MATLAB® Orientado a Objetos na Análise de Estruturas via Formulação Não-Convencional do MEF

RELATÓRIO FINAL – EDTIAL 01/2018

SÃO BERNARDO DO CAMPO

2018

Resumo. *Este Projeto de Pesquisa desenvolveu uma interface gráfica através da análise numérica de uma barra engastada (problema clássico mecânico estrutural unidimensional elástico) via Método dos Elementos Finitos Generalizados Estável (MEFGE) na Formulação Híbrido-Mista de Tensão (FHMT), denominada FHMT-MEFGE. A simulação numérica deste problema foi feita a partir do Método dos Elementos Finitos (método clássico) e dos métodos não-convencionais até a forma FHMT-MEFGE, via MATLAB®. A junção destes dois métodos foi fundamental para a determinação da solução aproximada em deslocamento e tensão, uma vez que a FHMT apresenta três campos independentes de aproximação (tensão e deslocamento no domínio e deslocamento no contorno) e o MEFGE apresenta o campo de deslocamento no domínio. Apesar da solução obtida por meio deste método apresentar uma solução inferior a solução obtida pela FHMT-MEFG, vale salientar que para o sistema proposto o condicionamento da matriz é superior ao utilizado em sistemas MEFG. Através destas análises a implementação da interface gráfica se deu por meio do MEF e da FHMT, uma vez que a implementação através de métodos não clássicos não foi concluída nessa pesquisa de iniciação científica.*

Palavras-Chave: *Elementos Finitos, Elementos Finitos, Formulação Híbrido-Mista de Tensão, Interface Gráfica MATLAB.*

Sumário

1 INTRODUÇÃO

Desde o surgimento do Método dos Elementos Finitos (MEF) para análise de problemas do âmbito da Mecânica das Estruturas, no ano de 1956, várias formas não-convencionais do MEF foram desenvolvidas no sentido de superar limitações apresentadas pela versão clássica em deslocamentos do MEF como, por exemplo, na simulação dos problemas de propagação de trincas. Aqui, especificamente, serão destacadas as formas não-convencionais do MEF denominadas de Método dos Elementos Finitos Generalizados (MEFG), Método dos Elementos Finitos Generalizados Estável (MEFGE), Formulação Híbrido-Mista de Tensão (FHMT), a aplicação do Métodos dos Elementos Finitos Generalizados na FHMT (FHMT-MEFG) e a aplicação do Métodos dos Elementos Finitos Generalizados Estável na FHMT (FHMT-MEFGE) com o objetivo de aplicá-las na solução do problema unidimensional de barra sob força normal.

Especificamente, o MEFG e o MEFGE são alternativas numéricas obtidas basicamente da composição de características positivas da forma convencional em deslocamentos do MEF e dos Métodos sem Malha - Babuška, Caloz e Osborn (1994), Duarte e Oden (1995, 1996), Babuška e Melenk (1997), Duarte (1996), Melenk e Babuška (1996), Oden, Duarte e Zienkiewicz (1998), Duarte, Babuška e Oden (2000), Strouboulis, Copps e Babuška (2000), Babuška e Banerjee (2012), Gupta et al. (2013) e Lins (2015).

No MEFG e MEFGE uma malha de cobertura é empregada para definir nuvens suporte dentro das quais as funções de forma do MEF clássico são usadas como Partição da Unidade (PU). Estas funções de forma podem então ser enriquecidas dentro de cada nuvem, mediante a multiplicação da PU por funções de interesse (polinomiais ou não); o enriquecimento resultante tem caráter nodal, Duarte, Babuška e Oden (2000), Babuška e Banerjee (2012), Gupta et al. (2013) e Lins (2015).

O enriquecimento nodal, resumidamente, pode ser definido como a ampliação das bases aproximativas envolvidas, sem a necessidade de introduzir novos pontos nodais no domínio. Vale ressaltar que existe uma pequena diferença entre as metodologias de enriquecimento nodal aplicada ao MEFG e MEFGE, Babuška e Banerjee (2012). Especificamente, na figura 01, apresenta-se a metodologia de enriquecimento nodal utilizada no MEFG. Essa estratégia difere, portanto, do processo p-adaptativo do MEF clássico, que, para certa classe de elementos finitos, exige a inserção de pontos nodais extras nos elementos para o aumento do grau de interpolação, Duarte e Oden (1995, 1996).

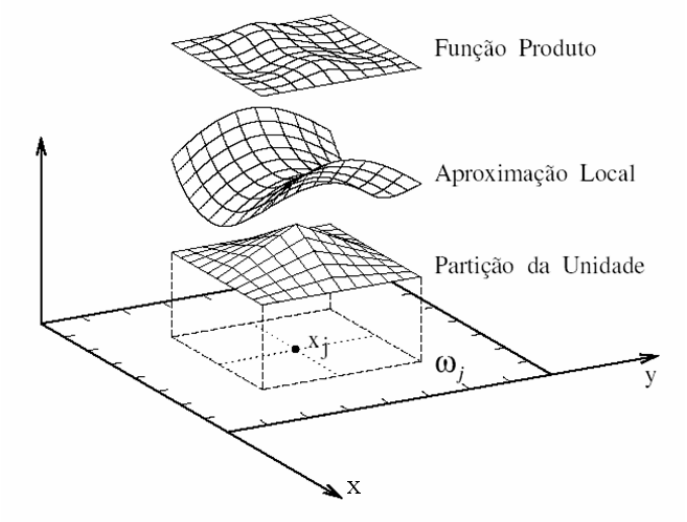

Figura 01 – Enriquecimento da Partição da Unidade num domínio global ***xy*** - MEFG.

Na figura 02, destaca-se a metodologia de enriquecimento nodal utilizada no MEFGE, Gupta et al. (2013). A diferença da aplicação da técnica de enriquecimento nodal no MEFG e MEFGE está na definição da própria função enriquecedora. Essa diferença será explicitada posteriormente.

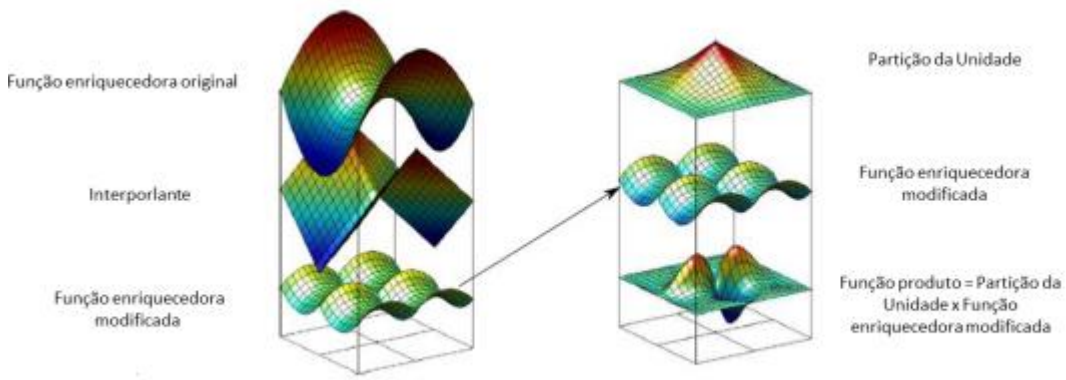

Figura 02 – Enriquecimento da Partição da Unidade num domínio global discretizado – MEFGE, adaptado de Gupta et al. (2013).

Já a formulação Híbrido-Mista de Tensão (FHMT), Góis (2009), Góis e Proença (2012) e Hass e Góis (2016), diferentemente das formulações do MEF, MEFG e MEFGE, que aproximam exclusivamente o campo de deslocamento, envolve aproximações simultâneas de três campos - tensões e deslocamentos no domínio e deslocamentos no contorno do problema. Assim, na FHMT duas malhas de cobertura – uma malha de domínio e outra de contorno - são necessárias para a discretização de um problema em elementos finitos.

Em Góis (2009), Góis e Proença (2012) e Hass e Góis (2016) propõem-se a aplicação da técnica de enriquecimento nodal (com funções polinomiais e trigonométricas) do MEFG na FHMT, possibilitando assim a FHMT-MEFG. Ainda neste trabalho será apresentada, de forma original, a aplicação do enriquecimento nodal do MEFGE na FHMT, constituindo a FHMT-MEFGE.

Espera-se que a solução numérica do problema de barra sob força normal, com estas formulações não-convencionais do MEF, via MATLAB®, permita um entendimento mais claro da aplicação destes métodos numéricos no âmbito de problemas da Mecânica Estrutural.

Na sequência do texto, a apresentação do problema de barra sob força normal, em sua forma dita forte, será desenvolvido no item 2. As relações fundamentais do MEFG, MEFGE,

FHMT, FHMT-MEFG e FHMT-MEFGE aplicadas ao problema base são destacadas, respectivamente, nos itens 3 e 4. No item 5 destacam-se as simulações numéricas desenvolvidas. Finalmente, no item 6, resumem-se as principais conclusões deste trabalho.

2 O PROBLEMA DE BARRA SOB FORÇA NORMAL

A apresentação a seguir utiliza, basicamente, a estrutura do problema de barra sob força normal ilustrada em Brasil, Balthazar e Góis (2015).

Assim, considere x um domínio unidimensional e $u(x)$ a função que descreve o deslocamento de todos os pontos do domínio da barra prismática apresentada na figura 03. Esta barra possui seção transversal A, está engastada em $x = 0$, tem comprimento $L, L >> A$ e submetida à carga uniformemente distribuída q (força por unidade de comprimento) em $0 \leq x \leq L$.

Admite-se ainda que no contorno $x = L$ existe uma força P (unidade de força) aplicada e que as forças q e P são normais à seção transversal A. Considere que o material constituinte da barra é homogêneo, elástico linear, isótropo e que o módulo de elasticidade longitudinal vale E (força por unidade de comprimento ao quadrado). Na modelagem em questão, considera-se também o equilíbrio imposto na configuração inicial do sistema, pequenos deslocamentos e giros das seções transversais.

O objetivo principal de qualquer problema mecânico estrutural é a determinação das tensões (força por unidade de comprimento ao quadrado), deformações e deslocamentos (unidade de comprimento) em todos os pontos do domínio de uma dada estrutura sob carregamento qualquer e vinculações impostas. No caso do problema de barra sob força normal, apresentado na figura 03, tem-se da Teoria da Elasticidade que os campos de tensão $(\sigma(x))$, deformação $(\varepsilon(x))$ e deslocamento $(u(x))$ variam no domínio da barra, $0 \leq x \leq L$, exclusivamente em função de x.

O equilíbrio de forças verticais de um segmento de comprimento dx da barra é

$$\frac{d\sigma(x)}{dx} = \frac{-q}{A} \tag{1}$$

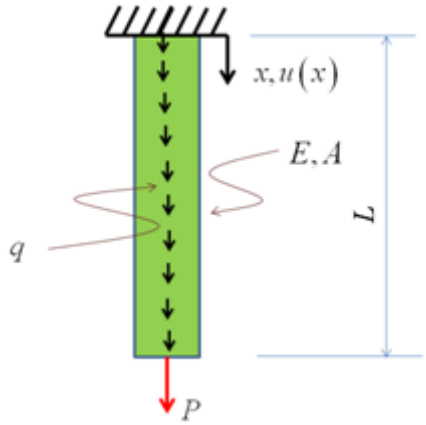

Figura 03 – Barra sob força normal – problema mecânico estrutural

Da Teoria da Elasticidade, pode-se definir a relação de compatibilidade entre a deformação $(\varepsilon(x))$ e o deslocamento $(u(x))$ por:

$$\varepsilon(x) = u'(x) \tag{2}$$

Para materiais elásticos lineares e isótropos, a relação tensão-deformação é dada pela Lei de Hooke

$$\sigma(x) = E\varepsilon(x) \tag{3}$$

Substituindo a Eq. (2) na Eq. (3) tem-se

$$\sigma(x) = Eu'(x) \tag{4}$$

Das Eqs. (1) e (4), pode-se escrever:

$$u''(x) = \frac{-q}{E \cdot A} \tag{5}$$

A Eq. (5) representa a equação diferencial (forma forte) do problema de barra sob força normal ilustrado na figura 03. Para a determinação da função deslocamento $(u(x))$ são necessárias duas condições de contorno.

A primeira condição de contorno em deslocamentos (condição de contorno essencial) é $u(0)=0$.

No segundo contorno do problema $(x=L)$ tem-se que a função força normal $N(x)$ vale P, então:

$$N(L)=P \tag{6}$$

Para o problema de barra sob força normal, as tensões normais $(\sigma(x))$ se relacionam com as forças normais $N(x)$ da seguinte forma:

$$\sigma(x)=\frac{N(x)}{A} \tag{7}$$

Especificamente, no contorno $(x=L)$

$$\sigma(L)=\frac{N(L)}{A}=\frac{P}{A} \tag{8}$$

A Eq. (4) avaliada em $(x=L)$ é

$$\sigma(L)=Eu'(L) \tag{9}$$

As Eqs. (8) e (9) são combinadas para a definição da condição de contorno em $x=L$ explicitada em função dos deslocamentos, ver Eq. (10).

$$u'(L)=\frac{P}{E\cdot A} \tag{10}$$

O Problema de Valor de Contorno (PVC) em questão - problema de barra sob força normal - descrito exclusivamente na variável deslocamento $(u(x))$, é definido pela Eq. (5) e condições de contorno $u(0)=0$ e $u'(L)=\frac{P}{E\cdot A}$.

Explicitamente, define-se este PVC da seguinte forma:

Dados q, P, L, E e A, determinar $u(x)$, tal que:

$$u''(x)+\frac{q}{E\cdot A}=0 \text{ em } 0\le x\le L \tag{11}$$

$$u(0)=0 \text{ (condição de contorno essencial – Dirischlet)} \tag{12}$$

$$u'(L)=\frac{P}{E\cdot A} \qquad \text{(condição de contorno natural – Neuman)} \tag{13}$$

As Eqs (11), (12) e (13) caracterizam a forma forte do PVC exposto.

Além de satisfazer as condições de contorno (Eqs. (12) e (13)) a função solução $u(x)$ deve, obrigatoriamente, ser contínua até a sua segunda derivada (ver Eq (11)). O termo "forma forte" decorre da característica que a Eq (11) deve ser satisfeita para todo o domínio de x definido, ou seja, $0\le x\le L$. Já na forma variacional "fraca", necessária para aplicação do MEF, essa exigência não é mantida, uma vez que, como será mostrada a seguir, a Eq (11) deve ser satisfeita num sentido de média ponderada.

3 MEF, MEFG E MEFGE APLICADOS AO PROBLEMA DE BARRA SOB FORÇA NORMAL

Como o MEFG e o MEFGE combinam a estrutura do MEF clássico com a técnica de enriquecimento nodal, nesta etapa do trabalho serão apresentadas à metodologia para construção das bases de aproximação do MEF e, posteriormente, as formas de enriquecimento, exclusivo do campo de deslocamento $u(x)$ do domínio, do MEFG e MEFGE. Já no próximo item, as formas integrais necessárias para o desenvolvimento da FHMT, FHMT-MEFG e FHMT-MEFGE envolvem os campos de tensão e deslocamento no domínio e deslocamento no contorno dos elementos finitos e assim, existirão bases de aproximação e possíveis funções de enriquecimento independentes para os campos de tensão e deslocamentos envolvidos nessas formulações.

Agora, pensando numa solução aproximada $\tilde{u}(x)$, para o campo de deslocamentos $u(x)$ do problema de barra sob força normal, modelado nas Eqs. (11), (12) e (13), construída com base no MEF, MEFG e MEFGE, tem-se a obrigatoriedade de desenvolver a forma integral enfraquecida desse PVC.

Assim, utilizando o método de Resíduos Ponderados – método de Galerkin - pode-se escrever a seguinte expressão integral:

$$\int_0^L E \cdot Au'(x)w(x)dx = \int_0^L qw(x)dx + Pw(L) \tag{14}$$

Na Eq. (14) $w(x)$ é a função definida no espaço W das funções de ponderação ou peso. Define-se formalmente W como:

$$W = \left\{ w(x) \middle| w(0) = 0, \int_0^L \left(w'(x)\right)^2 dx < \infty \right\} \tag{15}$$

A Eq. (15) garante que a $w(x)$ deve ser homogênea nas condições de contorno essenciais do PVC abordado (no problema em questão, $w(0)=0$). Ressalta-se ainda que a condição de contorno essencial, representada na Eq. (12), não foi imposta na Eq. (14) e deve ser garantida quando da aplicação do MEF, MEFG e MEFGE na aproximação do PVC.

Outra apresentação para o problema expresso na Eq. (14) pode ser escrita da seguinte forma:

Determinar $u(x) \in U$ tal

$$B(u(x), w(x)) = F(w(x)) \;\forall\; w(x) \in W \tag{16}$$

onde $B(u(x), w(x)) = \int_0^L E \cdot Au'(x) w(x) dx$ é a forma bilinear (produto interno energético), U é o espaço energia das funções cinematicamente admissíveis e com norma energia definida por $\|u(x)\|_U = \sqrt{\frac{1}{2} B(u(x), u(x))}$ e $F(w(x)) = \int_0^L qw(x)dx + Pw(L)$ é uma forma linear, Bathe (1996), Schwab (1998). Define-se, ainda, a energia de deformação como sendo o valor obtido de $\frac{1}{2} B(u(x), u(x))$.

Por outro lado, considerando-se que a abordagem adotada para gerar uma aproximação (via MEF, MEFG ou MEFGE, por exemplo) para a forma bilinear apresentada na Eq.(16) seja do tipo Galerkin (observe que Galerkin foi aplicado para obtenção da Eq.(14)), então o espaço $U = W$. Além disso, o emprego do MEF, MEFG e MEFGE para a estruturação das aproximações acaba por atribuir uma dimensão finita $\tilde{U}$ ao espaço U, com $\tilde{U} \subset U$.

Dessa forma, definem-se as seguintes aproximações, respectivamente, para $u(x)$ e $w(x)$:

$$\tilde{u}(x) = \phi_i(x)\alpha_i \text{ com } i = 1,...,n \tag{17}$$

$$\tilde{w}(x) = \phi_j(x)\beta_j \text{ com } j = 1,...,n \tag{18}$$

onde α_i e β_j são coeficientes constantes. Lembra-se aqui que os índices repetidos indicam somatório.

Nas Eq.(17) e (18) $\phi_i(x)$ e $\phi_j(x)$ são, respectivamente, as n funções de aproximação do deslocamento $u(x)$ e da ponderação $w(x)$; α_i e β_j são os pesos destas aproximações (coeficientes constantes). Vale lembrar que a aproximação $\tilde{u}(x)$ deve ser cinematicamente admissível (atender, obrigatoriamente, a condição de contorno essencial representada na Eq.(12)) ao espaço de dimensão finita que pertence ao espaço U, definido explicitamente na Eq. (19).

$$U = \left\{ u(x) \middle| u(0) = 0, \int_0^L (u'(x))^2 dx < \infty \right\} \tag{19}$$

Ainda da Eq. (19), ressalta-se que a aproximação $\tilde{u}(x)$ tem valor finito quando o operador integral for aplicado a sua derivada ao quadrado.

Substituindo as aproximações definidas nas Eq. (17) e (18) na Eq. (16) já particularizada para dimensão finita, após alguns desenvolvimentos, chega-se na expressão indicial para o sistema que permite determinar os coeficientes α_i:

$$\alpha_i K_{ij} = F_j \text{ com } i, j = 1, ..., n \tag{20}$$

onde:

$$K_{ij} = \int_0^L E \cdot A \phi_i' \phi_j' dx \text{ com } i, j = 1, ..., n \tag{21}$$

e

$$F_j = \int_0^L q\phi_j dx + P\phi_j(L) \text{ com } i, j = 1, ..., n \tag{22}$$

Finalmente, as n equações dadas na Eq.(20) podem ser representadas matricialmente reunindo-se os coeficientes α_i num vetor coluna **d** (dimensão $n \times 1$), os F_i num outro vetor coluna **f** (dimensão $n \times 1$) e os K_{ij} nas posições de linha i e coluna j de uma matriz **K** (dimensão $n \times n$), assim:

$$\mathbf{K^T\, d = f} \tag{23}$$

Observando a Eq.(20), tem-se que **K** é simétrica, já que $K_{ij} = K_{ji}$, então:

$$\mathbf{K d = f} \tag{24}$$

O MEF apresenta, justamente, uma metodologia para estruturar uma aproximação do tipo $\tilde{u}(x) = \alpha_i \phi_i(x)$, com $i = 1,...,n$, e percorre, basicamente, os passos que seguem:

- Distribuir uma quantidade n de pontos nodais (nós) no domínio do problema - unidimensional - $0 \le x \le L$.
- Esses n nós vão definir regiões discretas no domínio, $0 \le x \le L$, denominadas de elementos finitos.
- Para cada nó i definido nesse domínio, associa-se uma função $\phi_i(x)$, com $i = 1,...,n$, que vai compor a base de dimensão finita $\tilde{U} \equiv \tilde{W}$. Estas funções $\phi_i(x)$, com $i = 1,...,n$, por exemplo, podem ser polinômios (que devem ser contínuos no domínio em questão). Elas ainda devem ter regularidade compatível com a exigida na Eq. (16) e atender as condições impostas nas Eq. (15) e (19) (as $\phi_i(x)$, com $i = 1,...,n$, vão construir uma $\tilde{u}(x)$ que atende a Eq. (19), devem ser homogêneas nas condições de contorno essenciais - para a definição de $\tilde{w}(x)$ - e possuir derivada primeira quadrado integrável no domínio do problema estudado ($0 \le x \le L$)).

- Estabelecer o grau p da função polinomial $\phi_i(x)$, associada ao nó i, para estruturar o tipo de elemento finito que será utilizado na discretização do domínio do problema. A quantidade de nós de um dado elemento finito está atrelada ao grau p do polinômio $\phi_i(x)$. Para o exemplo estudado será adotado $p=1$ (aproximação linear) e o elemento finito da discretização possuirá dois nós. Então, para as funções $\phi_i(x)$ lineares, $p=1$, os n nós no domínio unidimensional vão definir uma discretização com $n-1$ elementos finitos, ver figura 04. Vale ressaltar que este grau de aproximação satisfaz as exigências definidas na Eq. (16).
- As funções $\phi_i(x)$ devem ter suporte compacto, ou seja, são diferentes de zero somente numa dada parte do domínio. Além disso, quando a função $\phi_i(x)$ for avaliada na coordenada do nó i ela, obrigatoriamente, deve valer a unidade (1), ver figura 04.

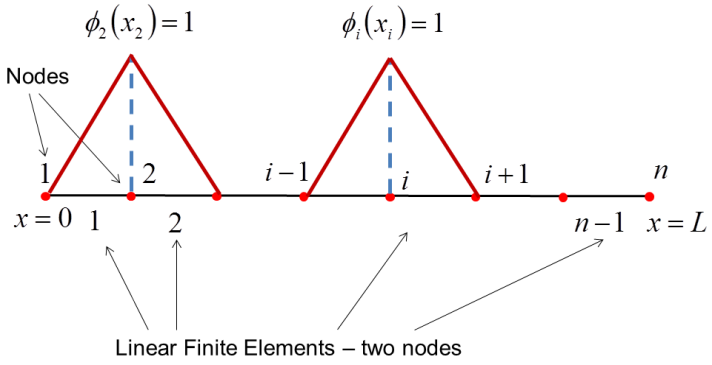

Figura 04 – Discretização do domínio do PVC estudado - $0 \le x \le L$ - com elementos finitos lineares (dois nós).

- Com as características das $\phi_i(x)$, apontadas no item anterior, pode-se escrever a aproximação $\tilde{u}(x)=\alpha_i\phi_i(x)$ da seguinte forma: $\tilde{u}(x)=\tilde{u}(x_i)\phi_i(x)$. Agora, os

parâmetros incógnitos do problema tem correspondência com os valores nodais da função $\tilde{u}(x)$ nas coordenadas dos nós i utilizados na discretização - $\alpha_i = \tilde{u}(x_i)$.

- Após discretizar o domínio unidimensional em elementos finitos, avaliar as Eqs. (21) e (22) nos subdomínios, definidos por cada elemento, para a composição da **K** e do vetor **f** .
- Impor as condições de contorno essenciais do problema no sistema da Eq. (24) de forma $\tilde{u}(x) = \tilde{u}(x_i)\phi_i(x)$ atender a Eq. (19).
- Resolver o sistema da Eq. (24), com as condições de contorno essenciais impostas.
- Finalmente, escrever a $\tilde{u}(x) = \tilde{u}(x_i)\phi_i(x)$.

Como exemplo de estruturação das funções $\phi_i(x)$, via MEF, para determinação de $\tilde{u}(x)$, considere o domínio do problema da figura 03, $0 \le x \le L$, discretizado com três nós $(i = 3)$, dois elementos finitos e coordenadas ilustradas na figura 05.

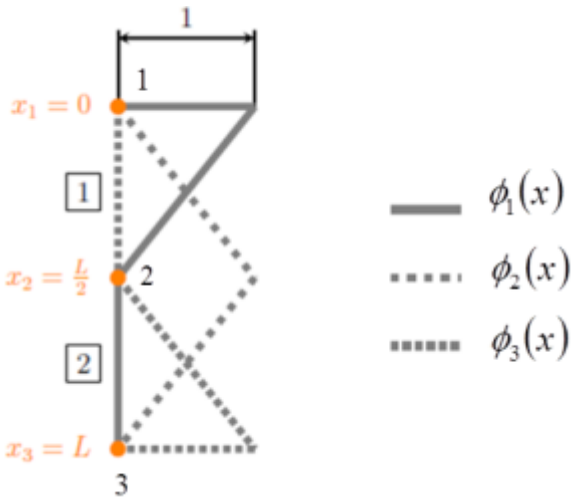

Figura 05 – Discretização do domínio do PVC modelo - $0 \le x \le L$ - com dois elementos finitos lineares (dois nós).

Na discretização da figura 05 foram associadas aos nós 1 com $(x_1 = 0)$, 2 com $\left(x_2 = \frac{L}{2}\right)$ e 3 com $(x_3 = L)$, respectivamente, as funções lineares $\phi_1(x), \phi_2(x)$ e $\phi_3(x)$.

As funções $\phi_1(x), \phi_2(x)$ e $\phi_3(x)$, nos subdomínios dos dois elementos finitos, foram determinadas a partir da figura 05. Assim:

$$\phi_1(x) = \begin{cases} \dfrac{-2x}{L} + 1, \ 0 \le x \le \dfrac{L}{2} \\ \\ 0, \dfrac{L}{2} \le x \le L \end{cases} \tag{25}$$

$$\phi_2(x) = \begin{cases} \dfrac{2x}{L}, \ 0 \le x \le \dfrac{L}{2} \\ \\ \dfrac{-2x}{L} + 2, \dfrac{L}{2} \le x \le L \end{cases} \tag{26}$$

$$\phi_3(x) = \begin{cases} 0, \ 0 \le x \le \dfrac{L}{2} \\ \\ \dfrac{2x}{L} - 1, \dfrac{L}{2} \le x \le L \end{cases} \tag{27}$$

Assim, para a discretização da figura 05 e as funções definidas nas Eqs.(25), (26) e (27) , cada termo da matriz **K** e do vetor **f** serão obtidos; os termos do vetor **d** determinados e, dessa forma, a aproximação $\tilde{u}(x)$ assume a seguinte estrutura:

$$\tilde{u}(x) = \tilde{u}(x_1)\phi_1(x) + \tilde{u}(x_2)\phi_2(x) + \tilde{u}(x_3)\phi_3(x) \tag{28}$$

A questão que se segue é apresentar a metodologia para definição da aproximação do tipo $\tilde{u}(x)=\alpha_i\phi_i(x)$, com $i=1,...,n$, associadas agora ao MEFG e MEFGE. Para a aplicação do MEFG e MEFGE no PVC em questão, definem-se suportes ou nuvens formadas pelos elementos finitos lineares de dois nós. Cada nuvem ω_i, ver figura 06, é formada pelos elementos finitos que compartilham um nó i em comum.

Agora, a fim de proporcionar o enriquecimento da aproximação $\phi_i(x)$ atrelada aos nós i da malha de elementos finitos, adotam-se, neste trabalho, funções polinomiais $h_i(x)$ em concordância com a metodologia de enriquecimento nodal do MEFG ou do MEFGE.

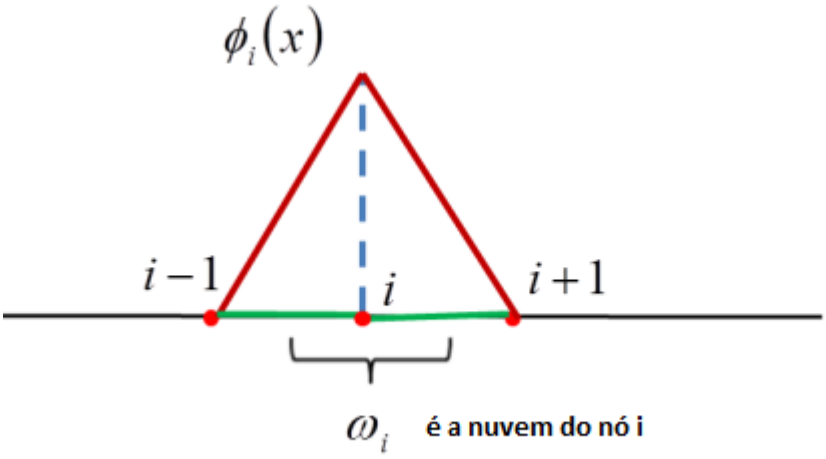

Figura 06 – Exemplo de uma nuvem para uma discretização unidimensional (elemento linear – dois nós).

Para o PVC em questão, as funções enriquecedoras $h_i(x)$ aplicadas ao MEFG são:

- x

$$h_i(x)=(x-x_i) \tag{29}$$

- x^2

$$h_i(x)=(x-x_i)^2 \tag{30}$$

Já para o MEFGE as funções enriquecedoras $h_i(x)$ são:

- x

$$h_i(x) = (x - x_i) - \sum_{j=1}^{2} \phi_j (x_j - x_i) \tag{31}$$

- x^2

$$h_i(x) = (x - x_i)^2 - \sum_{j=1}^{2} \phi_j (x_j - x_i)^2 \tag{32}$$

O interpolante apresentado na figura 02 é justamente o somatório presente nas Eq. (31) e Eq. (32). A mudança proposta nas funções enriquecedoras do MEFGE é justamente no sentido de melhorar o condicionamento da matriz de coeficientes **K** do sistema linear representado na Eq.(24) quando da aplicação do processo de enriquecimento nodal, Babuška e Banerjee (2012), Gupta et al. (2013) e Lins (2015). Ressalta-se também que todas as funções enriquecedoras $h_i(x)$ definidas estão no formato bolha.

O enriquecimento das $\phi_i(x)$, representada agora por uma $\phi_i^*(x)$, são definidos da seguinte forma:

$$\phi_i^*(x) = \phi_i(x) h_i(x) \tag{33}$$

Por fim, a escolha das funções enriquecedoras definidas nas Eq. (29) a (32) e as modificações do sistema de equações da Eq. (24), quando da aplicação do enriquecimento nodal do MEFG ou MEFGE, serão justificadas no item de Resultados Numéricos.

4 FHMT, FHMT-MEFG E FHMT- MEFGE APLICADOS AO PROBLEMA DE BARRA SOB FORÇA NORMAL

As derivações da FHMT, FHMT-MEFG e FHMT-MEFGE baseiam-se nos passos anteriores do MEF/MEFG/MEFGE, ou seja, determina-se a forma integral enfraquecida do PVC em questão para aplicação posterior da técnica de elementos finitos e enriquecimento nodal segundo o MEFG e MEFGE.

O principal objetivo desta seção é obter três campos de aproximação independentes: tensão e deslocamento no domínio e deslocamento no contorno. A equação de compatibilidade, Eq. (2) pode ser escrita como (34) em que $\varepsilon(x)$ representa o tensor de deformação unidimensional. Sob o estado plano de tensão a Lei de Hooke, Eq. (6) pode ser escrita como Eq. (35), com a relação inversa do Módulo de Young *E*. Substituindo a Eq. (34) em (35), obtêm-se uma nova formulação.

$$\varepsilon(x) - u'(x) = 0 \tag{34}$$

$$\varepsilon(x) = f\sigma(x) \text{ com } f = \frac{1}{E} \tag{35}$$

$$f\sigma(x) - u'(x) = 0 \tag{36}$$

A equação de equilíbrio, Eq. (1) pode ser escrita como a Eq. (37) em que $b_x = \frac{q}{A}$ é definido como o vetor de forças volúmicas na direção x. Geralmente o tensor de tensão de Cauchy é representado na forma $\bar{t} = n\sigma$ e relaciona o vetor unitário *n* com o vetor de tensão σ através de uma superfície imaginária perpendicular a *n.* Neste caso a tensão na fronteira $\sigma(L)$ no ponto $x = L$ é a tração e é representada pelo tensor de tensão de Cauchy $\bar{t}(L) = \sigma(L) = \frac{P}{A}$, Eq. (38), n na direção de x é considerada 1 porque ocorre na direção de *P*.

$$\frac{d\sigma(x)}{dx} + b_x = 0 \tag{37}$$

$$\sigma(L) - \bar{t}(L) = 0 \tag{38}$$

As expressões acima (36), (37) e (38) contêm a tensão $\sigma(x)$, a deformação $\varepsilon(x)$ e o deslocamento $u(x)$ em uma dimensão e são utilizadas para formar as relações de ponderação de Galerkin, Eq. (39), (40) e (41). A integração do domínio *x* é representada pelos limites do domínio *x = 0* até *x = L* e Γ_t $(x = L)$ verifica a condição estática de fronteira.

$$\int_0^L \delta u(x)\left[\frac{d\sigma(x)}{dx} + \frac{q}{A}\right]dx = 0 \tag{39}$$

$$\delta u_\Gamma(L)\left[\sigma(L) - \frac{P}{A}\right] = 0 \tag{40}$$

$$\int_0^L \delta\sigma(x)[f\sigma(x) - u'(x)]dx = 0 \tag{41}$$

O foco é na integração da Eq. (41). Portanto, após efetuar a integral por partes da integral $\int_0^L \delta\sigma(x)u'(x)dx = \delta\sigma(x)u(x)\big|_0^L - \int_0^L \frac{d\delta\sigma(x)}{dx}u(x)dx$ em (41) e aplicar a condição de fronteira essencial, segue para (42) em que $u(x)$ é o deslocamento prescrito no domínio e $u_\Gamma(L)$ é assumido como um campo de deslocamento independente na fronteira estática Γ_t $(x = L)$. Devido a essas características particulares, ambos os campos, domínio e fronteira, devem ser levados em conta ao usar o método dos elementos finitos com malhas. Além disso, (39) e (40) são mostradas de uma outra maneira como é ilustrada em (43) e (44) respectivamente.

$$\int_0^L \delta\sigma(x)f\sigma(x)dx + \int_0^L u(x)\frac{d\delta\sigma(x)}{dx}dx - \delta\sigma(L)u_\Gamma(L) = 0 \tag{42}$$

$$\int_0^L \delta u(x)\left[\frac{d\sigma(x)}{dx}\right]dx = -\int_0^L \delta u(x)\frac{q}{A}dx \tag{43}$$

$$-\delta u_\Gamma(L)\sigma(L) = -\frac{P}{A}\delta u_\Gamma(L) \tag{44}$$

Sobre as aproximações numéricas definidas no domínio *x* e na fronteira $\Gamma_t\ (x = L)$ da barra, três campos aproximados e independentes podem ser indicados através de interpolações dos valores nodais como foi definido, por exemplo, na Eq. (17). Agora, os termos de aproximação para a tensão no domínio $\tilde{\sigma}(x)$, deslocamento no domínio $\tilde{u}(x)$ e deslocamento na fronteira $\tilde{u}_\Gamma(x)$ podem ser vistos a seguir.

$$\tilde{\sigma}(x) = \phi_j(x)\sigma_j(x_j) \text{ com } j = 1,...,n \tag{45}$$

$$\delta\tilde{\sigma}(x) = \phi_i(x)\delta\sigma_i(x_i) \text{ com } i = 1,...,n \tag{46}$$

$$\tilde{u}(x) = \phi_j(x)u_j(x_j) \text{ com } j = 1,...,n \tag{47}$$

$$\delta\tilde{u}(x) = \phi_i(x)\delta u_i(x_i) \text{ com } i = 1,...,n \tag{48}$$

$$\tilde{u}_\Gamma(x) = \phi_j(x)u_{\Gamma j}(x_j) \text{ com } j = 1,...,n \tag{49}$$

$$\delta\tilde{u}_\Gamma(x) = \phi_i(x)\delta u_{\Gamma i}(x_i) \text{ com } i = 1,...,n \tag{50}$$

$\sigma_j(x_j)$ representa o vetor das variáveis de tensão nodal enquanto que $u_j(x_j)$ e $u_{\Gamma j}(x_j)$ refletem os vetores nodais dos graus de liberdade considerados para o deslocamento de todos os *n* nós definidos em uma discretização específica do problema analisado.

As integrais (42), (43) e (44) podem ser escritas como:

$$\left(\int_0^L \phi_i(x) f\phi_j(x)dx\right)\sigma_j(x_j) + \left(\int_0^L \phi_i'(x)\phi_j(x)dx\right)u_j(x_j) + \left(-\phi_i(L)\phi_j(L)\right)u_{\Gamma j}(x_j) = 0 \tag{51}$$

$$\left(\int_0^L \phi_j'(x)\phi_i(x)dx\right)u_j(x_j) = -\int_0^L \phi_i(x)\frac{q}{A}dx \tag{52}$$

$$\left(-\phi_i(L)\phi_j(L)\right)u_j\left(x_j\right) = -\frac{P}{A}\phi_i(L) \tag{53}$$

Considerando todos os três campos de aproximação é possível escrever as relações acima como um sistema linear de equações.

$$\begin{bmatrix} F & A_\Omega & -A_{\Gamma t} \\ A_\Omega^T & 0 & 0 \\ -A_{\Gamma t}^T & 0 & 0 \end{bmatrix} \begin{Bmatrix} \sigma_j\left(x_j\right) \\ u_j\left(x_j\right) \\ u_{\Gamma j}\left(x_j\right) \end{Bmatrix} = \begin{Bmatrix} 0 \\ -Q_\Omega \\ -Q_{\Gamma t} \end{Bmatrix} \tag{54}$$

Os vetores e os elementos das submatrizes do sistema linear de equações são mostrados abaixo:

$$F = F_{ij} = \int_0^L \phi_j \, \boldsymbol{f} \phi_i dx \tag{55}$$

$$A_\Omega = A_{ij\Omega} = \int_0^L \phi_j \, \boldsymbol{f} \phi'_i dx \tag{56}$$

$$A_{\Gamma t} = A_{ij\Gamma} = -\phi_j(L)\phi_i(L) \tag{57}$$

$$Q_\Omega = Q_{\Omega i} = -\int_0^L \phi_j \, \boldsymbol{b} x \tag{58}$$

$$Q_{\Gamma t} = Q_{\Gamma t i} = -\phi_j(L)\bar{t}_x(L) \tag{59}$$

$$e_{\Gamma u} = \int_{\Gamma u} S_\Omega^T \bar{u} d\Gamma = 0 \tag{60}$$

4.1 SOLUÇÃO DO SISTEMA DE EQUAÇÕES DA FHMT COM O MÉTODO DE BABUŜKA

Nesta seção será apresentado um método numérico capaz de resolver o Sistema Linear de Equações em (54) primeiramente introduzido por Strouboulis, Copps e Babuška (2000) e posteriormente processado por Góis (2009). Checando o sistema em (61) e (62) (igual para (54)) obtêm-se diferentes autovalores após a aplicação das condições de contorno. Valores negativos estão atrelados ao deslocamento, já valores positivos estão associados à tensão e os autovalores nulos são associados aos modos espúrios do sistema. O método de Babuška irá alterar o sistema a fim de que a matriz A (matriz que contêm as submatrizes do sistema e esparsa) deixe de ser singular (determinante da matriz é igual a 0, logo essa matriz não possui uma inversa). Para resolver o sistema imposto pelo método de Babuška foi utilizado a decomposição de Cholesky.

O sistema linear de equações em (54) pode ser escrito como elementos da matriz A e do vetor b como é visto em (61).

$$\boldsymbol{Ax = b} \tag{61}$$

Em que

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{A}_\Omega & -\boldsymbol{A}_\Gamma \\ \boldsymbol{A}_\Omega^T & 0 & 0 \\ -\boldsymbol{A}_\Gamma^T & 0 & 0 \end{bmatrix}; \boldsymbol{x} = \begin{pmatrix} \boldsymbol{s}_\Omega \\ \boldsymbol{q}_\Omega \\ \boldsymbol{q}_\Gamma \end{pmatrix} \text{ and } \boldsymbol{b} = \begin{pmatrix} \boldsymbol{e}_\Gamma \\ -\boldsymbol{Q}_\Omega \\ -\boldsymbol{Q}_\Gamma \end{pmatrix} \tag{62}$$

Com o auxílio da matriz diagonal $\boldsymbol{B}$ em (63) o sistema pode escrito como em (64)

$$\boldsymbol{B} = \frac{1}{\sqrt{diag(F)}} \tag{63}$$

$$\bar{A}\bar{x} = \bar{b} \tag{64}$$

O novo sistema com a introdução da matriz diagonal ***B*** pode ser escrito como

$$\bar{A} = \begin{bmatrix} BFB & BA_\Omega & -BA_\Gamma \\ BA_\Omega & 0 & 0 \\ -BA_\Gamma & 0 & 0 \end{bmatrix};\ \bar{x} = \begin{pmatrix} s_\Omega \\ \boldsymbol{q}_\Omega \\ \boldsymbol{q}_\Gamma \end{pmatrix} e\ \bar{b} = \begin{pmatrix} e_\Gamma \\ -Q_\Omega \\ -Q_\Gamma \end{pmatrix} \tag{65}$$

Nota-se que

$$diag(\boldsymbol{BFB}) = I \tag{66}$$

e que

$$s_\Omega = \boldsymbol{B}\overline{s_\Omega} \tag{67}$$

A Eq. (67) é usada apenas quando a solução do sistema é obtida. Pelo método, obtêm-se a seguinte aproximação

$$\tilde{A} = \bar{A} + \varepsilon I \tag{68}$$

ε é um número pequeno e positivo. A Eq. (68) garante que $\tilde{\mathbf{A}}$ seja uma matriz positiva definida. O seguinte algoritmo foi desenvolvido para explicar método de Babuŝka de forma cronológica em que a_j é o erro da iteração j e $\bar{r}_j$ é o resíduo da iteração j.

1. $j = 0;\ \overline{x_o} = 0;\ \overline{r_o} = \bar{b}$

2. $Resolver\ \tilde{A}a_j = \overline{r_j};\ usar\ a\ Decomposição\ de\ Cholesky$

3. $\bar{x}_{j+1} = \bar{x}_j + \bar{a}_j$

4. $\bar{r}_{j+1} = \bar{r}_j - \bar{A}_j\, a_j$

5. $j = j + 1$

$$6. Se \left(\frac{\bar{r}_j^T \bar{r}_j}{x_j^T \bar{A} \bar{x}_j} > tol \right) então\ volte\ para\ o\ item\ 2$$

Esse algoritmo pode ser usado para a FHMT-MEFG e FHMT-MEFGE apenas modificando as funções de enriquecimento, assim como acontece no MEFG e MEFGE a partir do MEF clássico.

5 IMPLEMENTAÇÃO COMPUTACIONAL

A implementação computacional utilizada seguiu modelos exemplo de Interface Gráfica do MATLAB juntamente com os códigos desenvolvidos e automatizados do MEF e da FHMT. Na figura 06 é apresentado a versão 1 da interface desenvolvida.

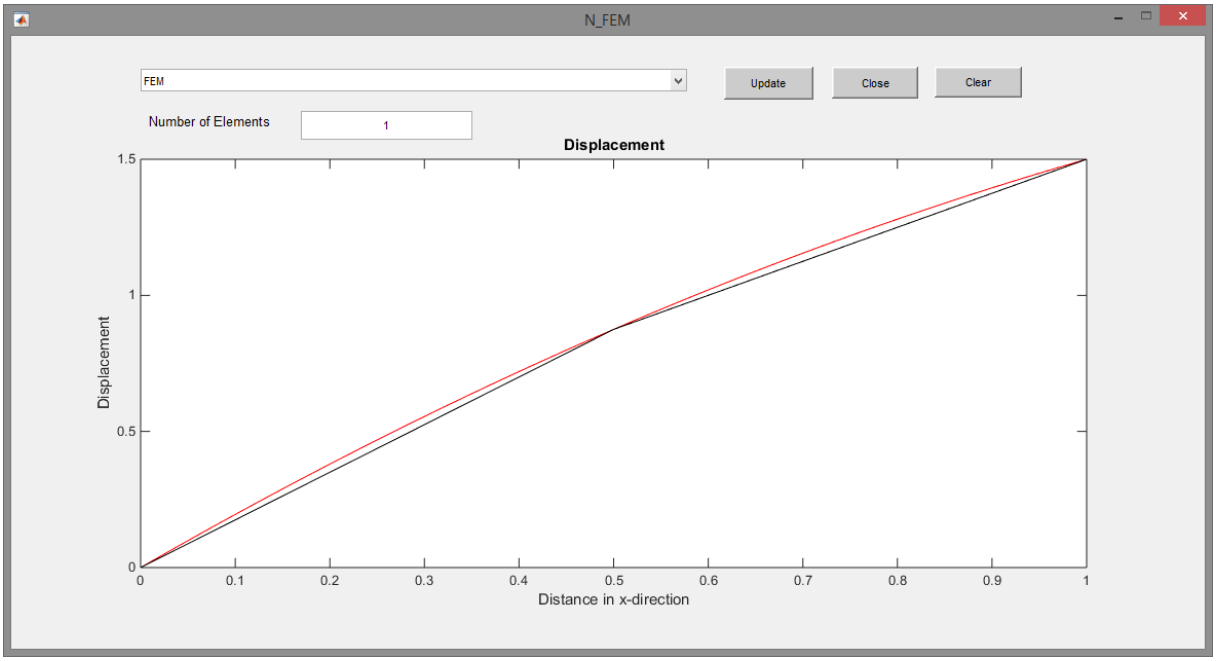

Figura 06 – Interface Gráfica N_FEM desenvolvida no MATLAB®.

Com a interface é possível, por exemplo, verificar a solução aproximada indicando a quantidade de elementos no campo “Number of Elements” e assim através do botão “Update” o gráfico é atualizado. Caso haja necessidade de limpar o gráfico, o botão “Clear” zera a plotagem e por fim para fechar a aplicação, o botão “Close” é utilizado. No Apêndice tem o código para a interface.

6 RESULTADOS NUMÉRICOS

Esta seção apresentará os resultados obtidos para o problema unidimensional da barra sob força normal com dois elementos, três nós, engastada em uma das extremidades *u(0) = 0;* para o MEF, MEFG (enriquecimento com a Eq. (29) - por simplificação, denominada nas figuras como x) , MEFGE (enriquecimento com a Eq. (32) - por simplificação, denominada nas figuras como x^2), FHMT, FHMT-MEFG (enriquecimento com a Eq. (29) – x) e FHMT-MEFGE (enriquecimento com a Eq. (32)). Para facilitar a representação e análise da solução os seguintes parâmetros foram normalizados: P = 1, E = 1, Area = 1, q = 1, L = 1.

6.1 ANÁLISE NUMÉRICA PARA O MEF CLÁSSICO

Na figura 07 é apresentada a solução exata do problema, assim como as soluções aproximadas do MEF, MEFG e MEFGE com relação ao campo de deslocamento do domínio em x.

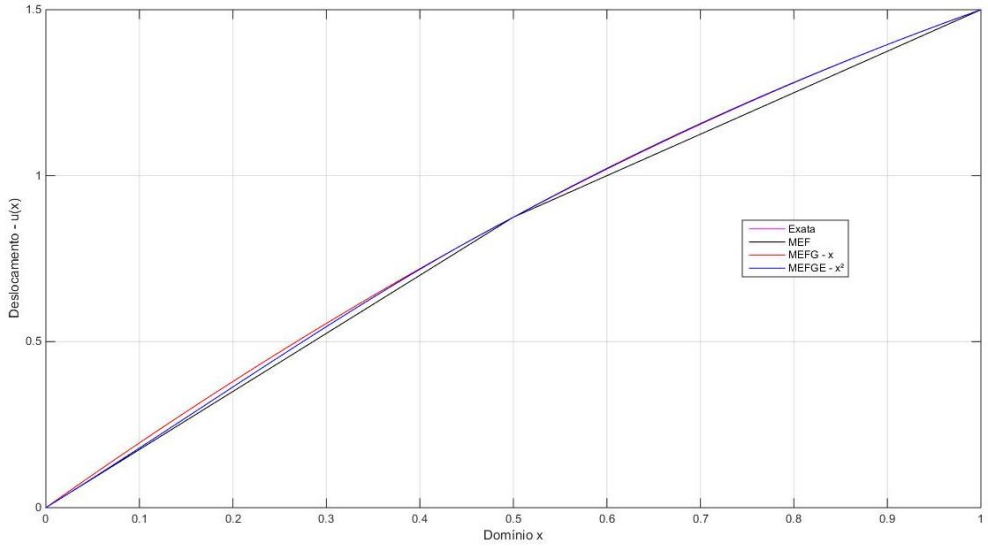

Figura 07 – Comparação entre a solução exata, MEF, MEFG e MEFGE no deslocamento.

Nota-se que com um enriquecimento em x do MEFG a solução exata é recuperada, já a solução do enriquecimento em x^2 do MEFGE aproxima-se da solução exata no primeiro intervalo e apenas no segundo intervalo a solução é recuperada.

Apesar de apresentar uma solução aproximada inferior ao do MEFG, deve-se observar que a matriz A do sistema generalizado apresenta um mau condicionamento, ao contrário do que acontece num sistema generalizado estável. Isso acontece porque é adicionado à função enriquecedora um interpolante que tem a função de linearizar o sistema e torná-lo mais estável. Não é objetivo deste trabalho a análise das questões de condicionamento e estabilidade das soluções numéricas apresentadas aqui.

7 CONCLUSÃO

Neste projeto de pesquisa uma nova formulação numérica foi desenvolvida a fim de que a análise unidimensional do problema de barra sob força normal pudesse apresentar uma solução aproximada próxima da solução de referência quando da utilização da combinação de métodos não-convencionais. O principal objetivo desta proposta foi o estudo do Método dos Elementos Finitos Generalizados Estável (MEFGE) na Formulação Híbrido-Mista de Tensão (FHMT) em problemas 1-D, particularmente, um problema clássico da mecânica estrutural com o intuito de entender toda a teoria do Método dos Elementos Finitos, tanto o método clássico quanto os métodos não-convencionais aqui abordados para suprir toda a deficiência teórica quando aplicados a problemas bi ou tridimensionais.

Como a formulação não-convencional (FHMT) apresenta uma forma direta da aproximação do campo de tensão, verificou-se que este método apenas apresenta solução ótima nos campos de tensão, ao contrário do que acontece em deslocamento, como foi possível observar nas análises realizadas sem o enriquecimento. Por meio das análises com o enriquecimento (na estrutura do MEFGE) como uma nova formulação não-convencional (FHMT-MEFGE), verificou-se que a solução aproximada obteve um bom resultado tanto no campo de tensão, quanto no campo de deslocamento quando comparada com a solução de exata do problema proposto. Além disso, este novo método (FHMT-MEFGE) apresentou resultados totalmente satisfatórios quando comparado aos outros métodos, tal como o MEF com relação à solução aproximada – deslocamento e tensão (visto na figura 08 e 09.

Desta forma, pode-se concluir que a FHMT-MEFGE conseguiu de forma inovadora aproximar tanto o campo de deslocamento, quanto o campo de tensão para um problema unidimensional, particularmente o problema de barra sob força normal.

Sendo assim, este método poderá ser implementado para outros problemas da mecânica estrutural com o intuito de aprimorar a técnica de enriquecimento a partir do sistema MEFGE em uma formulação não-convencional, aqui escolhida a FHMT. Pensando-se em trabalhos futuros, destaca-se a possibilidade do emprego desta nova formulação (FHMT-MEFGE) para análise de problemas bi e tridimensionais e o fortalecimento do entendimento de questões relacionadas com estabilidade e convergência de solução numérica via formulações ditas não-convencionais.

8 REFERÊNCIAS BIBLIOGRÁFICAS

Babuška, I., 1971. Error bounds for finite element methods. ***Numerische Mathematik,*** **v.16, p. 322-333.**

Babuška, I., 1973. The finite element method with lagrange multipliers. ***Numerische Mathematik,*** **v.20, p. 179-192.**

Babuška, I.,1996. ***On the inf-sup (babuška-brezzi) condition.*** **The University of Texas at Austin. Technical Report #5. TICAM.**

Babuška, I., Caloz, G., Osborn, J. E., 1994. Special finite element method for a classe of second order elliptic problems with rough coefficients. ***SIAM Journal on Numerical Analysis,*** **v.31, n.4, p. 727-981.**

Babuška, I., Melenk, J. M., 1997. The partition of unity method. ***International Journal for Numerical Methods in Engineering,*** **v. 40, p. 727-758.**

Bathe, K. J., 1996. ***Finite element procedures.*** **2.ed. New Jersey: Prentice-Hall.**

Bathe, K. J., Hendriana, D., Brezzi, F., Sangalli, G., 2000. Inf-sup test of upwind methods. ***International Journal for Numerical Methods in Engineering,*** **v.48, p.745-760.**

Bathe, K. J., Iosilevich, A., Chapelle, D., 2000. An inf-sup test for shell finite elements. ***Computer and Structures,*** **v.75, p. 439-456.**

Brezzi, F., 1974. On the existence, uniqueness and approximation of saddle point problems arising from lagrange multipliers. RAIRO, v.8 (r-2), p. 127-151.

Brezzi, F., Bathe, K. J., 1990. A discourse on the stability conditions for mixed finite element formulation. ***Computer Methods in Applied Mechanics and Engineering,*** **v.82, p. 27-57.**

Brezzi, F., Fortin, M., 1991. ***Mixed and hybrid finite element methods*****. New York: Springer Verlag.**

Chapelle, D., Bathe, K. J., 1993. The inf-sup test. ***Computers & Structures*****, v. 47, n.4/5, p. 537-545.**

Duarte, C. A., 1996. ***The hp-cloud method. Tese (Doutorado)*** **- The University of Texas at Austin.**

Duarte, C. A., Babuška, I., Oden, J. T., 2000. Generalized finite element methods for three-dimensional structural mechanics problems. ***Computers & Structures*****, v. 77, n. 2, p. 215-232.**

Duarte, C. A., Oden, J. T., 1995. ***Hp clouds – a meshless to solve boundary - value problem*****. The University of Texas at Austin. Technical Report. TICAM.**

Duarte, C. A., Oden, J.T., 1996. Hp clouds – an hp meshless method. ***Numerical Methods for Partial Differential Equations*****. John Wiley & Sons, p. 1 - 34.**

Góis. W., 2009. Stress Hybrid and Hybrid-Mixed Finite Elements with Nodal Enrichment,

PhD Thesis (in Portuguese), São Carlos School of Engineering, University of São Paulo.

Góis, W., Proença, S.P.B., 2012. Generalized finite element method on nonconventional hybrid-mixed formulation. ***International Journal of Computational Methods*****, v.9, n.3, p. 1250038-1 -1250038-24.**

Melenk, J. M., Babuška, I., 1996. The partition of unity finite element method: basic theory and applications. ***Computer Methods in Applied Mechanics and Engineering*****, v.139, p.289-314.**

Oden, J. T., Duarte, C.A., Zienkiewicz, O. C., 1998. A new cloud – based hp finite element method. ***Computer Methods in Applied Mechanics and Engineering*****, v. 153, p. 117-126.**

Schwab, CH., 1998. p- and hp- finite element methods: theory and applications in solid and fluid mechanics. Oxford: Oxford University Press Inc.

Strouboulis, T., Babuška, I., Copps, K., 2000. The design and analysis of the generalized finite element method. ***Computer Methods in Applied Mechanics and Engineering*****, v. 181, n. 1-3 , p. 43-69.**

Szabó, B., Babuška, I., 1991. Finite element analysis. John Wiley & Sons.

Zienkiewicz, O. C. et al., 1986. The patch test for mixed formulation. ***International Journal for Numerical Methods in Engineering*****, v.23, p. 1873-1882.**

APÊNDICES

N_FEM.m

```
function varargout = N_FEM(varargin)
% N_FEM MATLAB code for N_FEM.fig
%      N_FEM, by itself, creates a new N_FEM or raises the existing
%      singleton*.
%
%      H = N_FEM returns the handle to a new N_FEM or the handle to
%      the existing singleton*.
%
%      N_FEM('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in N_FEM.M with the given input arguments.
%
%      N_FEM('Property','Value',...) creates a new N_FEM or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before N_FEM_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to N_FEM_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help N_FEM

% Last Modified by GUIDE v2.5 05-Sep-2019 21:50:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @N_FEM_OpeningFcn, ...
                   'gui_OutputFcn',  @N_FEM_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before N_FEM is made visible.
function N_FEM_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to N_FEM (see VARARGIN)

% Choose default command line output for N_FEM
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using N_FEM.
if strcmp(get(hObject,'Visible'),'off')
    n = 2; % número de elementos
    P = 1; E = 1; A = 1; q = 1; L = 1; % constantes do problema
    coord = linspace(0,L,n+1); % coordenadas dos nós
    Le = coord(2)-coord(1); % tamanho do elemento

    f(1:n+1) = q*Le;
    f(1) = q*Le/2;
    f(n+1) = q*Le/2; % força distribuida
    f(n+1) = f(n+1) + P; % força concentrada no último nó
    f = transpose(f);

    % matriz de rigidez
    diagon(1:n+1) = 2;
    inf_diagon(1:n) = -1;
    sup_diagon(1:n) = -1;
    K = diag(diagon) + diag(inf_diagon,-1) + diag(sup_diagon,1);
    K(1,1) = 1; K(n+1,n+1) = 1; K = E*A/Le*K;

    % condições de contorno
    K(1,:) = 0; K(:,1) = 0; K(1,1) = 1; f(1) = 0;

    u = K\f; % solução direta

    x = linspace(0,L); % solução exata
    u_exato = -q/(2*E*A)*x.^2 + (P+q*L)/(E*A)*x;
    plot(x,u_exato,'r'); title('Displacement'); xlabel('Distance in x-
direction'); ylabel('Displacement'); hold on

    eta = linspace(-1,1); % funções de aproximação parametrizadas
    fi_1 = (1-eta)/2;
    fi_2 = (1+eta)/2;

    for i = 1:n % plot solução aproximada
        u_aprox = fi_1*u(i)+fi_2*u(i+1); % deslocamento
        x_aprox = fi_1*coord(i) + fi_2*coord(i+1);
        plot(x_aprox,u_aprox,'k'); hold on
    end
end

% UIWAIT makes N_FEM wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = N_FEM_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1);
a = get(handles.input_editText,'String');
n = str2double(a);
cla;

popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
    %n = 2; % número de elementos
    P = 1; E = 1; A = 1; q = 1; L = 1; % constantes do problema
    coord = linspace(0,L,n+1); % coordenadas dos nós
    Le = coord(2)-coord(1); % tamanho do elemento

    f(1:n+1) = q*Le;
    f(1) = q*Le/2;
    f(n+1) = q*Le/2; % força distribuida
    f(n+1) = f(n+1) + P; % força concentrada no último nó
    f = transpose(f);

    % matriz de rigidez
    diagon(1:n+1) = 2;
    inf_diagon(1:n) = -1;
    sup_diagon(1:n) = -1;
    K = diag(diagon) + diag(inf_diagon,-1) + diag(sup_diagon,1);
    K(1,1) = 1; K(n+1,n+1) = 1; K = E*A/Le*K;

    % condições de contorno
    K(1,:) = 0; K(:,1) = 0; K(1,1) = 1; f(1) = 0;

    u = K\f; % solução direta

    x = linspace(0,L); % solução exata
    u_exato = -q/(2*E*A)*x.^2 + (P+q*L)/(E*A)*x;
    plot(x,u_exato,'r'); title('Displacement'); xlabel('Distance in x-
direction'); ylabel('Displacement'); hold on

    eta = linspace(-1,1); % funções de aproximação parametrizadas
    fi_1 = (1-eta)/2;
    fi_2 = (1+eta)/2;

    for i = 1:n % plot solução aproximada
        u_aprox = fi_1*u(i)+fi_2*u(i+1); % deslocamento
        x_aprox = fi_1*coord(i) + fi_2*coord(i+1);
        plot(x_aprox,u_aprox,'k'); hold on
    end

    case 2
        %n = 2;                                  % number of elements
```

```
        P = 1; E = 1; Area = 1; q = 1; L = 1;    % constants, P = P2, q=P1
        coord = linspace(0,L,n+1);               % node coordinates;
generates a row vector coord of n+1 elements linearly spaced between 0 and
L.
        Le = coord(2)-coord(1);                  % length of a element

        % Defining matrix F
        Fdiagon(1:n+1) = 2/3;
        Finf_diagon(1:n) = 1/6;
        Fsup_diagon(1:n) = 1/6;
        F = diag(Fdiagon) + diag(Finf_diagon,-1) + diag(Fsup_diagon,1);
        F(1,1) = 1/3; F(n+1,n+1) = 1/3;
        F=F*Le/E;

        % Defining matrix A_dom
        A_dom_diagon(1:n+1) = 0;
        A_dom_inf_diagon(1:n) = 1/2;
        A_dom_sup_diagon(1:n) = -1/2;
        A_dom = diag(A_dom_diagon) + diag(A_dom_inf_diagon,-1) +
diag(A_dom_sup_diagon,1);
        A_dom(1,1) = -1/2; A_dom(n+1,n+1) = 1/2;
        A_dom;

        % Defining matrix A_bound
        A_bound = zeros(n+1);   % Consists only of zeros
        A_bound(n+1,n+1) = -1;  % instead of (n+1,n+1)
        A_bound;

        % Assemble global "bilinear" matrix
        dof = 3*(n+1);
        A = zeros(dof);
        A(1:dof/3,1:dof/3) = F;
        A(1:dof/3,dof/3+1:2*dof/3) = A_dom;
        A(1:dof/3,2*dof/3+1:dof) = A_bound;
        A(dof/3+1:2*dof/3,1:dof/3) = transpose(A_dom);
        A(2*dof/3+1:dof,1:dof/3) = transpose(A_bound);
        A;

        % Define "Linear" force vector
        f = zeros(dof,1);
        f(dof) = -P/Area; f(dof/3+1:2*dof/3) = -P*Le/Area;
        f(dof/3+1) = -P*Le/(2*Area); f(dof*2/3) = -P*Le/(2*Area);
        f;

        % Solve linear equation system

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Check of matrix A for the eigenvalues to evaluate the needed
solving system
        A(2*dof/3+1,:) = 0; A(:,2*dof/3+1) = 0; A(2*dof/3+1,2*dof/3+1) = 1;
f(2*dof/3+1) = 0;
        eigenvalue_A = eig(A); % Calculate the eigenvalues before applying
Babuska method

        % Initial steps
        diagF = diag(F);
        diagFF = diag(diagF);
        B = sqrt(inv(diagFF));   % Determination of B (Equation 4.198 in
PhD of Wesley Góis )
```

```
        % Matrix change by multiplying elements of Abar with B (4.199)
        Abar = A;
        Abar(1:dof/3,1:dof/3) = B*A(1:dof/3,1:dof/3)*B;
        Abar(1:dof/3,dof/3+1:2*dof/3) = B*A(1:dof/3,dof/3+1:2*dof/3);
        Abar(1:dof/3,2*dof/3+1:dof) = B*A(1:dof/3,2*dof/3+1:dof);
        Abar(dof/3+1:2*dof/3,1:dof/3) = A(dof/3+1:2*dof/3,1:dof/3)*B;
        Abar(2*dof/3+1:dof,1:dof/3) = A(2*dof/3+1:dof,1:dof/3)*B;

        % ATTENTION - MatLab can´t calculate the eignevalues of matrix Abar
        eigenvalue_Abar = eig(Abar + eye(dof)*10^-8);    % Check
eigenvalues again

        % Usage of the inverse power method to get the lowest eigenvalue
(epsilon)
        [vector,lowest] = myipm(Abar+eye(dof)*10^-8,100);

        % Babuska method steps (4.203)
        Atil = Abar + eye(dof)*10^-8;   %eye=I,(4.202)
        err = 1;
        tol = 10^-5;
        j = 0;
        x = 0;
        r = f;

        %while loop applying cholesky
        while err > tol
            a = cholesky(Atil,r);
            x = x + a;
            r = r - Abar*transpose(a);
            j = j + 1;
            err = real(sqrt((transpose(r)*r)/(x*Abar*transpose(x))))
        end

        tenson = B*transpose(x(1:dof/3));
        displ_boun = transpose(x(2*dof/3+1:dof));
        displ_dom = transpose(x(dof/3+1:2*dof/3));
        displ_dom(1) = displ_boun(1);
        displ_dom(length(displ_dom)) = displ_boun(length(displ_boun));

        % Displacement
        X = linspace(0,L);                                          % Exactly
solution
        u_exato = -q/(2*E*Area)*X.^2 + (P+q*L)/(E*Area)*X;
        plot(X,u_exato,'r');title('Displacement');
ylabel('Displacement');xlabel('Distance in x-direction'); hold on;

        eta = linspace(-1,1);                                       %
Parametrisized shape functions
        fi_1 = (1-eta)/2;
        fi_2 = (1+eta)/2;

        for i = 1:n                                                 % Plot
approximated solution
            u_aprox = fi_1*displ_dom(i)+fi_2*displ_dom(i+1);  %
Displacement
            x_aprox = fi_1*coord(i) + fi_2*coord(i+1);
            tal_aprox = fi_1*tenson(i)+fi_2*tenson(i+1);        % Tension
            plot(x_aprox,u_aprox,'k'); hold on;
        end
```

```
end


% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'FEM', 'FHMT'});



function input_editText_Callback(hObject, eventdata, handles)
% hObject    handle to input_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of input_editText as text
%        str2double(get(hObject,'String')) returns contents of
input_editText as a double

input = str2num(get(hObject,'String'));

%Checks to see if input is empty. If so, default input_editText to one
if(isempty(input))
    set(hObject,'String','1')
end


% --- Executes during object creation, after setting all properties.
function input_editText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to input_editText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton_close.
function pushbutton_close_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)
```

```
% --- Executes on button press in pushbutton_CLEAR.
function pushbutton_CLEAR_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_CLEAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.axes1, 'reset');
guidata(hObject, handles); % update de handles
```

Cholesky.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cholesky decomposition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% https://de.wikipedia.org/wiki/Cholesky-Zerlegung
% https://en.wikipedia.org/wiki/Cholesky_decomposition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [x] = cholesky(A,b)

L = zeros(length(A));

for k = 1:length(A)
    for i = 1:k-1
        sum = 0;
        for j = 1:i-1
            sum = sum - L(i,j)*L(k,j);
        end
        L(k,i) = (A(k,i)+ sum)/L(i,i);
    end

    for j = 1:k-1
        L(k,k) = L(k,k) - L(k,j)^2;
    end
    L(k,k) = sqrt(L(k,k)+A(k,k));
end

x = 0;
Temp = 0;
% Determination of L*Temp = b - Substituation
Temp(1) = b(1)/L(1,1);
for i = 2:length(b)
    Temp(i) = b(i);
    for j = 1:i-1
        Temp(i) = Temp(i) - L(i,j)*Temp(j);
    end
    Temp(i)=Temp(i)/L(i,i);
end

% Determination of Ux=Temp - Replacement
U = transpose(L);
x(length(b))=Temp(length(b))/U(length(b),length(b));
for i = length(b)-1:-1:1
    x(i) = Temp(i);
    for j = i+1:length(b)
```

```
        x(i) = x(i) - U(i,j)*x(j);
    end
    x(i) = x(i)/U(i,i);
end
end
```

myipm.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The power method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Th power method is a numerical methods to determine eigenvalues
% Reference: http://www.math.ohiou.edu/courses/math3600/lecture16.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [v e] = myipm (A , n)

% Performs the inverse power method
% Inputs: A= a square matrix
% n= the number of iterations to perform
% Outputs: v= the estimated eigenvector
% e= the estimated eigenvalue

[ L, U, P] = lu(A );     % LU decomposition of A with pivoting
m = size (A ,1);         % determine the size of A
v = ones (m ,1);         % make an initial vector with ones

for i = 1: n
pv = P*v ;               % Apply pivot
y = L \ pv ;             % solve via LU
v = U \y;

% Figure out the maximum entry in absolute value, retaining its sign
M = max ( v );
m = min ( v );
if abs (M ) >= abs (m )
el = M;
else
el = m;
end
v = v / el ;             % Divide by the estimated eigenvalue of the inverse
of A
end
e = 1/ el ;              % Reciprocate to get an eigenvalue of A
```