Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

DESENVOLVIMENTO DE INTERFACE GRÁFICA DE USUÁRIO NO PROGRAMA SCILAB PARA ANÁLISE NÃO LINEAR FÍSICA DE BARRA

DEVELOPMENT OF GRAPHIC USER INTERFACE IN THE SCILAB SOFTWARE FOR PHYSICAL NONLINEAR ANALYSIS OF BAR

André Kazuhiro Shiotani [1]
Luiz Antonio Farani de Souza [2]

Resumo: Em geral, problemas estruturais que envolvem uma análise não linear física e/ou geométrica tornam sua solução manual impraticável. Programas específicos para esses problemas não são acessíveis ao grande público, seja por inviabilidade econômica ou por questões de requisitos de poder computacional. Assim, a criação de ferramentas próprias se mostra uma alternativa visto que há diversas opções de plataformas de programação. Neste artigo, objetivou-se desenvolver uma interface gráfica de usuário por meio do *GUI Builder*, disponível no *software* livre Scilab, para a análise não linear física de um problema de barra biengastada. Verificou-se a validade dos resultados com os dados de referência, e conclui-se que interfaces gráficas facilitam a utilização de ferramentas computacionais de análise estrutural.

Palavras-chaves: *GUI Builder*, Modelo de Dano, Interface Gráfica.

Abstract: *In general, structural problems that involves a physical and/or geometrical nonlinear analysis make their hand solution unpractical. Specifics programs aren't accessible to the general public, either by their cost or by computational restraints. Therefore, the development of own tools proves to be an alternative, as there're several programming platforms options. In this paper, the objective was to develop a graphical user interface by the GUI Builder, available in the free Scilab software, for the physical nonlinear analysis of a fixed ended bar problem. The validity of the results with the reference data was verified and it was concluded that graphical interfaces facilitate the use of computational tools of structural analysis.*

Keywords: *GUI Builder, Damage Model, Graphical Interface.*

[1] *UEM, Campus Maringá – Brasil, email: andre.shiotani@outlook.com*
[2] *UTFPR, Campus Apucarana – Brasil, email: lasouza@utfpr.edu.br*

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

1 Introdução

No estudo de sistemas estruturais complexos, diversas considerações de cálculo são feitas para simplificar o problema estrutural e tornar a análise exequível sob o ponto de vista de poder computacional. Uma dessas simplificações está no comportamento não linear dos materiais e do arranjo estrutural.

Muitos problemas de engenharia são caracterizados por uma resposta não linear. A modelagem matemática de tais problemas leva a sistemas de equações não lineares que são resolvidos numericamente com procedimentos incrementais e iterativos. Efeitos não lineares de diferentes fontes podem estar presentes na modelagem matemática da resposta das estruturas de engenharia. Isso inclui as não linearidades geométricas associadas a grandes deslocamentos e grandes deformações, e respostas não lineares materiais descritas por meio de uma das muitas teorias disponíveis, tais como plasticidade, dano e fratura (MUÑOZ; ROEHL, 2017).

Desde o colapso catastrófico das torres do *World Trade Center* (WTC) em 11 de setembro de 2001, tornou-se mais evidente que compreender a falha ou o colapso progressivo em estruturas é de suma importância. Para evitar o colapso estrutural total quando partes de uma estrutura são danificadas ou destruídas, uma estrutura deve ser capaz de redistribuir esforços internos, fornecendo caminhos de carga alternativos. Para que isso ocorra, a estrutura deve ter um grau adequado de redundância, e os engenheiros precisam entender como a estrutura pode responder progressivamente sob diferentes condições de carregamento (LAM; KITIPORNCHAI; REICHL, 2010).

Muitos educadores e pesquisadores perceberam a importância de entender o comportamento estrutural e tentaram comunicar isso por meio de vários métodos. Al-Ansari e Senouci (1999), Harada (2004) e Cedeño-Rosete (2007) empregaram aplicativos de *software* como Microsoft Excel, Mathcad® e Scilab (http://www.scilab.org) para desenvolver pacotes educacionais apropriados (LAM; KITIPORNCHAI; REICHL, 2010).

Os *softwares* desenvolvidos para análises não lineares são robustos e a implementação de métodos de cálculo alternativos aos pré-programados são difíceis ou até impossíveis. Não obstante, a maioria desses *softwares* não são economicamente acessíveis para o grande público. Portanto, meios alternativos devem ser considerados para viabilizar o desenvolvimento de pesquisas acadêmicas.

A utilização e o desenvolvimento de ferramentas computacionais próprias no estudo de estruturas e materiais de comportamento não linear facilita o entendimento de rotinas de cálculo complexas e permite a verificação de diversos cenários de solicitação e geometrias diferentes em tempo reduzido.

Nesse âmbito, diversas plataformas de programação se tornaram populares e auxiliam o usuário no desenvolvimento de ferramentas próprias de análise sem que exija conhecimentos aprofundados em programação computacional. Alguns *softwares* que se destacam são: Matlab; Octave; e Scilab.

O Scilab é um pacote computacional numérico de código aberto e multiplataforma, além de ser uma linguagem de programação de alto nível orientada numericamente; sua sintaxe é amplamente baseada na linguagem Matlab. O *software* Scilab também permite que os usuários do Matlab utilizem o pacote sem problemas. Para ajudar nesse processo, há um tradutor de código embutido que ajuda o usuário a converter seu código Matlab em um código Scilab. Os principais recursos do Scilab incluem: centenas de funções matemáticas; visualização 2-D/3-D; computação numérica; análise de dados; e interface com Fortran, C, C++ e Java (SHARMA; GOBBERT, 2010).

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

Em geral, para utilizar uma rotina de cálculo em qualquer uma das plataformas supracitadas ou para implementar uma geometria diferente da pré-programada é necessário um estudo das funções desenvolvidas e das rotinas de cálculo, o que dificulta o uso das ferramentas. O desenvolvimento de interfaces gráficas de usuário (*GUI - Graphical User Interface*) se mostra como ferramenta importante para permitir o fácil acesso e manipulação de entrada e saída de dados para usuários inexperientes. Assim, este artigo tem por objetivo desenvolver uma interface gráfica de usuário na plataforma Scilab (SCILAB, 2020) para análise não linear física de um problema de barra biengastada.

2 Materiais e Métodos

2.1 Objeto do estudo

Seja o problema de uma estrutura composta por uma barra biengastada sujeita a uma força concentrada axial P, conforme ilustrado na Figura 1. Barra biengastada: (a) modelo estrutural de barra; e (b) relação constitutiva do material

. Esse problema foi proposto por Bathe (2006). A barra tem área da seção transversal A = 1,0 cm² e módulo de elasticidade longitudinal inicial $E_0 = 1{,}0 \times 10^7$ N/cm². Considera-se o comportamento material elastoplástico com encruamento positivo ($E_T = 1{,}0 \times 10^5$ N/cm²) tanto na tração quanto na compressão.

Para a solução do problema estrutural, utilizou-se o procedimento incremental e iterativo de Newton-Raphson padrão com controle de carga força constante, considerando o incremento de carga $\Delta P = 1{,}0 \times 10^4$ N e a tolerância tol = $1{,}0 \times 10^{-4}$ para o critério de convergência.

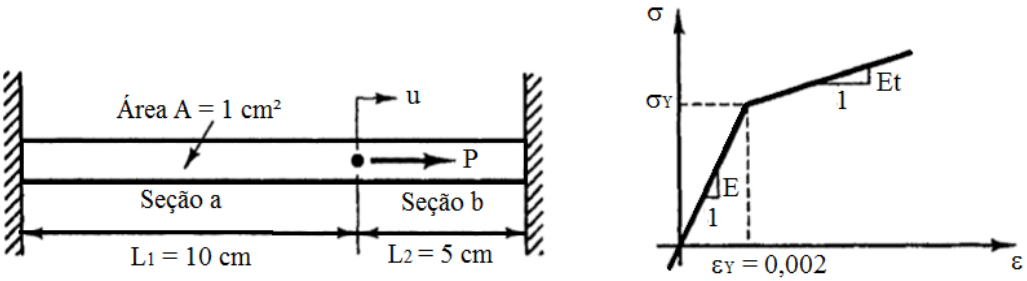

Figura 1. Barra biengastada: (a) modelo estrutural de barra; e (b) relação constitutiva do material

O carregamento é aplicado de forma lenta (análise estática), cujas deformações e deslocamentos sejam pequenos o suficiente (hipótese de deformações infinitesimais) para determinar a resposta da estrutura considerando apenas a não linearidade física. Os resultados numéricos obtidos por Bathe (2006) para esse problema estão apresentados na **Erro! Fonte de referência não encontrada.**.

Tabela 1. Resultados numéricos obtidos por Bathe (2006)

Força P (kN)	Deslocamento u (m)
20	$6{,}6667 \times 10^{-3}$
40	$1{,}9269 \times 10^{-2}$

Fonte: Bathe (2006)

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

2.2 Modelo de Dano

O comportamento do material é baseado na teoria da Mecânica do Dano, e utiliza-se o modelo de dano isotrópico proposto por Manzoli et al. (1998). Para o caso de um elemento unidimensional submetido a uma força axial, a tensão efetiva $\bar{\sigma}$ é expressa pela Equação (1) (SOUZA; SANTOS; KAWAMOTO, 2019).

$$\bar{\sigma} = E_0 \varepsilon, \qquad (1)$$

na qual E_0 é o módulo de elasticidade não danificado do material e ε é a deformação específica. O limite de dano inicial r_0 é uma propriedade do material e pode ser relacionado à tensão de escoamento σ_Y, conforme apresentado pela Equação (2).

$$r_0 = \frac{\sigma_Y}{\sqrt{E_0}}. \qquad (2)$$

A norma da tensão efetiva $\tau(\bar{\sigma})$ é definida pela Equação (3).

$$\tau(\bar{\sigma}) = \sqrt{\frac{\bar{\sigma}^2}{E_0}}. \qquad (3)$$

Por sua vez, o critério de degradação $f(\bar{\sigma})$ é expresso pela Equação (4).

$$f(\bar{\sigma}) = \tau(\bar{\sigma}) - r_0 \leq 0. \qquad (4)$$

A partir das relações de Kuhn - Tucker, o limite de dano é dado pelo máximo valor da variável τ durante o processo de carregamento de modo a se obter a Equação (5).

$$r = \text{máx}(r, \tau). \qquad (5)$$

Combinando as Equações (1) e (3), τ pode ser expresso em termos da deformação ε - vide Equação (6).

$$\tau(\varepsilon) = \sqrt{E_0}\, \varepsilon. \qquad (6)$$

A tensão normal σ na barra é obtida por meio da Equação (7).

$$\sigma = (1 - d) E_0 \varepsilon, \qquad (7)$$

na qual o dano d é um escalar escrito em função do módulo H e do limite de dano r, conforme exposto pela Equação (8).

$$d = \frac{r - r_0}{r(1 + H)}. \qquad (8)$$

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

Variando-se a função que descreve o módulo H, tem-se na Figura 2 em (a) o regime elastodegradável perfeito, em (b) o encruamento linear positivo (endurecimento), em (c) o encruamento linear negativo (abrandamento) e em (d) o abrandamento exponencial.

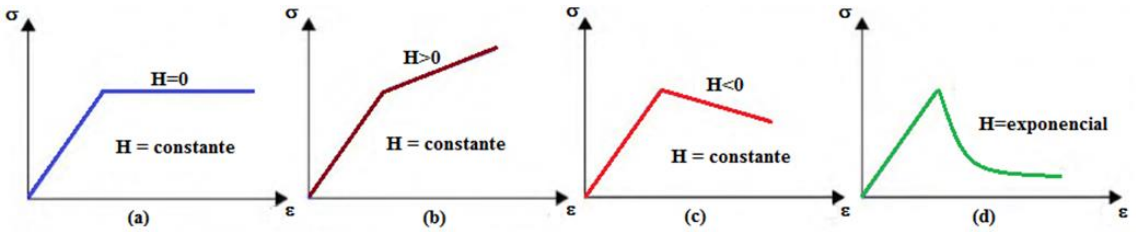

Figura 2. Comportamentos distintos de endurecimento/abrandamento
Fonte: Souza, Santos e Kawamoto (2019)

2.3 *GUI Builder*

Para o desenvolvimento da interface gráfica, utilizou-se o módulo de interface gráfica *GUI Builder* (*Graphic User Interface Builder*) (LUH; VIOLEAU, 2017), disponível no programa Scilab (SCILAB, 2020) no gerenciador de módulos ATOMS (Figura 3). O *GUI Builder* encontra-se na versão 4.2.1 e foi desenvolvido por Tan Chin Luh e David Violeau. Para instalá-lo, o usuário deve acessar o menu "Aplicativos → Gerenciador de módulos - ATOMS → *GUI* → *GUI Builder*" e pressionar "Instalar". Para iniciar o aplicativo, o operador deve digitar "guibuilder" no *console* e executar com a tecla "enter".

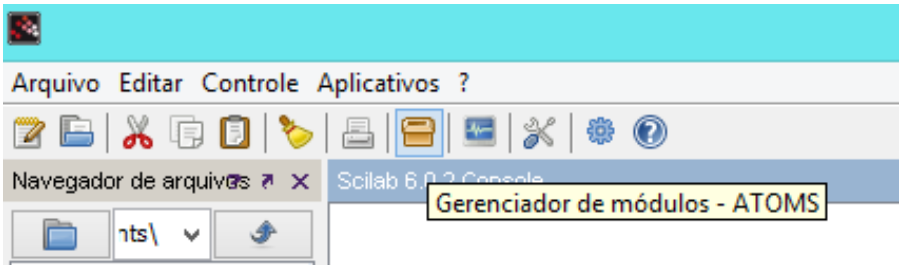

Figura 3. Gerenciador de módulos - ATOMS no menu aplicativos
Fonte: Os Autores (2020)

Uma *GUI* é uma interface humano-computador que utiliza objetos gráficos como janelas, menus e ícones para interagir com os usuários por meio do uso de mouse e teclado (geralmente de maneira limitada). A principal vantagem do uso de *GUIs* é a capacidade de tornar as operações no computador mais intuitivas e, portanto, mais fáceis de aprender e usar, especialmente para novos usuários. Nessa direção, é muito mais fácil mover arquivos arrastando ícones com o mouse do que lembrar o comando que executa a mesma tarefa.

O procedimento típico de projeto de uma *GUI* consiste em: criar a janela gráfica; adicionar objetos de controle à janela nas posições desejadas; atribuir propriedades a cada objeto de controle; escrever funções de retorno de chamada para que a *GUI* execute a tarefa desejada; adicionar um menu, caso houver; e testar o projeto (BHALEKAR, 2016). A função do Scilab *uicontrol* é utilizada para criar um objeto *GUI*, cuja sintaxe é: h = *uicontrol*(*parent*, *PropertyName*, *PropertyValue*, ...). A janela de ferramentas, a janela de criação da interface gráfica e um exemplo de código gerado automaticamente pelo *GUI Builder* estão ilustrados nas Figuras 4 a 6.

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

Na Figura 6 é possível verificar que todas as propriedades dos elementos gráficos foram criadas automaticamente, incluindo a posição, o tamanho e a etiqueta de chamada para posterior uso do programador. Nessa figura, nota-se também que ao inserir elementos do tipo botão, as chamadas das funções também são geradas, bastando o usuário inserir as rotinas desejadas.

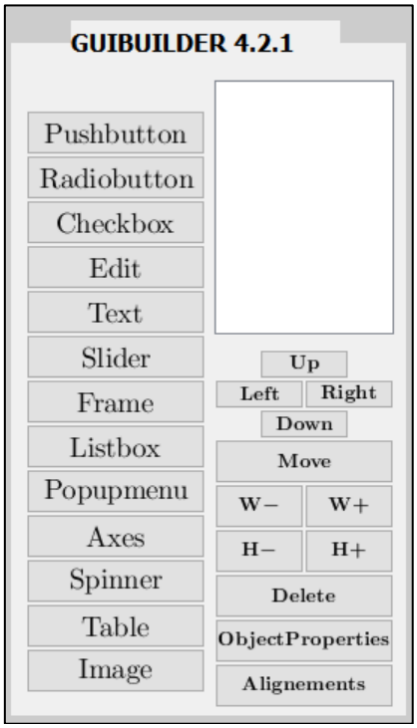

Figura 4. Janela de ferramentas disponíveis no *GUI Builder* 4.2.1
Fonte: Os Autores (2020)

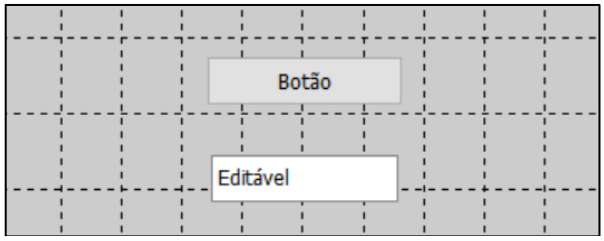

Figura 5. Janela de criação da interface gráfica
Fonte: Os Autores (2020)

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

```
// This GUI file is generated by guibuilder version 4.2.1
//////////
f=figure('figure_position',[400,50],'figure_size',[378,264],'auto_resize','on','background',[33],'figure_name','Janela gráfica número %d','dockable','off','infobar_visible','off','toolbar_visible','off','menubar_visible','off','default_axes','on','visible','off');
//////////
handles.dummy = 0;
handles.teste=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.3417949,0.6186364,0.3283974,0.135],'Relief','default','SliderStep',[0.01,0.1],'String','Botão','Style','pushbutton','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','teste','Callback','teste_callback(handles)')
handles.edit=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.3494231,0.3677273,0.3111538,0.1286364],'Relief','default','SliderStep',[0.01,0.1],'String','Editável','Style','edit','Value',[0],'VerticalAlignment','middle','Visible','on','Tag','edit','Callback','')


f.visible = "on";


//////////
// Callbacks are defined as below. Please do not delete the comments as it will be used in coming version
//////////

function teste_callback(handles)
//Write your callback for  teste  here

endfunction

```

Figura 6. Código gerado automaticamente pelo *GUI Builder*
Fonte: Os Autores (2020)

2.4 Desenvolvimento da interface gráfica

Foram desenvolvidas duas *GUIs*, a primeira com o menu (Figura 7), que apresenta as opções de *GUI* com as rotinas pré-programadas, e a segunda (Figura 8), dedicada especificamente ao problema da barra biengastada proposto por Bathe (2006).

A interface de resolução do problema conta com um controle *edit* para entrada de dados para cada um dos parâmetros e um controle *text* de texto que apresenta uma figura ilustrativa do problema analisado. A inserção da imagem na interface é possível ao se modificar o valor da propriedade *string* do controle para o nome do arquivo de imagem seguido de sua extensão, conforme é destacado na Figura 9.

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

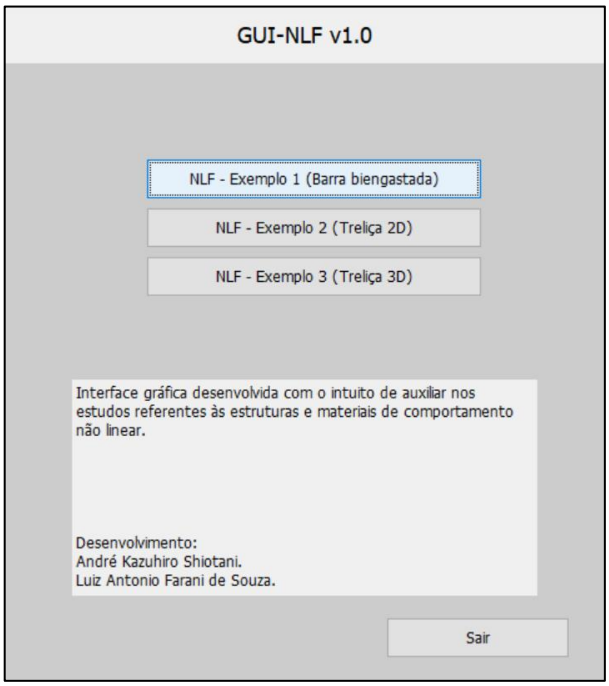

Figura 7. Interface Menu
Fonte: Os Autores (2020)

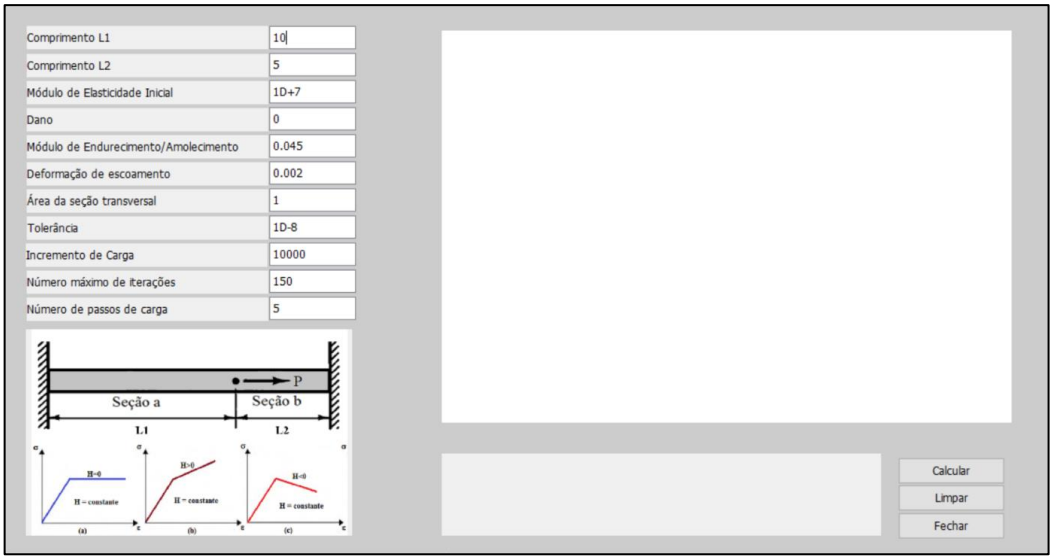

Figura 8. Interface de resolução do problema da barra
Fonte: Os Autores (2020)

```
handles.exemplo=uicontrol(f,'unit','normalized','BackgroundColor',[-1,-1,-1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[1],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Position',[0.0205357,0.0928571,0.3125,0.2064626],'Relief','default','SliderStep',[0.01,0.1],'String','Exemplo1.png','Style','image','Value',[0.2,0.2,0,0,0],'VerticalAlignment','middle','Visible','on','Tag','exemplo','Callback','exemplo_callback(handles)')
```

Figura 9. Inserção de imagem no controle *edit*
Fonte: Os Autores (2020)

Além disso, na interface foi inserido um controle *edit* para a apresentação do gráfico deslocamento u *versus* carga P (trajetória de equilíbrio) e um controle *text* para a apresentação

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

do número total de iterações acumuladas até o fim da simulação, para uma dada tolerância. Para a resolução do problema foram criadas três funções (sub-rotinas), sendo elas: a função "NLF", que executa a análise não linear física da barra; a função "dano", em que foi implementado o modelo de dano isotrópico; e a função "result", na qual foram inseridos os resultados numéricos obtidos por Bathe (2006). O código computacional da função "NLF" é apresentado na Figura 10.

```
function start_callback(handles)
//Write your callback for start  here
limpar_callback(handles);
exec('dano.sci',0);
exec('result.sci',0);

//Dados de entrada (pré-processamento)
L1=get(handles.L1,'string');
L(1)=strtod(L1); //Comprimento indeformado
L2=get(handles.L2,'string');
L(2)=strtod(L2);
E0gui=get(handles.E0,'string');
E0valor=strtod(E0gui);
dgui=get(handles.d,'string');
dvalor=strtod(dgui);
Hgui=get(handles.H,'string');
Hvalor=strtod(Hgui);
eygui=get(handles.ey,'string');
eyvalor=strtod(eygui);
Agui=get(handles.A,'string');
Avalor=strtod(Agui);
for i=1:2
   E0(i)=E0valor; //Módulo de elasticidade inicial
   d(i)=dvalor; //Dano inicial
   H(i)=Hvalor; //Módulo de
endurecimento/amolecimento
   ey(i)=eyvalor; //Deformação de escoamento
   Sy(i)=10^7*ey(i); //Tensão de escoamento
   A(i)=Avalor; //Área da seção transversal da barra
   r0(i)=Sy(i)/sqrt(E0(i)); //Limite de dano inicial
end
E=E0;
r=r0;
tolgui=get(handles.tol,'string');
tol=strtod(tolgui); //Tolerância
Frgui=get(handles.Fr,'string');
Fr=strtod(Frgui); //Incremento de carga
kmaxgui=get(handles.kmax,'string');
kmax=strtod(kmaxgui); //Número máximo de
iterações
nmaxgui=get(handles.nmax,'string');
nmax=strtod(nmaxgui); //Número de passos de carga

//Processamento
u=0;
Fint=0;
vu(1)=0;
vf(1)=0;
ktotal=0;
```

Continuação

```
tic()
for lambda=1:nmax //Passos de carga
   Fext=lambda*Fr;
   k=0;
   while k<=kmax //Ciclo iterativo
      k=k+1; //Contador (iteração)
      for i=1:2
         Kel(i)=E(i)*A(i)/L(i); //Matriz de rigidez elementar
      end
      K=Kel(1)+Kel(2); //Matriz de rigidez global
      g=lambda*Fr-Fint; //Vetor de forças desequilibradas
      du=K\g; //Solução do sistema (Vetor subincremento de
deslocamentos)
      u=u+du;  //Vetor de deslocamento
      for i=1:2
         e(i)=u/L(i); //Deformação específica de engenharia
         [E,r,S,d]=dano(i,e,E0,d,H,r,r0,E);
         Fel(i)=A(i)*S(i); //Vetor de força interna elementar
      end
      Fint=Fel(1)+Fel(2); //Vetor de forças internas global
      g=Fext-Fint; //Vetor de forças desequilibradas
      if abs(g)<=tol //Critério de convergência
         then
         break
      end
   end
   vu(lambda+1)=u; //Vetor armazena o deslocamento
   vf(lambda+1)=Fext; //Vetor armazena a força
   ktotal=ktotal+k; //Acumula as iterações
end
toc()//

//Saída de dados (pós-processamento)
//Gráfico trajetória de equilíbrio
[v1,vf1]=result();
plot(v1,vf1,'blacks','markersize',9);
set(gca(),"auto_clear","off");
plot(vu,vf,'s-*');
gca().grid=[1 1 1];  //Linhas de grade
xlabel('Deslocamento u (cm)'); //Eixo x
ylabel('Força P (N)'); //Eixo y
legend('Bathe (2006)','Modelo de Dano',2); //Legenda
//Número de iterações acumuladas até o final da simulação
t.text=string(' Número total de iterações: '+string(ktotal))
set(handles.texto_saida,'string',t.text);

endfunction
```

Figura 10. Código computacional da função "NLF"
Fonte: Os Autores (2020)

O código computacional da função "dano" é apresentado na Figura 11. Essa função verifica o comportamento não linear do material com o acréscimo de tensão utilizando o

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

método do dano isotrópico (descrito na Seção 2.3). Como a interface permite a alteração do coeficiente de dano e do módulo de endurecimento/amolecimento, é possível verificar o comportamento de materiais com encruamento positivo (H > 0), encruamento negativo (H < 0) e perfeitamente plástico (H = 0).

```
function [E, r, S, d]=dano(i, e, E0, d, H, r, r0, E)
  tal=sqrt(E0(i))*abs(e(i));
  fy=tal-r(i);
  if fy<=0
    S(i)= E(i)*e(i);
  else
    r(i)=tal;
    d(i)=(r(i)-r0(i))/(r(i)*(1+H(i)));
    E(i)=(1-d(i))*E0(i);
    S(i)= E(i)*e(i);
  end
  if d(i)<0
    d(i)=0;
  end
  if d(i)>1
    d(i)=1;
  end
endfunction
```

Figura 11. Código computacional da função "dano"
Fonte: Os Autores (2020)

Os resultados numéricos obtidos por Bathe (2006) para o problema estudado estão na função “result” (Figura 12).

```
function [v1, vf1]=result()
//Resultados de Bathe (2006)
v1=[6.6667*10^-3;
1.9269*10^-2];
vf1=[2*10^4;
4*10^4];
endfunction
```

Figura 12. Código computacional da função "result"
Fonte: Os Autores (2020)

3 Resultados e discussões

Após a verificação e validação das rotinas e comparação com os valores de referência de Bathe (2006), as interfaces gráficas foram desenvolvidas para realizar a entrada de dados e apresentação dos resultados. Na Figura 13 nota-se que os resultados obtidos pela ferramenta desenvolvida coincidem com os valores de referência. A Figura 14 apresenta em detalhe a trajetória de equilíbrio juntamente com pontos de equilíbrio obtidos por Bathe (2006).

A utilização do *GUI Builder* facilita a criação de ferramentas gráficas, no entanto, toda a programação e configuração de entrada dos dados e apresentação dos resultados dependem do programador. Apesar de a linguagem computacional ser relativamente simples, ainda assim, é necessário que o usuário convencional do Scilab se acostume com o uso de controles gráficos e suas propriedades.

Além disso, é evidente que a plataforma Scilab ainda não permite, nativamente, a criação de *GUIs*, o que gera diversos problemas durante a programação das interfaces e impõe algumas limitações ao programador.

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

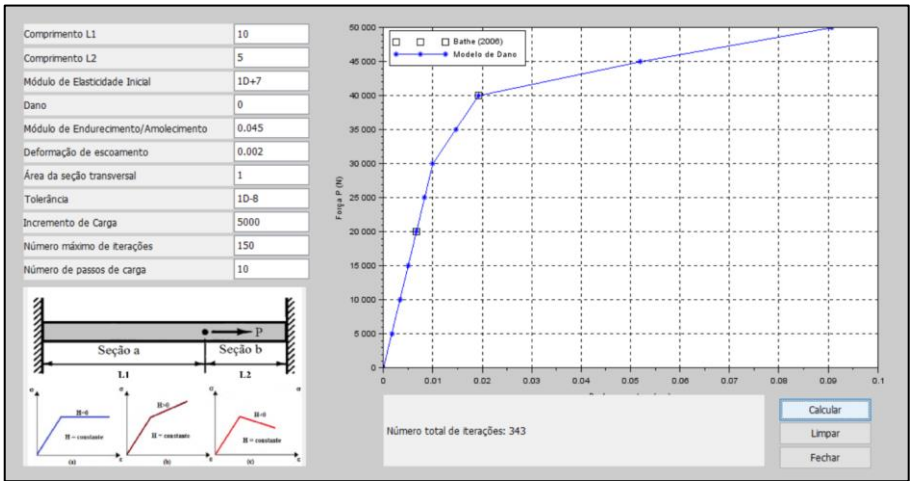

Figura 13. Demonstração da resolução do problema proposto
Fonte: Os Autores (2020)

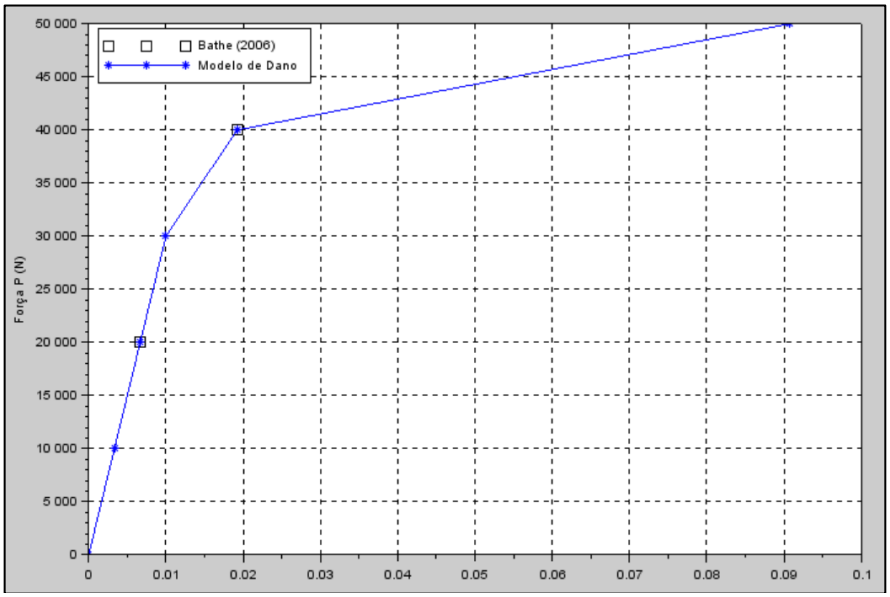

Figura 14. Resultados obtidos pela ferramenta em comparação com os valores obtidos por Bathe (2006)
Fonte: Os Autores (2020)

4 Conclusões

Neste artigo foi desenvolvida uma interface gráfica de usuário para a solução de um problema de uma barra biengastada com não linearidade física na plataforma Scilab, e os resultados numéricos foram validados com os valores numéricos de Bathe (2006).

Conclui-se que a criação de *GUIs* se torna mais simples pelo uso de módulos apropriados, tais como o *GUI Builder*, e que sua implementação possibilita que usuários menos experientes possam interagir com os programas de maneira mais rápida, sem que exija dos mesmos qualquer conhecimento de programação.

A escolha do *software* Scilab para o desenvolvimento da ferramenta gráfica se deu por dois motivos: é um *software* distribuído gratuitamente via internet, cujos usuários podem obter e usá-lo em seus próprios PCs com muita facilidade; e é fácil de executar operações matriciais no ambiente Scilab com sua linguagem de programação de alto nível, sendo adequado para a análise estrutural.

Revista Tecnológica - Universidade Estadual de Maringá - ISSN 1517-8048
DOI: 10.4025/revtecnol.v29i2.52674

Por fim, salienta-se que existe um potencial muito grande na criação de ferramentas gráficas para fins educacionais, exigindo do programador apenas um pequeno esforço de programação das interfaces.

Referências

AL–ANSARI, M. S.; SENOUCI, A. B. Use of Mathcad as a teaching and learning tool for reinforced concrete design of footings. **Computer Applications in Engineering Education**, v. 7, n. 3, p. 146-154, 1999.

BATHE, K. J. **Finite Element Procedures.** Prentice Hall, 2006.

BHALEKAR, S. **Graphical User Interface (GUI) in Scilab**. Department of Mathematics, Shivaji University, Kolhapur, 2016. Acessado em 17/03/2020. Web Page https://www.researchgate.net/publication/303592049_Graphical_User_Interface_GUI_in_Scilab

CEDEÑO-ROSETE, R. Stiffness matrix structural analysis educational package by Mathcad. **Computer Applications in Engineering Education**, v. 15, n. 2, p. 107-112, 2007.

HARADA, Y. Development of courseware for introduction of nonlinear frame analysis using free scientific software package. **Computer Applications in Engineering Education**, v. 12, n. 4, p. 224-231, 2004.

LAM, H. F.; KITIPORNCHAI, S.; REICHL, T. Development of a virtual testing application for the teaching and learning of structural engineering. **The IES Journal Part A: Civil & Structural Engineering**, v. 3, n. 2, p. 119-130, 2010.

LUH, T. C.; VIOLEAU, D. **GUI Builder**. Versão 4.2.1. Trity Technogy, Lycee Saint Louis, 2017.

MANZOLI, O.; OLIVER, J.; RUIZ, M. C. **Un Modelo analítico y numérico para la simulación de discontinuidad fuertes en la mecánica de sólidos**. Barcelona: Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE), 1998.

MUÑOZ, L. F. P.; ROEHL, D. A continuation method with combined restrictions for nonlinear structure analysis. **Finite Elements in Analysis and Design**, v. 130, p. 53-64, 2017.

SCILAB. Versão 6.1.0. **ESI Group**, 2020.

SHARMA, N.; GOBBERT, M. K. **A comparative evaluation of Matlab, Octave, FreeMat, and Scilab for research and teaching**. UMBC Faculty Collection, 2010.

SOUZA, L. A. F.; SANTOS, D. F.; KAWAMOTO, R. Y. M. Análise não linear física de treliças com ciclos de carregamento e descarregamento. **Revista Tecnológica**, v. 28, n. 1, p. 101-118, 2019.