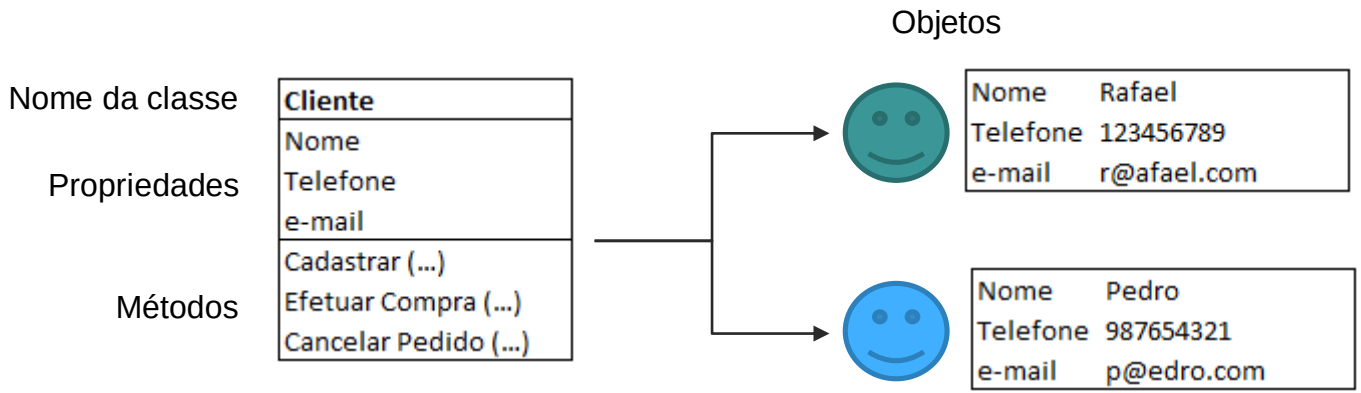


Programação Orientada a Objetos (POO)

Baseia-se na composição e interação de unidades de software chamadas objetos.

Definições importantes:

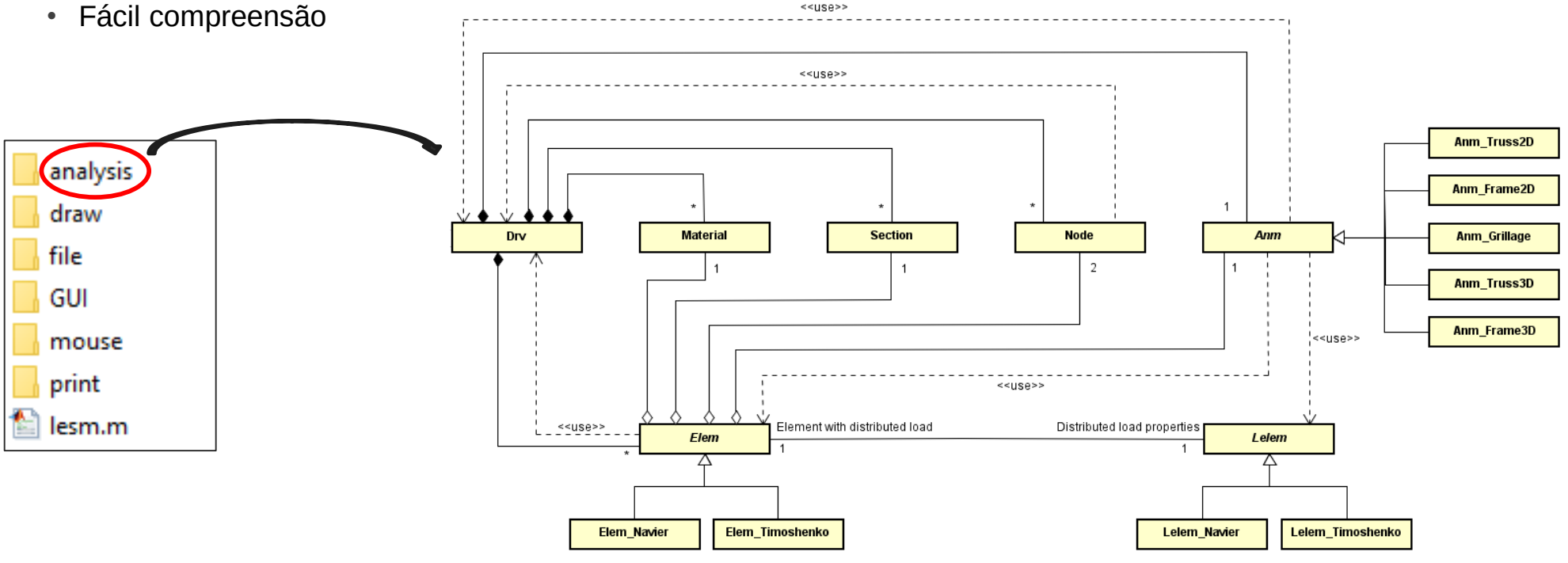
- **Classe**: Conjunto de variáveis e funções que agrupam e dão origem a objetos
- **Objeto**: Instâncias de uma classe
- **Propriedades**: Variáveis que representam as características dos objetos
- **Métodos**: Funções que definem as habilidades de um objeto
- **Herança**: Permite reaproveitar o código de uma classe base para implementações específicas em subclasses



Programação Orientada a Objetos (POO)

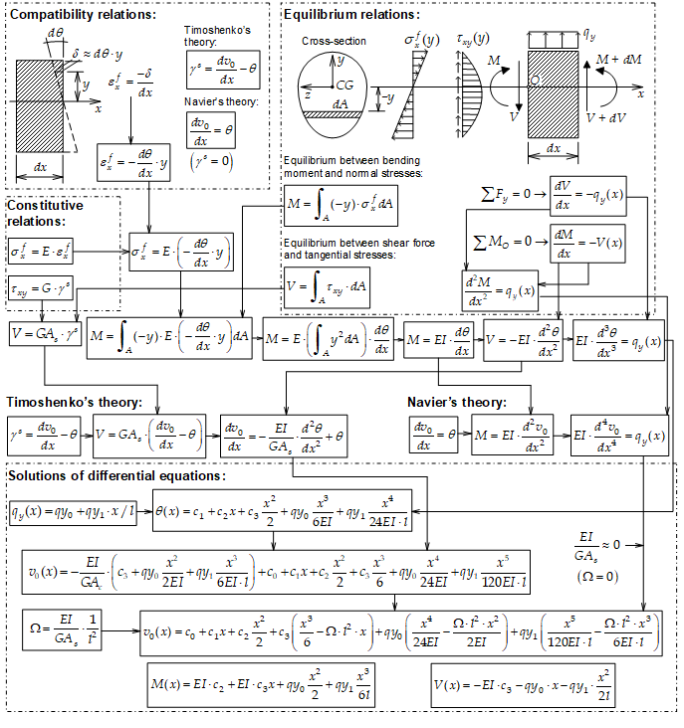
Vantagens do uso da POO

- Modularização do código
- Facilidade para manutenção e expansão do programa
- Fácil compreensão



Ambiente de Álgebra Simbólica

- Manipulação de equações matemáticas e expressões em forma simbólica
- Utilizado para gerar as expressões dos modelos matemáticos que representam o comportamento das barras
- Os resultados podem ser exportados para o formato de código MATLAB, LATEX, etc.



$$N2v(x) = \frac{2x^3}{L^3} - \frac{3x^2}{L^2} + 1$$

$$N3v(x) = x - \frac{2x^2}{L} + \frac{x^3}{L^2}$$

$$N5v(x) = \frac{3x^2}{L^2} - \frac{2x^3}{L^3}$$

$$N6v(x) = \frac{x^3}{L^2} - \frac{x^2}{L}$$

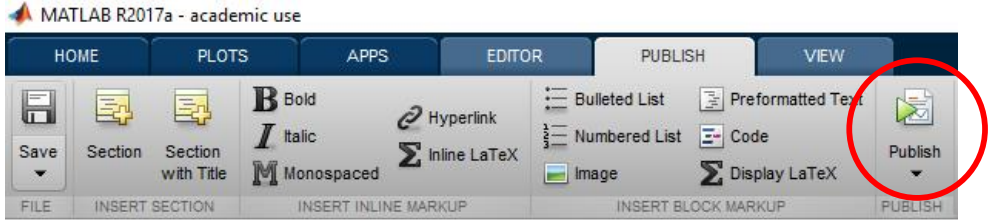
```
N2v(x) =

(2*x^3)/L^3 - (3*x^2)/L^2 + 1


N3v(x) =

x - (2*x^2)/L + x^3/L^2


N5v(x) =

(3*x^2)/L^2 - (2*x^3)/L^3


N6v(x) =

x^3/L^2 - x^2/L
```

Documentação

Um código bem documentado é a chave para o seu entendimento, manutenção, modificação e expansão.

Comando *Publish*

- A documentação de um código é geralmente feita através de comentários, necessários para esclarecer os procedimentos realizados
- O comando *Publish* do MATLAB permite criar arquivos formatados para serem publicados e compartilhados a partir dos comentários do código fonte.



Documentação

Arquivo *.m*

lesm.m

```
%% LESM - Linear Elements Structure Model
%
% This is the main driver file of LESM. This is a MATLAB program for
% linear-elastic, displacement-based, static analysis of bi-dimensional
% and tri-dimensional linear elements structure models, using the direct
% stiffness method. For each structural analysis, the program assembles a
% system of equilibrium equations, solves the system and displays the
% analysis results.
%
% The program handles some reticulated structure analysis models:
% 2D and 3D trusses, 2D and 3D frames, and grillages. In addition, two
% types of flexural behavior for linear elements are considered:
% Euler-Bernoulli (Navier theory) elements and Timoshenko elements.
%
% The program may be used in a non-graphical version or in a GUI
% (Graphical User Interface) version.
% The non-graphical version reads a structural model from a neutral
% format file and prints model information and analysis results in the
% the default output (MATLAB command window).
% In the GUI version, a user may create a structural model with
% attributes through the program graphical interface. The program can
% save and read a structural model data stored in a neutral format file.
% The standalone executable of the GUI version can be downloaded from the
% LESM website: <https://web.tecgraf.puc-rio.br/lesm/>
%
%% Authors
%%%
% * Luiz Fernando Martha (lfm@tecgraf.puc-rio.br)
%%%
% * Rafael Lopez Rangel (rafaelrangel@tecgraf.puc-rio.br)
%%%
% * Pedro Cortez Lopes (cortezpedro@tecgraf.puc-rio.br)
%%%
% Pontifical Catholic University of Rio de Janeiro - PUC-Rio
%
% Department of Civil and Environmental Engineering and Tecgraf Institute
% of Technical-Scientific Software Development of PUC-Rio (Tecgraf/PUC-Rio)
```

Arquivo *.html*

Web Browser - LESM - Linear Elements Structure Model

LESM - Linear Elements Structure Model

Location: file:///C:/Users/rafaelrangel/Desktop/html/lesm.html

LESM - Linear Elements Structure Model

This is the main driver file of LESM. This is a MATLAB program for linear-elastic, displacement-based, static analysis of bi-dimensional and tri-dimensional linear elements structure models, using the direct stiffness method. For each structural analysis, the program assembles a system of equilibrium equations, solves the system and displays the analysis results.

The program handles some reticulated structure analysis models: 2D and 3D trusses, 2D and 3D frames, and grillages. In addition, two types of flexural behavior for linear elements are considered: Euler-Bernoulli (Navier theory) elements and Timoshenko elements.

The program may be used in a non-graphical version or in a GUI (Graphical User Interface) version. The non-graphical version reads a structural model from a neutral format file and prints model information and analysis results in the the default output (MATLAB command window). In the GUI version, a user may create a structural model with attributes through the program graphical interface. The program can save and read a structural model data stored in a neutral format file. The standalone executable of the GUI version can be downloaded from the LESM website: https://web.tecgraf.puc-rio.br/lesm/

Contents

Authors

- Luiz Fernando Martha (lfm@tecgraf.puc-rio.br)
- Rafael Lopez Rangel (rafaelrangel@tecgraf.puc-rio.br)
- Pedro Cortez Lopes (cortezpedro@tecgraf.puc-rio.br)

Pontifical Catholic University of Rio de Janeiro - PUC-Rio

Documentação

Diagramas UML (Unified Modeling Language)

O padrão UML de modelagem de software apresenta uma documentação formal para aplicações através de diagramas que mostram graficamente diferentes aspectos da arquitetura do programa.

Diagramas implementados no programa:

- Diagrama de Classe: Expõe a organização das classes e a forma com a qual elas se relacionam.
- Diagrama de Sequencia: Exibe a chamada dos métodos e interações entre objetos na ordem de execução
- Diagrama de Atividade: Fluxograma descritivo das etapas do programa

Documentação

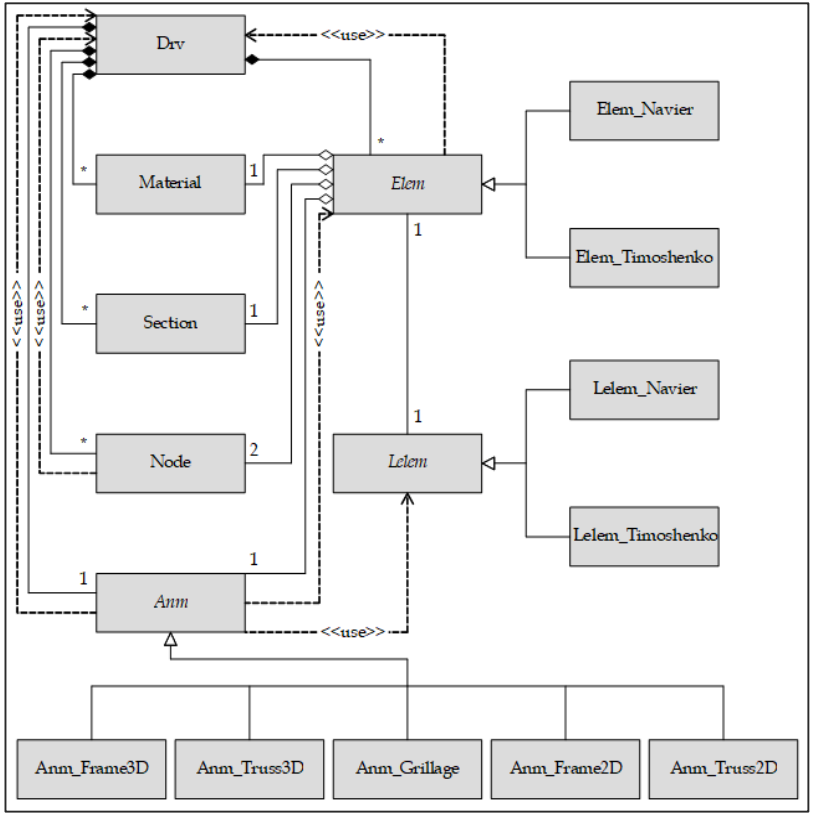


Diagrama de Classe

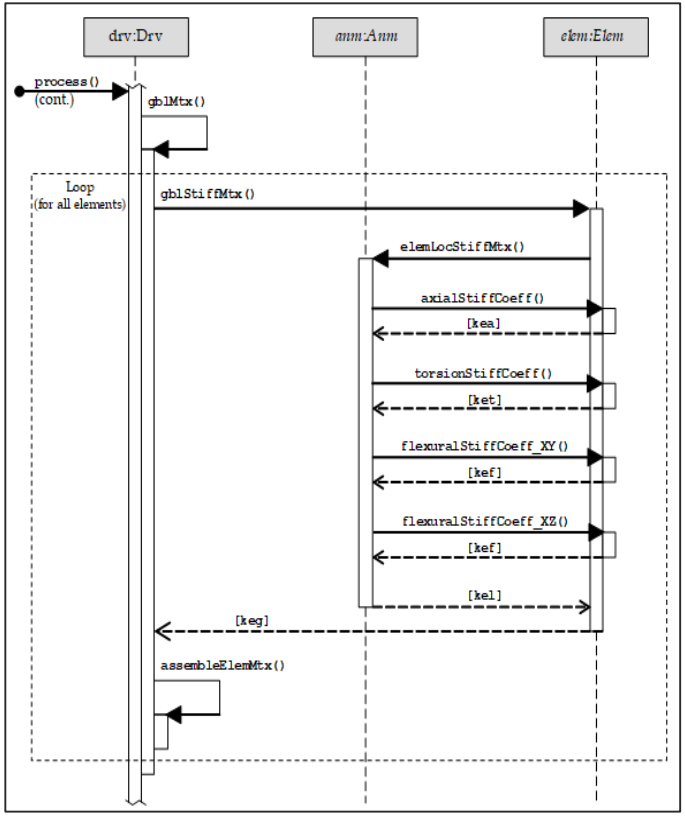


Diagrama de Sequencia

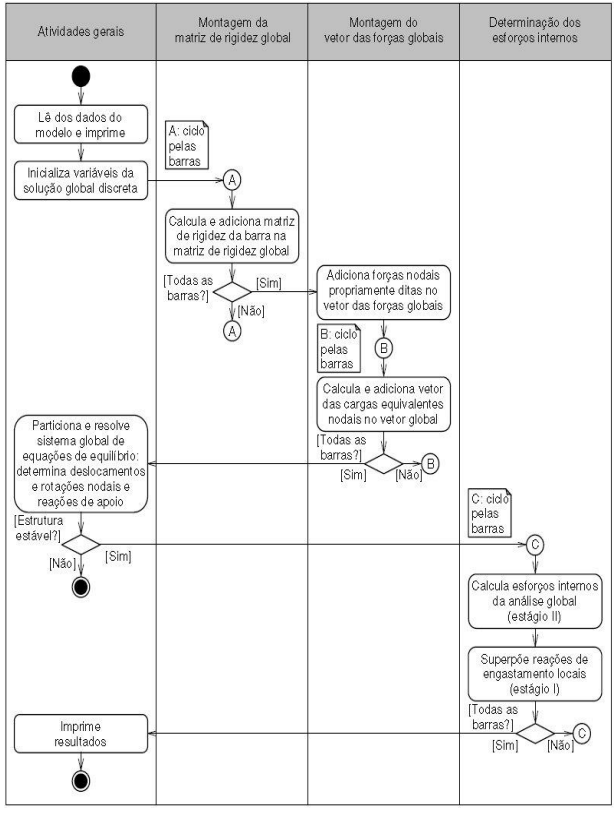


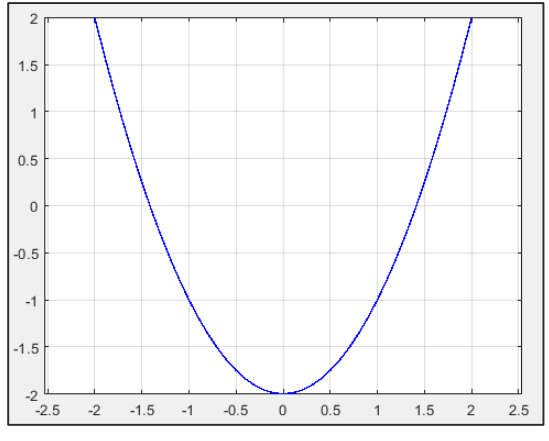
Diagrama de Atividade

Funcionalidades gráficas
Construção e gerenciamento de interfaces
Interatividade via mouse

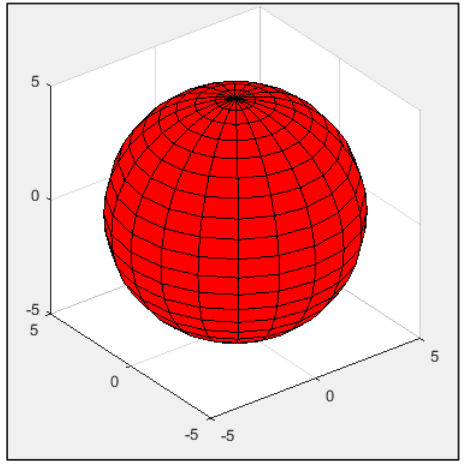
Funções Gráficas do MATLAB

- Ampla biblioteca para plotagem de primitivas gráficas (pontos, segmentos, polilinhas, etc.).
- A plotagem é feita em um objeto próprio de eixos 2D ou 3D, geralmente utilizado para a exibição de gráficos e curvas. No caso de aplicações gráficas, é chamado de *canvas*, onde trabalha-se para representar visualmente e interagir com os modelos.

```
>> % y(x) = x² - 2
>> x = -2:0.1:2;
>> y = polyval([1, 0, -2],x);
>> plot(x,y,'color','blue','linewidth',1.1)
>> axis equal
>> grid on
```

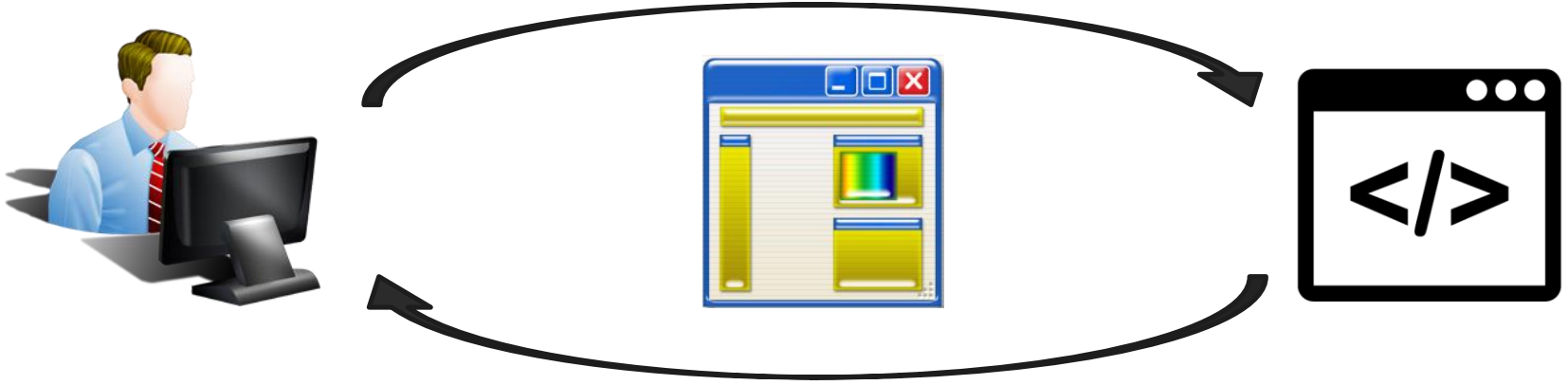



```
>> [x,y,z] = sphere;
>> s = surf(5*x,5*y,5*z);
>> set(s, 'Edgecolor', 'black','FaceColor', 'red');
>> axis equal
```



Construção de Diálogos

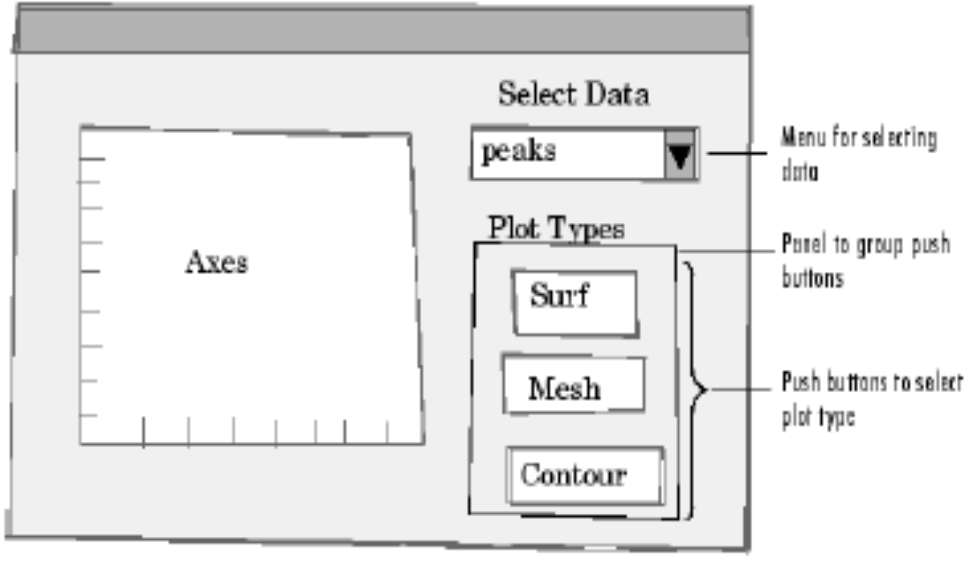
- Uma **interface gráfica** (*GUI* – Graphical User Interface) permite a interação do usuário do computador com um programa por meio de **componentes gráficos** (botões, menus, etc.).
- O MATLAB possui o **ambiente GUIDE** (*Graphical User Interface Development Environment*) para construção de diálogos de forma interativa.
- A comunicação entre as ações realizadas pelo usuário sobre os componentes gráficos e a resposta do programa se dá por meio do paradigma da **Programação Orientada a Eventos**.



Construção de Diálogos

Projeto da interface gráfica

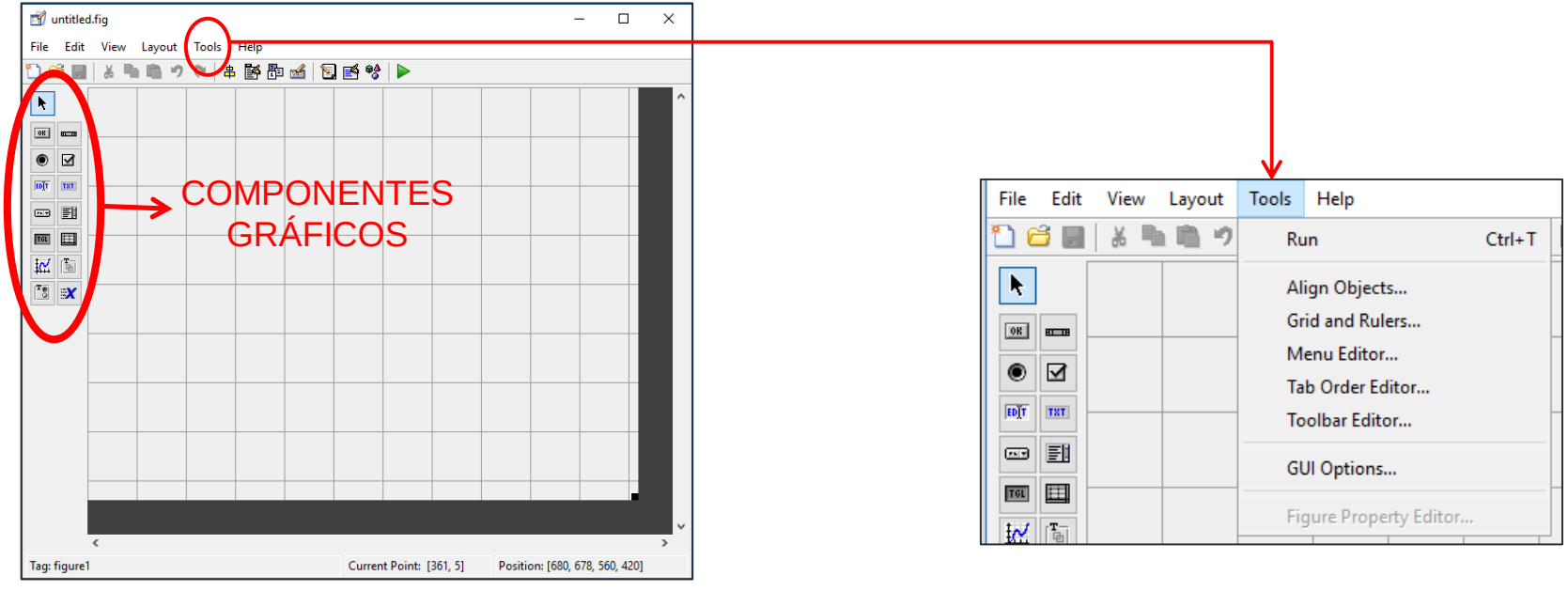
- Quais funcionalidades estarão disponíveis e como serão executadas
- Quais os componentes serão utilizados para permitir o usuário realizar essas tarefas
- Prever possíveis erros de uso do programa para cercar as ações do usuário
- Organização do layout



Construção de Diálogos

Ambiente GUIDE

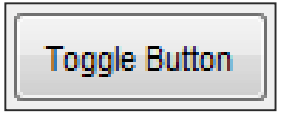
- Desenvolvimento de interfaces gráficas de forma interativa
- Os componentes são adicionados por meio de sistema de *drag-and-drop*
- Um arquivo *.m* baseado no paradigma de orientação a eventos, e um arquivo *.fig*, que descreve a composição do diálogo, são automaticamente gerados



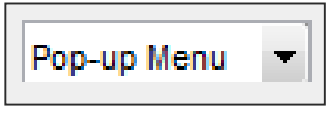
Construção de Diálogos

Componentes gráficos

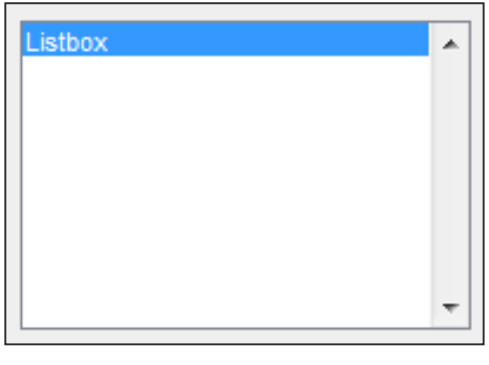
Botão de estado único.
Dispara uma ação através do *click*.



Botão com estado ativo e inativo.
Dispara uma ação através do *click*.

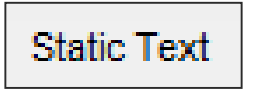


Menu *dropdown*. Dispara uma ação mediante seleção de uma opção.

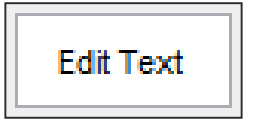


Lista interativa de dados.

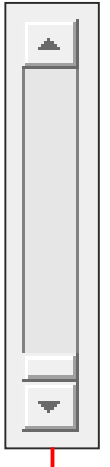
Seleção de estado (ativo ou inativo). Podem disparar ações mediante o *click.*



Texto sem interatividade.



Caixa de texto editável.



Slider para regulagem de valores. Dispara uma ação na mudança de posição.

Construção de Diálogos

Agrupa componentes variados

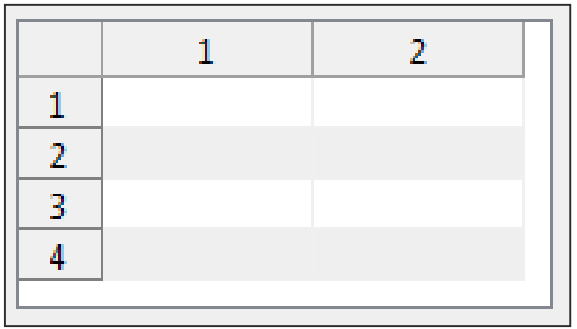
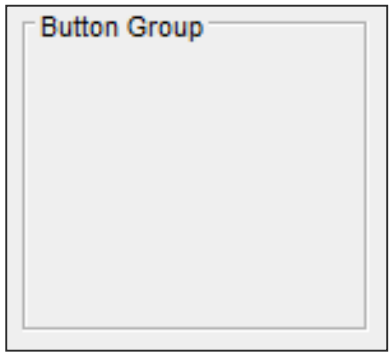
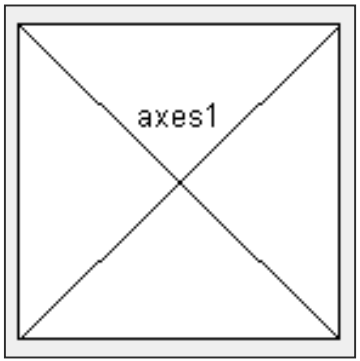


Tabela interativa para exposição, entrada e modificação de dados

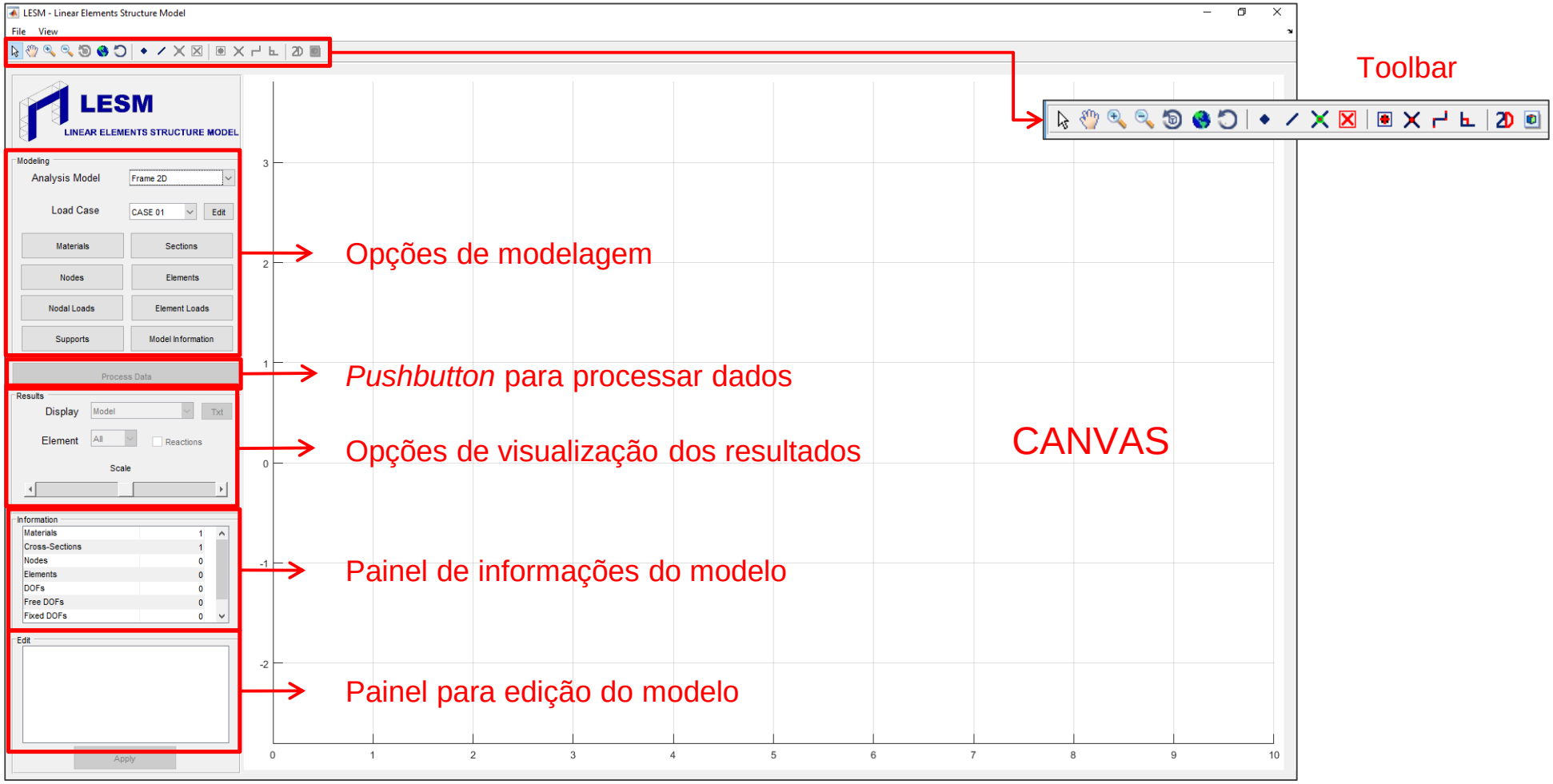


Agrupa *radio buttons* de forma que trabalhem com exclusividade mútua

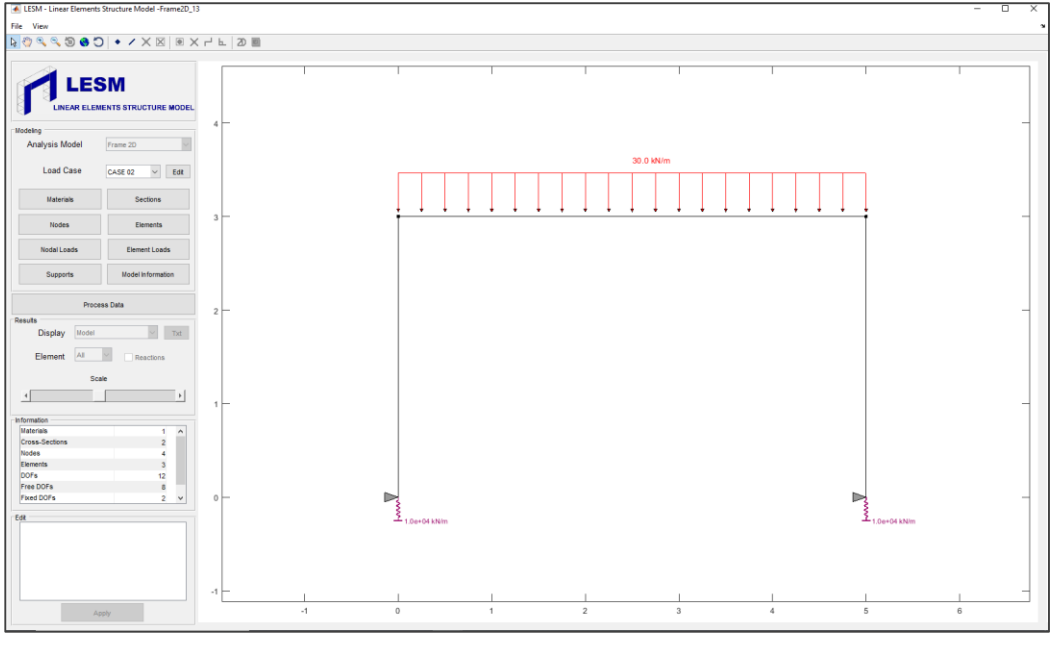


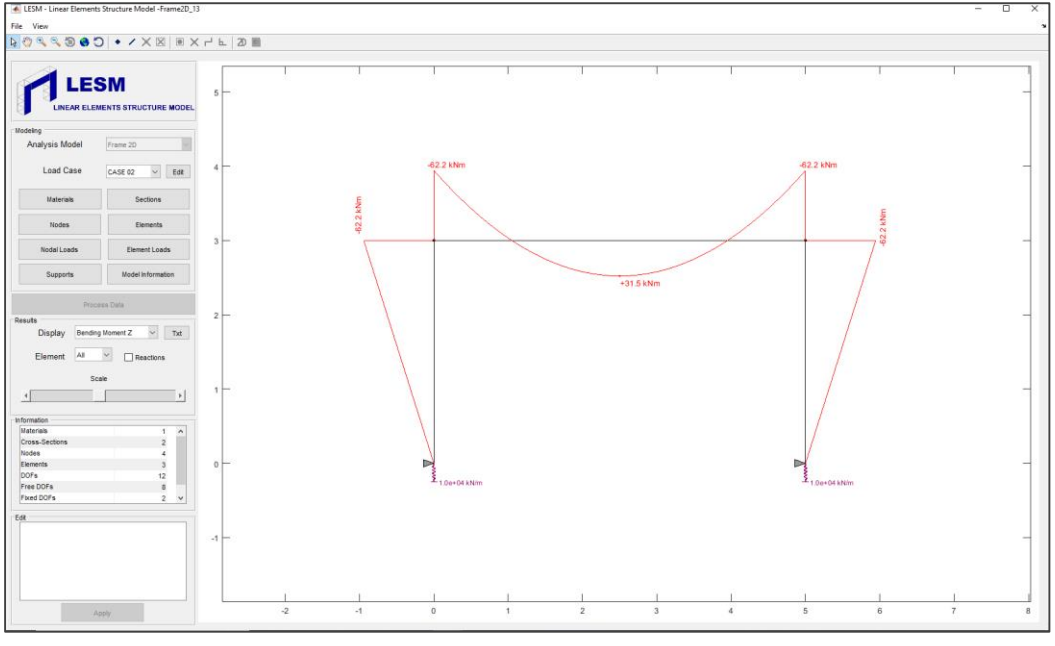
Eixos para plotagens e exibição de imagens

LESM – Linear Elements Structure Model



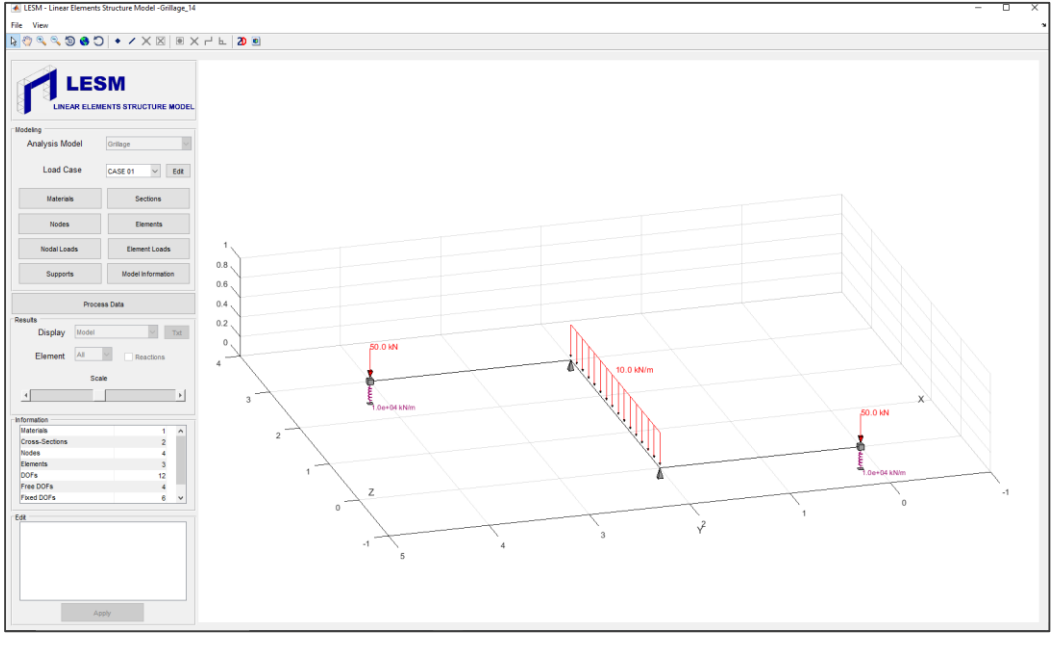
LESM – Linear Elements Structure Model

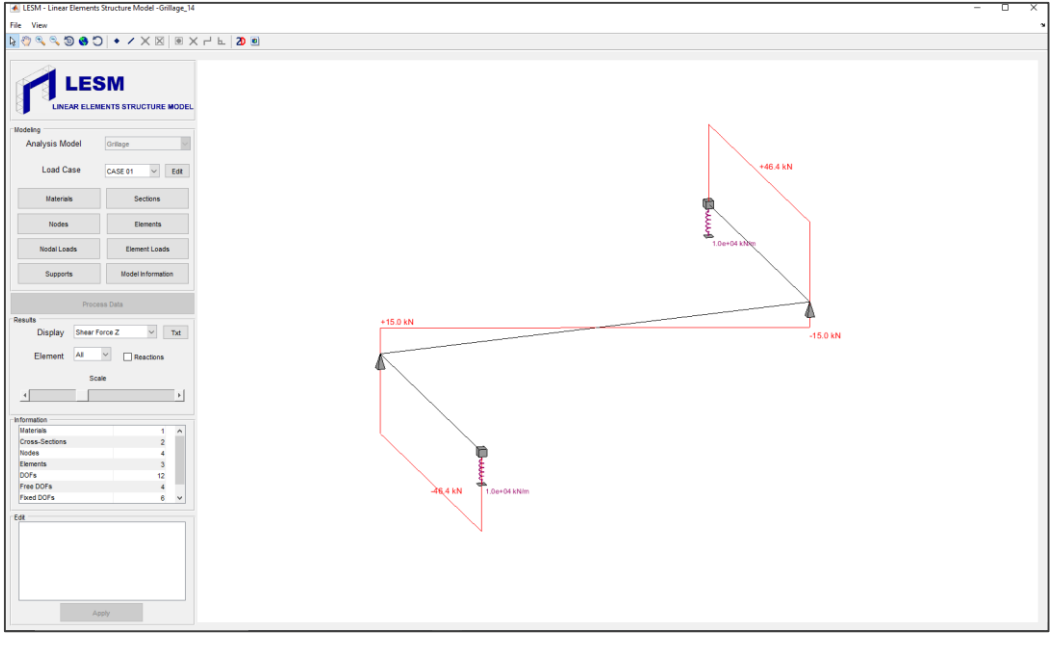




Modelo de pórtico plano e diagrama de momento fletor

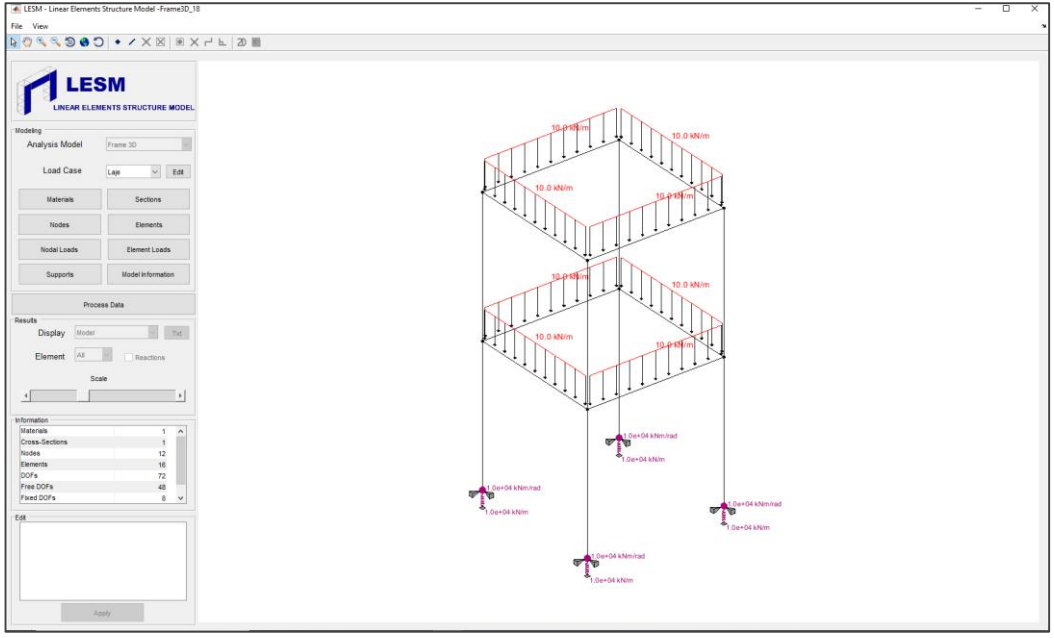
LESM – Linear Elements Structure Model

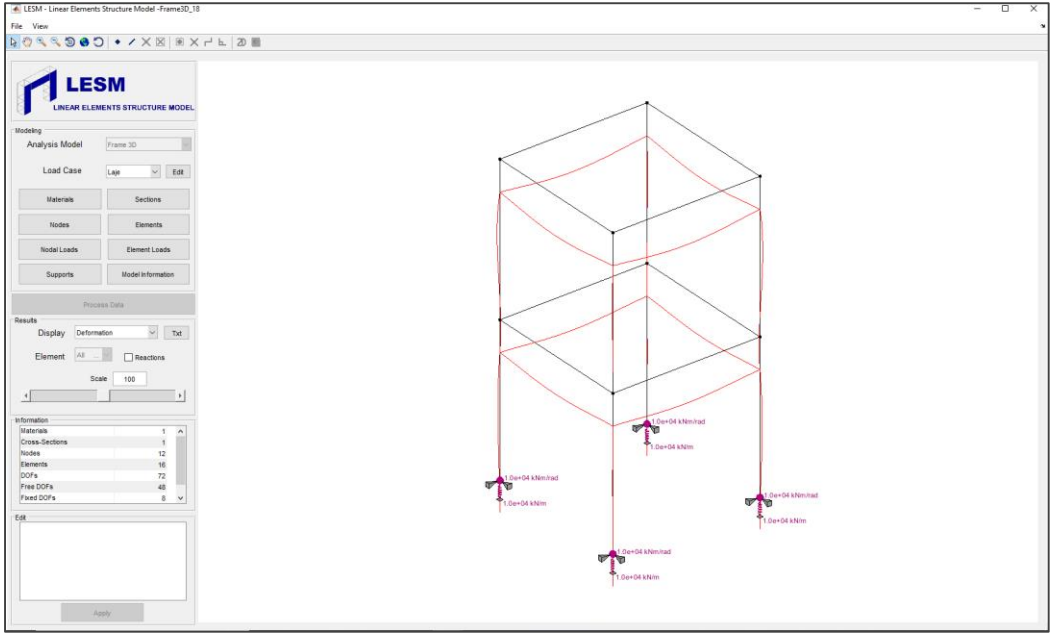




Modelo de grelha e diagrama de esforço cortante

LESM – Linear Elements Structure Model

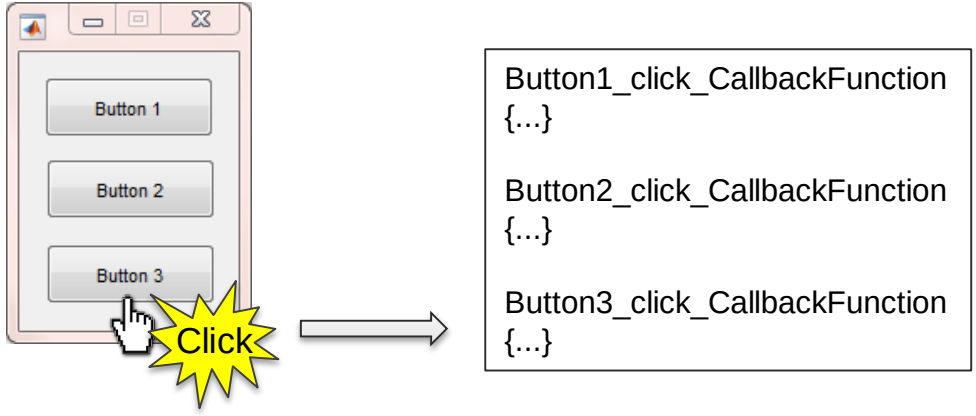




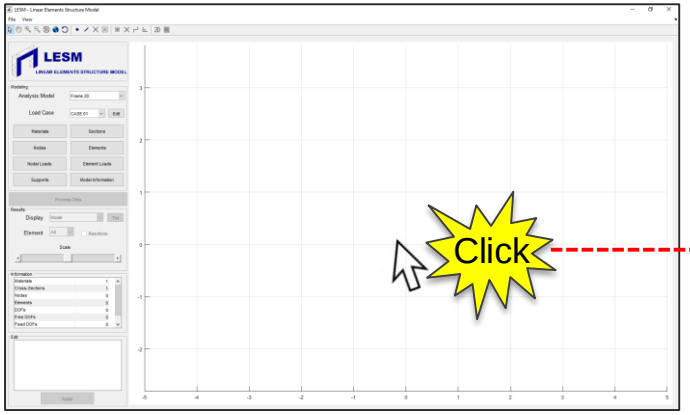
Modelo de pórtico espacial e configuração deformada

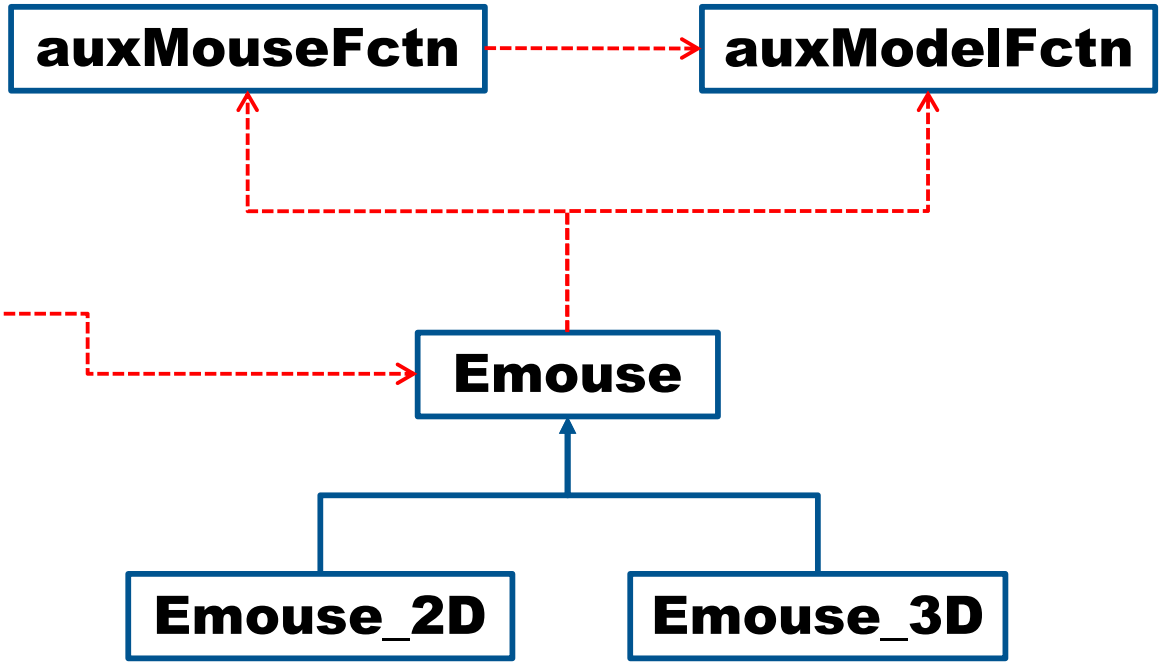
Programação Orientada a Eventos

- O fluxo do código é guiado por indicações externas chamadas eventos.
- Eventos são as diferentes ações que usuários podem realizar sobre os componentes adicionados à interface.
- Cada evento está associado à uma função chamada *callback*, disparada quando se verifica a ocorrência de tal evento, que define a reação do programa.



Gerenciamento de Eventos de Mouse no *Canvas*





Gerenciamento de Eventos de Mouse no *Canvas*

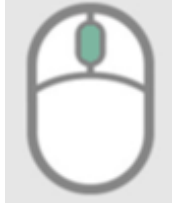
Autor:
Emersson Torres
(mestrado em Eng. Civil)

Classe Emouse

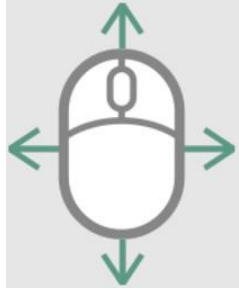
- Superclasse genérica que gerencia eventos de mouse

```
set( gcf, ‘Windowbuttonmotionfcn', @eMouseMove );
set( gcf, ‘Windowbuttondownfcn',   @ebuttonDown );
set( gcf, ‘Windowbuttonupfcn',       @ebuttonUp );
set( gcf, 'WindowScrollWheelFcn', @eUseScroll);
```

ebuttonUp

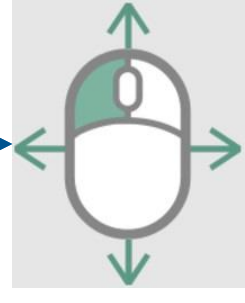


eUseScroll



eMouseMove

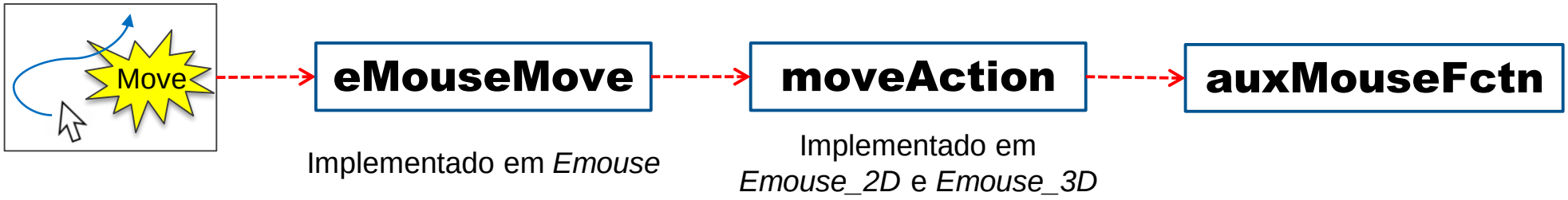
ebuttonDown



Gerenciamento de Eventos de Mouse no *Canvas*

Subclasses Emouse_2D e Emouse_3D

- Subclasses utilizadas para implementar métodos abstratos da classe base (Emouse) e armazenar propriedades específicas de modelos 2D e 3D no programa LESM.
- Seus métodos chamam funções auxiliares para a realização de tarefas como desenhar nós e elementos, sinalizar que o cursor está próximo de algum componente do modelo (*snap*), obter coordenadas em um *grid*, entre outras.



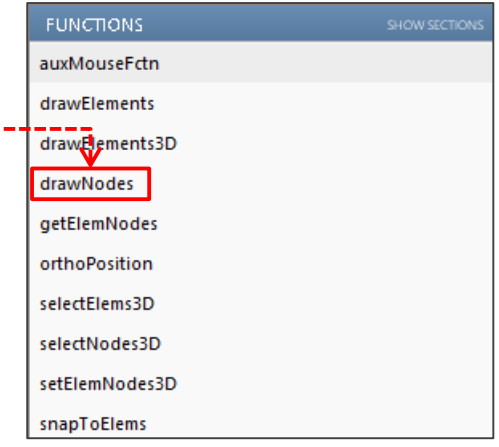
Gerenciamento de Eventos de Mouse no *Canvas*

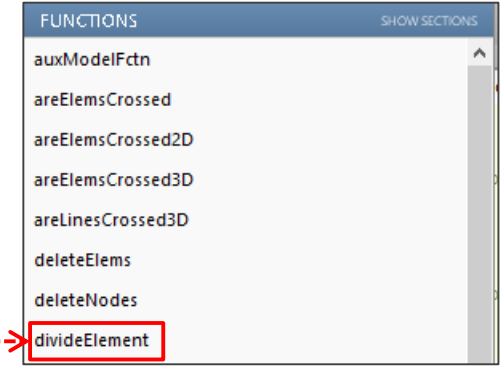
Funções Auxiliares

- Pela organização do código, foram criados dois arquivos de funções auxiliares (*auxMouseFctn* e *auxModelFctn*) onde foram implementadas rotinas sequenciais, que são chamadas diversas vezes por diferentes eventos de mouse, para realizar tarefas específicas, como desenhar nós e elementos.
- As funções auxiliares foram separadas em dois arquivos de forma que operações dependentes do objeto *Emouse* (*auxMouseFctn*) fiquem separadas das que não dependem do mesmo (*auxModelFctn*).
- Funcionam como interruptores, um de seus argumentos de *input* é uma *string* com o nome da função a ser chamada por uma expressão condicional *switch-case*.

```
[~] = auxMouseFctn('drawNodes',Emouse,{coords,inWhichElems});
```

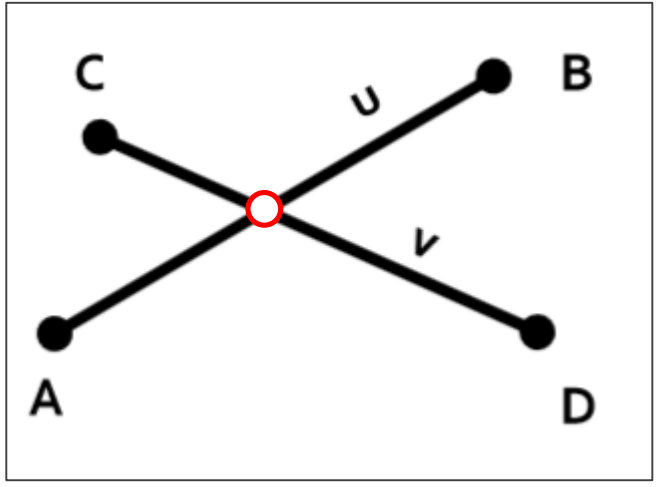
```
[~] = auxModelFctn('divideElement',{e,n});
```





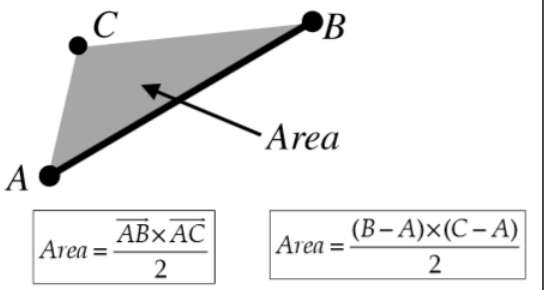
Computação Gráfica

- Uma vez que eventos de mouse passam a ser contemplados e utilizados pelo programa, surgem questões a serem resolvidas, principalmente no âmbito da modelagem interativa.
- Foram utilizados algoritmos de computação gráfica para tratar situações como o cruzamento e a colinearidade de elementos, a identificação de objetos próximos ao cursor e a obtenção de posição no espaço (3D) para a modelagem em perspectiva.



Cruzamento de Linhas no 2D – Área Orientada de Triângulos

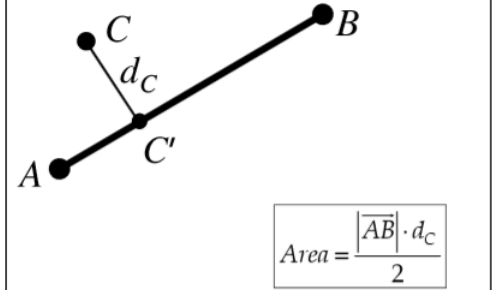
- O cruzamento de retas com pontos e/ou outras retas, em modelos 2D, foi tratado a partir do cálculo de áreas orientadas.



$$Area = \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2} \qquad Area = \frac{(B-A)\times(C-A)}{2}$$

$$Area = \frac{1}{2} \times \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix}$$

Área orientada de um triângulo, nota-se que pode ser negativa.

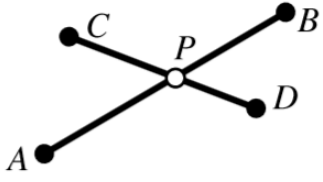


$$Area = \frac{|\overrightarrow{AB}| \cdot d_C}{2}$$

$$d_c = \frac{2 \times Area}{|\overrightarrow{AB}|}$$

Se a distância d_c for pequena (dentro de uma tolerância numérica), o ponto *C* está contido na linha *AB*.

Valor paramétrico do ponto P na reta CD:



$$t_{CD} = \frac{d_C}{d_C - d_D}$$

$$d_C = \frac{orient2d(A,B,C)}{|\overrightarrow{AB}|} \qquad d_D = \frac{orient2d(A,B,D)}{|\overrightarrow{AB}|}$$

Portanto:

$$t_{CD} = \frac{orient2d(A,B,C)}{orient2d(A,B,C) - orient2d(A,B,D)} \qquad P = C + t_{CD}(D-C)$$

Analogamente:

$$t_{AB} = \frac{orient2d(C,D,A)}{orient2d(C,D,A) - orient2d(C,D,B)} \qquad P = A + t_{AB}(B-A)$$

Fonte:
Ricardo Marques

Cruzamento de Linhas no 2D – Área Orientada de Triângulos

Segment_Segment_Intersection(u, v):

if both endpoints of u are over v then
  return false
end if
$orient2d(C,D,A) > 0$ $orient2d(C,D,B) > 0$

if both endpoints of u are under v then
  return false
end if
$orient2d(C,D,A) < 0$ $orient2d(C,D,B) < 0$

if both endpoints of v are over u then
  return false
end if
$orient2d(A,B,C) > 0$ $orient2d(A,B,D) > 0$

if both endpoints of v are under u then
  return false
end if
$orient2d(A,B,C) < 0$ $orient2d(A,B,D) < 0$

if u and v are collinear then
  return false
end if
$orient2d(C,D,A) = 0$ $orient2d(C,D,B) = 0$
ou
$orient2d(A,B,C) = 0$ $orient2d(A,B,D) = 0$

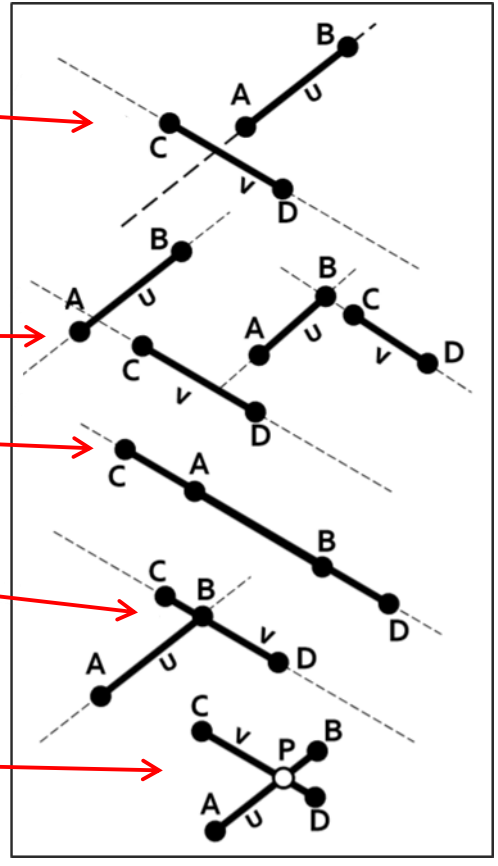
if u touches v then (there are many cases)
  return true
end if
$orient2d(C,D,B) = 0$ $orient2d(C,D,A) < 0$
$P = B$

// When get to this point, there is an intersection point

$$t_{CD} = \frac{orient2d(A,B,C)}{orient2d(A,B,C) - orient2d(A,B,D)}$$

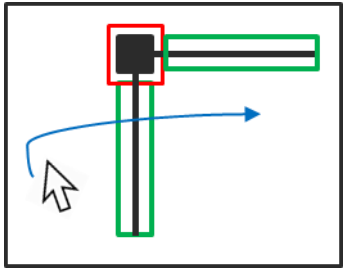
return true

$$P = C + t_{CD}(D - C)$$

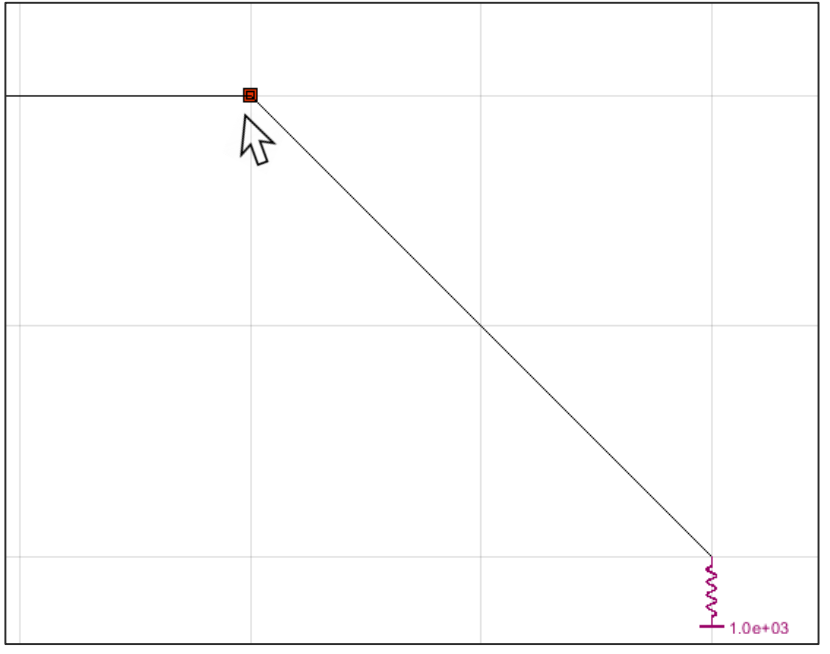


Snap de Objetos no 2D – Função *inpolygon*

- Em modelos planos, a identificação, por parte do programa, de que o cursor está próximo de um nó, elemento ou interseção se dá pelo uso da função *inpolygon*. É checado se a posição corrente do cursor está dentro de retângulos invisíveis previamente definidos, em torno de objetos gráficos.
- Foi adotada uma ordem de preferência, caso hajam diferentes objetos próximos ao cursor. Primeiramente, procura-se nós, seguidos de interseções, elementos e *grid*, respectivamente.

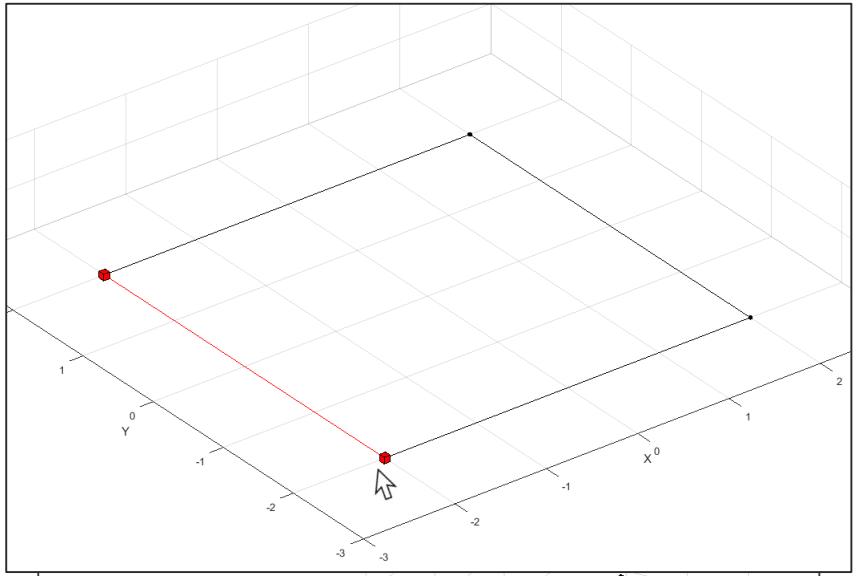


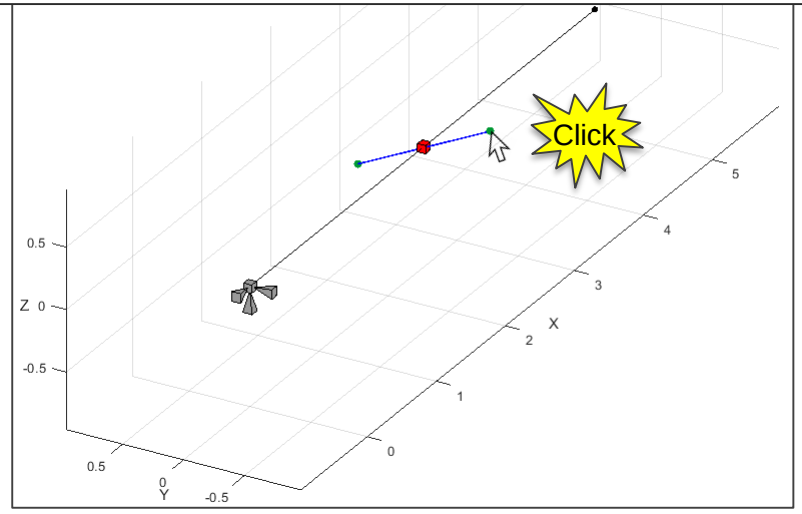
```
if inpolygon(x,y,snapX,snapY) == true
  % Resposta para objeto próximo
else
  % Resposta para cursor longe de objetos
end
```



Ponto Corrente no 3D

- No MATLAB, a propriedade *CurrentPoint* de eixos retorna as coordenadas de dois pontos, que denotam as coordenadas de entrada e saída da “caixa” que contém o *canvas*, partindo do cursor.
- Em modelos 2D, tais pontos descrevem uma reta ortogonal ao plano *XY*, logo suas coordenadas em *x* e *y* já correspondem ao ponto corrente.
- Em modelos 3D, a reta formada por tais pontos deve ser analisada para que seja obtido um ponto corrente no espaço.
- Nas análises de grelha (3D, porém com elementos apenas em *z = 0*), a modelagem em perspectiva é habilitada a partir de pontos correntes correspondentes à interseção da reta com o plano *XY*.
- Nos outros modelos espaciais, o ponto corrente é obtido com base em objetos já existentes, criados via *input* feito em diálogos auxiliares.





Cruzamento de Linhas no 3D – Linha Perpendicular

- Em modelos espaciais, as operações relativas à posição corrente do cursor trabalham com uma reta, ao invés de um ponto. Por conta disso, a tratativa de cruzamento de linhas no 3D é responsável por definir o comportamento de *snap*, seleção e plotagem de objetos.
- Foi implementado um algoritmo para a obtenção da menor reta perpendicular entre dois segmentos no espaço, cuja norma corresponde à menor distância entre os mesmos.

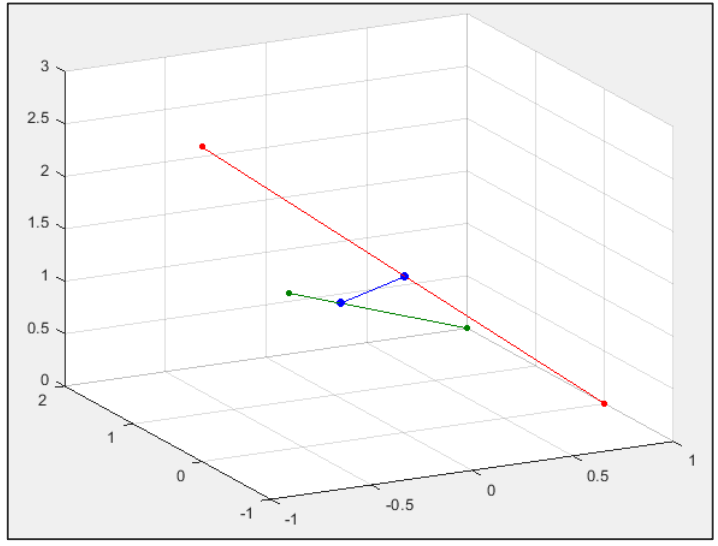
Sejam duas retas, *r_1* e *r_2*, com direção *dir_1* e *dir_2*.

$$dir_3 = dir_1 \times dir_2$$

$$A = \begin{bmatrix} dir_{1x} & dir_{2x} & dir_{3x} \\ dir_{1y} & dir_{2y} & dir_{3y} \\ dir_{1z} & dir_{2z} & dir_{3z} \end{bmatrix} \qquad d = \begin{bmatrix} r_{2xi} - r_{1xi} \\ r_{2yi} - r_{1yi} \\ r_{2zi} - r_{1zi} \end{bmatrix}$$

$$A\tau = d \quad \rightarrow \quad \boxed{\tau = A \backslash d}$$

τ é um vetor *3x1* cujos 2 primeiros termos são as coordenadas paramétricas, dentro de cada reta, dos pontos que descrevem a menor reta perpendicular entre elas. Se a norma desta for menor que uma tolerância numérica, considera-se que as retas se cruzam.



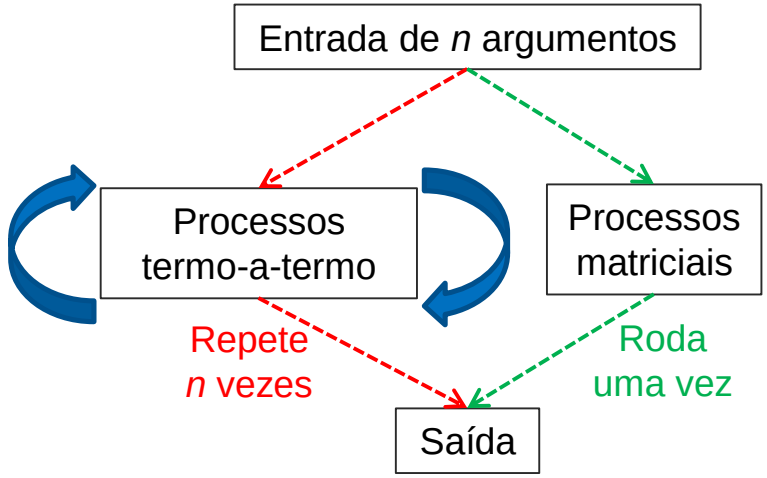
Sugestões para Eficiência com Funcionalidades Gráficas

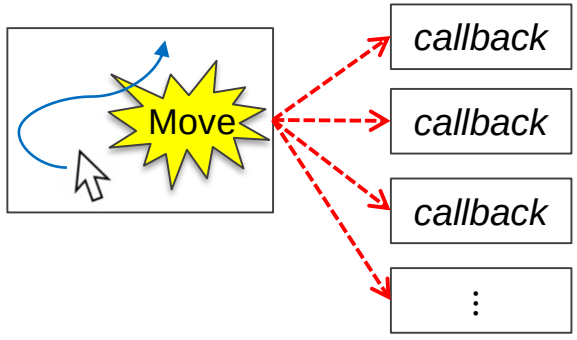
- O MATLAB é preparado para trabalhar com matrizes, por conta disso, sempre que possível, optar por operações matriciais ao invés de utilizar laços (*loops*). Gera-se ganhos de eficiência computacional, tanto para realizar cálculos, quanto para o uso de funções gráficas.

- Simplificar ao máximo possível funções *callback* de eventos sensíveis que ocorram muitas vezes, como o mover do mouse.

- Controlar plots por meio da propriedade *tag*, ou seja, nomear objetos gráficos. Essa prática garante o fácil acesso posterior a objetos gráficos, para que sejam modificados ou deletados, assim evitando que o modelo seja redesenhado por inteiro em ocasiões onde isso é desnecessário.

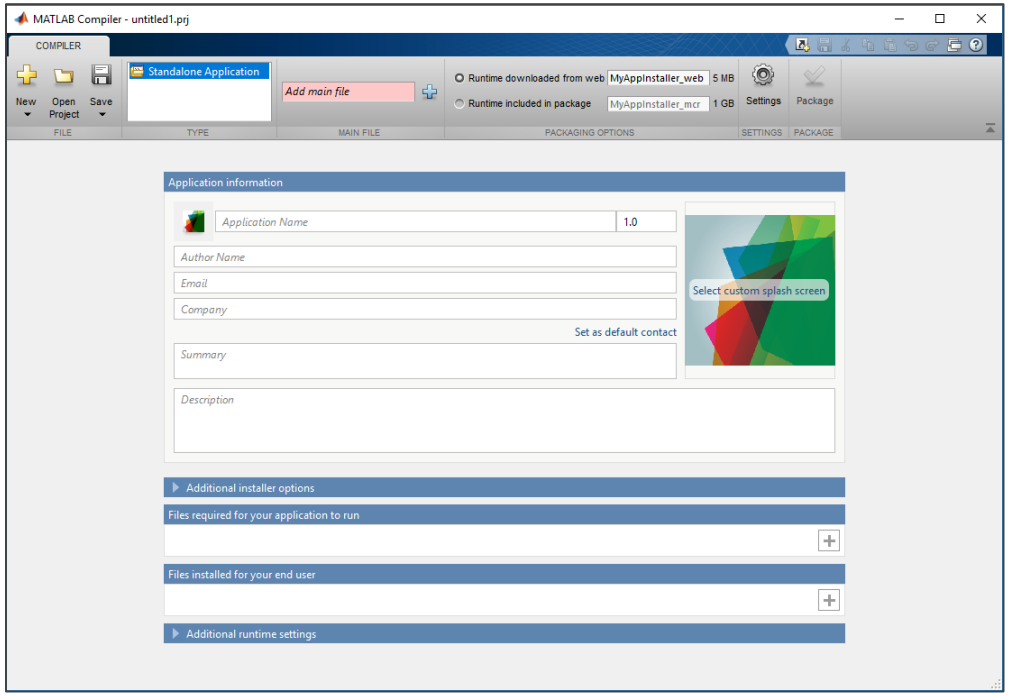
```
line(X, Y, 'Color','black','tag','drawElements')
elements = findobj('tag','drawElements');
delete(elements)
```





StandAlone e Instalador

- O MATLAB dispõe de muitos recursos extras para que sejam baixados e acrescidos ao software, como *toolboxes* e aplicações.
- A ferramenta *Application compiler* permite que programas sejam compilados em executáveis *StandAlone*, que podem ser utilizados mesmo por usuários que não possuam o MATLAB, já que disponibiliza juntamente as bibliotecas necessárias para o funcionamento do código.



Site do Programa e Arquivos Disponíveis

- Executável *StandAlone*
- Código para rodar diretamente no MATLAB
- Código fonte aberto
- Documentação
- Exemplos de modelos

https://web.tecgraf.puc-rio.br/lesm/

OPENCADD

MODEL - BASED DESIGN DRIVEN COMPANY

Modeling for life!