UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

COPPE – PROGRAMA DE ENGENHARIA CIVIL (PEC)

CPC752 – ELEMENTOS FINITOS I

PROGRAMA DE ELEMENTOS FINITOS

(Prof. Fernando Luiz Bastos Ribeiro)

Felipe Pinheiro de Souza Bastos

Rafaela Veiga Pillar

Vinicio da Silva Lopes Avelino

Junho de 2013

Índice

[1. Introdução 2](#_Toc358887484)

[2. Metodologia 2](#_Toc358887485)

[2.1. Método do gradiente conjugado 2](#_Toc358887486)

[2.1. Elemento quadrilátero 6](#_Toc358887487)

[3. O programa 9](#_Toc358887488)

[3.1. Implementação do elemento quadrilátero bilinear 9](#_Toc358887489)

[3.2. Implementação do método dos Gradientes Conjugados 15](#_Toc358887490)

[3.3. Adaptação da saída para o ParaView 20](#_Toc358887491)

[4. Exemplos numéricos 22](#_Toc358887492)

[4.1. Resultados 22](#_Toc358887493)

[5. Conclusões 33](#_Toc358887494)

[6. Referências bibliográficas 33](#_Toc358887495)

# Introdução

Este trabalho faz parte da avaliação da disciplina “Elementos Finitos I”, do curso de pós-graduação do Programa de Engenharia Civil (PEC) da COPPE/UFRJ.

O trabalho pode ser dividido em duas partes: elaboração do programa e a parte textual.

Para a elaboração do programa, foi aproveitado o código do programa de elementos finitos apresentado na disciplina, em Fortran 77. Neste programa, estão implementados elementos triangulares lineares e o método de resolução do sistema é direto. A saída foi preparada para o programa View3D.

Na parte do programa, o trabalho consiste em fazer as seguintes alterações:

1. Implementar o elemento quadrilátero bilinear para o Estado Plano de Tensão (EPT) e para o Estado Plano de Deformação (EPD);
2. Implementar o Método dos Gradientes Conjugados para a resolução do sistema;
3. Preparar a saída do programa para o ParaView.

Já na parte textual (este documento), é apresentada a metodologia, como as alterações foram feitas e também é apresentado um exemplo numérico, rodado para diferentes malhas, com os resultados e a comparação entre eles.

# Metodologia

Este item apresenta a fundamentação teórica das modificações implementadas.

## Método do gradiente conjugado

O método do gradiente conjugado é um algoritmo utilizado para soluções numéricas de sistemas particulares de equações lineares de matriz simétrica e positiva definida. O método é iterativo e especialmente útil para solução de sistemas esparsos.

Começamos chamando de função quadrática uma função da seguinte forma:

onde A é uma matriz simétrica, b e x são vetores e c é um escalar.

De acordo com a figura a seguir, observa-se que a função representa uma superfície no espaço. A interseção dessa superfície por planos horizontais correspondem a curvas de nível com o valor f(x) constante.



Superfície no espaço



Curvas de nível da função quadrática

O gradiente de uma função f(x) é sempre ortogonal às curvas de nível e representa a direção em que a função f(x) varia mais rapidamente.

Quando o gradiente de f(x) for igual a zero, ou seja, f’(x)=0, significa que encontrou-se o ponto crítico da função.

A derivada de uma função quadrática é:

A matriz é simétrica. Logo,

Iguala-se o gradiente à zero para se obter o ponto crítico:

Como a matriz A é simétrica e definida positiva, então a função quadrática f(x) é um paraboloide com concavidade para cima, e a solução de f’(x) = 0, ou de Ax = b, é o ponto de mínimo de f(x).

O método tem como objetivo chegar a um ponto extremo do paraboloide seguindo sempre a direção de maior variação.

O método do gradiente conjugado é semelhante ao método do gradiente, mas a principal diferença é que ele não usa como vetor direção o resíduo.

Inicia-se a partir de um ponto x(0), onde o subscrito corresponde ao número da iteração (no caso, é o valor inicial). A seguir, calcula-se:

, onde

O processo é explicado a seguir:

* Toma-se um passo inicial x(0)
* O resíduo e a searchdirection ficam:
* Calcula-se alfa
* A solução x(1) fica
* O novo resíduo torna-se
* Calcula-se a constante Gram-Schmidt, já com o novo resíduo:
* Finalmente, a nova searchdirection fica:
* E o novo alfa:
* A solução x(2) fica:

E assim sucessivamente, até que o vetor x(n) se aproxime o bastante da solução exata, de acordo com a tolerância adotada e requerida no problema.

Embora essa eliminação reduza o esforço por iteração, a desvantagem é que o cálculo de  é feito sem nenhum valor de retorno de . Isso faz com que este cálculo acumule erros de arredondamento. Isto pode ser minimizado retornando-se periodicamente, a cada determinado número de iterações, à equação inicial  para o novo cálculo do resíduo correto.

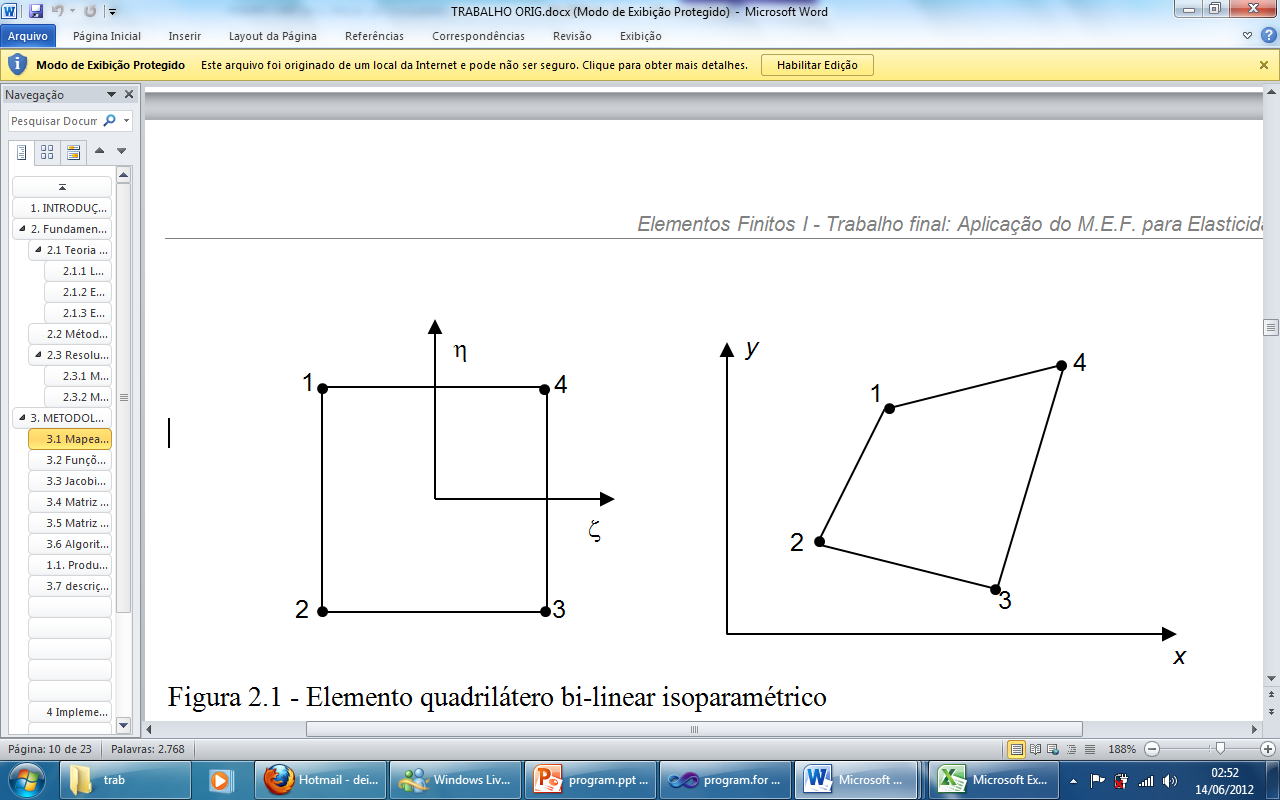
Para que os elementos ocupem o menor espaço possível utiliza-se a técnica do precondicionador, dessa forma o novo sistema será equivalente ao anterior porém com uma solução mais facilmente encontrada. Em geral o precondicionador é uma matriz M responsável pela transformação do sistema.

Neste trabalho, utilizou-se como precondicionador uma matriz diagonal M que contém os valores da diagonal de A, desta forma tem-se o precondicionador diagonal:

, onde jdiag é o vetor apontador dos elementos da diagonal da matriz A.

## Elemento quadrilátero

O elemento quadrilátero implementado no programa tem 4 nós e funções de interpolação lineares ζ e η.



Elemento quadrilátero bilinear isoparamétrico

A geometria é definida por:



As funções de interpolação, correspondentes da família de Lagrange são obtidas a partir da seguinte expressão:



As funções de interpolação do elemento quadrilátero bilinear de 4 nós são:



E suas derivadas:

Assim, a matriz Jacobiana é dada por:



Para a montagem da Matriz de Rigidez da Estrutura é necessário estabelecer a Matriz do Operador Diferencial, dada por:





A matriz rigidez de um elemento parametrizado no domínio natural (ζ,η) pode ser obtida a partir da seguinte expressão:



Como *dΩ =detJd*ζ*d*η, a integral acima pode ser reescrita da forma:



onde *ki,j* é a força nodal no grau de liberdade *i* devido à aplicação de um deslocamento unitário no grau de liberdade *j*.

A integral acima é resolvida numericamente como um somatório.

*ki j* = 

onde:

g (ξi,ηj) = [*B*i]T⋅ [*D*] ⋅ [B]j⋅det J

São usados quatro pontos de integração. Assim:

⎪ξi⎪= ⎢ηi⎢= 0.577350269189626

O peso do ponto de integração vale αi = αj = 1.

Conhecidas a matriz de rigidez e as forças nodais da malha, apenas as equações para os graus de liberdade não restringidos dos nós da malha precisam ser resolvidas. O vetor de forças nodais deve ser então corrigido com o valor das forças internas decorrentes dos graus de liberdade restringidos.

O sistema de equações a ser resolvido é:

{*F*} = [*K*] x {u}

Onde {*F*} é o vetor de forças nodais e [*K*] é a matriz de rigidez global da malha.

# O programa

Neste item é apresentado como foram feitas as alterações no programa.

O código fonte do programa é apresentado no capítulo 7.

## Implementação do elemento quadrilátero bilinear

A implementação dos elementos quadriláteros seguiu a mesma lógica das sub-rotinas dos elementos triangulares. Foi utilizada uma variável *isw* para indicar se a sub-rotina está sendo chamada para leitura das propriedades (*isw*=1) ou para cálculo da matriz de rigidez do elemento (*isw*=2).

Foram implementados os elementos quadriláteros de 4 nós (bilineares) de Estado Plano de Tensão (EPT) e Estado Plano de Deformação (EPD). O primeiro passo para implementar esses elementos no programa foi adicionar os elementos à biblioteca de elementos.

Na sub-rotina *elmlib()* – biblioteca de elementos – foram adicionadas chamadas às sub-rotinas *elmq01()*, correspondente ao elemento quadrilátero bilinear de EPD, e *elmq02()*, correspondente ao elemento quadrilátero bilinear de EPT. A sub-rotina*elmlib()*é apresentada a seguir.

subroutine elmlib(e,xl,ul,fl,sl,nel,iel,ndm,nst,isw,nin)

c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

integer nel,iel,ndm,nst,isw

real\*8 e(\*),xl(\*),ul(\*),fl(\*),sl(\*)

c ..........................................................

goto (100,200,300,400) iel

print\*, ' TIPO DE ELEMENTO ',iel,' NÃO EXISTENTE, ELEMENTO : ',nel

stop

100 call elmt01(e,xl,ul,fl,sl,nel,ndm,nst,isw,nin)

return

200 call elmt02(e,xl,ul,fl,sl,nel,ndm,nst,isw,nin)

return

300 call elmq01(e,xl,ul,fl,sl,nel,ndm,nst,isw,nin)

return

400 call elmq02(e,xl,ul,fl,sl,nel,ndm,nst,isw,nin)

return

end

A sub-rotina *elmq01()* é chamada quando o elemento 3 é utilizado na entrada do programa e a sub-rotina *elmq02()*é chamada quando é escolhido o elemento 4.

Depois de adicionar as chamadas às sub-rotinas, foram criadas as sub-rotinas dos elementos propriamente ditas. A seguir é apresentada a sub-rotina *elmq01()*:

subroutine elmq01(e,x,u,p,s,nel,ndm,nst,isw,nin)

c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

c Estado plano de deformacao - elemento quadrilatero bilinear

c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

integer nel,ndm,nst,isw,i,j,k,l,m,n

real\*8 e(\*),x(ndm,\*),u(nst),p(nst),s(nst,nst),xj(ndm,2)

real\*8 det,hx(4),hy(4),xji(ndm,2),hr(4),hs(4)

real\*8 rr,ss,ri,si,d11,d12,d21,d22,d33

real\*8 my,nu,a,b,c

c .......................................................

goto(100,200) isw

c

c leitura das constantes físicas:

c

100 continue

read(nin,\*) e(1),e(2)

return

c

c

200 continue

c

c Matriz constitutiva:

c

c

my = e(1)

nu = e(2)

a = 1.d0+nu

b = a\*(1.d0-2.d0\*nu)

c = my\*(1.d0-nu)/b

d11 = c

d12 = my\*nu/b

d21 = d12

d22 = c

d33 = my/(2.d0\*a)

c

c Matriz de rigidez:

c

do i = 1, nst

do j = 1, nst

s(i,j)=0.d0

enddo

enddo

do i = 1, 2

do j = 1, 2

if (i .eq. 1 .and. j .eq. 1) then

rr = 0.577350269189626

ss = -0.577350269189626

else if (i .eq. 1 .and. j .eq. 2) then

rr = 0.577350269189626

ss = 0.577350269189626

else if (i .eq. 2 .and. j .eq. 1) then

rr = -0.577350269189626

ss = 0.577350269189626

else if (i .eq. 2 .and. j .eq. 2) then

rr = -0.577350269189626

ss = -0.577350269189626

endif

c derivadas em relacao a r :

c

hr(1) = (1.d0+ss) / 4.d0

hr(2) = - (1.d0+ss) / 4.d0

hr(3) = - (1.d0-ss) / 4.d0

hr(4) = (1.d0-ss) / 4.d0

c

c derivadas em relacao a s :

c

hs(1) = (1.d0+rr) / 4.d0

hs(2) = (1.d0-rr) / 4.d0

hs(3) = -(1.d0-rr) / 4.d0

hs(4) = -(1.d0+rr) / 4.d0

c Matriz Jacobiana

c

do k = 1 , 2

xj(1,k) = 0.d0

xj(2,k) = 0.d0

do l = 1 , 4

xj(1,k) = xj(1,k) + hr(l) \* x(k,l)

xj(2,k) = xj(2,k) + hs(l) \* x(k,l)

enddo

enddo

c

c ... Determinante da matriz Jacobiana:

c

c ......................................................................

det = xj(1,1)\*xj(2,2)-xj(2,1)\*xj(1,2)

c ......................................................................

c

c ... Inversa da matriz Jacobiana:

c

xji(1,1) = xj(2,2) / det

xji(1,2) = -xj(1,2) / det

xji(2,1) = -xj(2,1) / det

xji(2,2) = xj(1,1) / det

c

c ... Derivadas das funcoes de interpolacao:

c

do k = 1, 4

hx(k) = xji(1,1)\*hr(k) + xji(1,2)\*hs(k)

hy(k) = xji(2,1)\*hr(k) + xji(2,2)\*hs(k)

enddo

do m = 1, 4

k = (m-1)\*2+1

do n = 1, 4

l = (n-1)\*2+1

c ------------------------------------------------------------------

s(l,k)= s(l,k) +( hx(n)\*d11\*hx(m) + hy(n)\*d33\*hy(m) ) \* det

c ------------------------------------------------------------------

s(l,k+1)= s(l,k+1)+( hx(n)\*d12\*hy(m) + hy(n)\*d33\*hx(m) )\*det

c ------------------------------------------------------------------

s(l+1,k)=s(l+1,k)+ ( hy(n)\*d12\*hx(m) + hx(n)\*d33\*hy(m) )\*det

c ------------------------------------------------------------------

s(l+1,k+1)=s(l+1,k+1)+( hy(n)\*d22\*hy(m) + hx(n)\*d33\*hx(m))\*det

c ------------------------------------------------------------------

enddo

enddo

enddo

enddo

c

c produto p = s u :

c

call lku(s,u,p,nst)

return

1000 continue

print\*, '\*\*\* Subrotina elmq01: determinante nulo ou negativo do el

.emento ',nel

stop

end

A seguir é apresentada a sub-rotina *elmq02()*:

subroutine elmq02(e,x,u,p,s,nel,ndm,nst,isw,nin)

c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

c Estado plano de tensao - elemento quadrilatero bilinear

c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

integer nel,ndm,nst,isw,i,j,k,l,m,n

real\*8 e(\*),x(ndm,\*),u(nst),p(nst),s(nst,nst),xj(ndm,2)

real\*8 det,hx(4),hy(4),xji(ndm,2),hr(4),hs(4)

real\*8 rr,ss,ri,si,d11,d12,d21,d22,d33

real\*8 my,nu,a,b,c,thic,wt

c .......................................................

goto(100,200) isw

c

c leitura das constantes físicas:

c

100 continue

read(nin,\*) e(1),e(2),e(3)

return

c

c

200 continue

c

c Matriz constitutiva:

c

my = e(1)

nu = e(2)

thic = e(3)

a = my/(1.d0-nu\*nu)

d11 = a

d12 = a\*nu

d21 = d12

d22 = a

d33 = my/(2.d0\*(1.d0+nu))

c

c Matriz de rigidez:

c

do i = 1, nst

do j = 1, nst

s(i,j)=0.d0

enddo

enddo

do i = 1, 2

do j = 1, 2

if (i .eq. 1 .and. j .eq. 1) then

rr = 0.577350269189626

ss = -0.577350269189626

else if (i .eq. 1 .and. j .eq. 2) then

rr = 0.577350269189626

ss = 0.577350269189626

else if (i .eq. 2 .and. j .eq. 1) then

rr = -0.577350269189626

ss = 0.577350269189626

else if (i .eq. 2 .and. j .eq. 2) then

rr = -0.577350269189626

ss = -0.577350269189626

endif

c derivadas em relacao a r :

c

hr(1) = (1.d0+ss) / 4.d0

hr(2) = - (1.d0+ss) / 4.d0

hr(3) = - (1.d0-ss) / 4.d0

hr(4) = (1.d0-ss) / 4.d0

c

c derivadas em relacao a s :

c

hs(1) = (1.d0+rr) / 4.d0

hs(2) = (1.d0-rr) / 4.d0

hs(3) = -(1.d0-rr) / 4.d0

hs(4) = -(1.d0+rr) / 4.d0

c Matriz Jacobiana

c

do k = 1 , 2

xj(1,k) = 0.d0

xj(2,k) = 0.d0

do l = 1 , 4

xj(1,k) = xj(1,k) + hr(l) \* x(k,l)

xj(2,k) = xj(2,k) + hs(l) \* x(k,l)

enddo

enddo

c

c ... Determinante da matriz Jacobiana:

c

c ......................................................................

det = xj(1,1)\*xj(2,2)-xj(2,1)\*xj(1,2)

c ......................................................................

c

c ... Inversa da matriz Jacobiana:

c

xji(1,1) = xj(2,2) / det

xji(1,2) = -xj(1,2) / det

xji(2,1) = -xj(2,1) / det

xji(2,2) = xj(1,1) / det

c

c ... Derivadas das funcoes de interpolacao:

c

wt = det \* thic

do k = 1, 4

hx(k) = xji(1,1)\*hr(k) + xji(1,2)\*hs(k)

hy(k) = xji(2,1)\*hr(k) + xji(2,2)\*hs(k)

enddo

do m = 1, 4

k = (m-1)\*2+1

do n = 1, 4

l = (n-1)\*2+1

c ------------------------------------------------------------------------------

s(l,k) = s(l,k) + ( hx(n)\*d11\*hx(m) + hy(n)\*d33\*hy(m) ) \* wt

c ------------------------------------------------------------------------------

s(l,k+1)= s(l,k+1) + ( hx(n)\*d12\*hy(m)+ hy(n)\*d33\*hx(m)) \* wt

c ------------------------------------------------------------------------------

s(l+1,k)= s(l+1,k) + ( hy(n)\*d12\*hx(m)+hx(n)\*d33\*hy(m) ) \* wt

c ------------------------------------------------------------------------------

s(l+1,k+1)=s(l+1,k+1) + ( hy(n)\*d22\*hy(m)+hx(n)\*d33\*hx(m))\*wt

c ------------------------------------------------------------------------------

enddo

enddo

enddo

enddo

c

c produto p = s u :

c

call lku(s,u,p,nst)

return

1000 continue

print\*, '\*\*\* Subrotina elmq02: determinante nulo ou negativo do el

.emento ',nel

stop

end

O primeiro passo das sub-rotinas é descobrir se elas estão sendo chamadas para ler as propriedades do elemento, quando *isw*=1, ou se elas estão sendo chamadas para calcular a matriz de rigidez do elemento, quando *isw*=2.

1. **Leitura das propriedades**

No caso do elemento de EPD, duas propriedades são lidas: módulo de elasticidade e coeficiente de Poisson, nessa ordem. No caso do elemento de EPT, além dessas propriedades, é lida, em seguida, a espessura do elemento.

1. **Matriz de rigidez**

Na situação em que as sub-rotinas são chamadas para calcular a matriz de rigidez do elemento, os seguintes procedimentos são adotados.

1. Primeiro é calculada a matriz constitutiva;
2. Depois são feitos dois loops, um dentro do outro, para calcular a matriz jacobiana do elemento, utilizando 4 pontos de Gauss. Dentro desses loops são calculadas as derivadas das funções de interpolação Ni(r,s) em relação a r e s, avaliadas em *rr* e *ss* (Pontos de Gauss). Em seguida são feitas as somas para se obterem os termos da matriz jacobiana;
3. Fora dos loops, é calculado o determinante da matriz jacobiana e, em seguida, a inversa da matriz jacobiana.
4. Calcula-se as derivadas das funções de interpolação em relação a x e y.
5. Dentro de dois loops, calculam-se os termos da matriz de rigidez em grupos de 4 por cada iteração: s(l,k),s(l,k+1),s(l+1,k) e s(l+1,k+1).
6. Por fim, chama-se a sub-rotina lku() para transformar deslocamentos prescritos nas direções dos nós do elemento em forças.

Após a realização desses procedimentos, retorna-se a sub-rotina anterior.

## Implementação do método dos Gradientes Conjugados

Para a implementação do Método do Gradiente Conjugado, foram criadas quatro sub-rotinas. Uma delas é a *prodmv()*, uma rotina auxiliar que faz o produto entre uma matriz e um vetor, considerando a forma de armazenamento do programa (*skyline*).

Essa sub-rotina é apresentada a seguir.

subroutine prodmv(jdiag,a,vector,neq)

implicit real\*8 (a-h,o-z)

integer i,j,clnefet,neq

real\*8 vector(neq), aux(neq)

dimension a(\*), jdiag(\*)

do j=1, neq

aux(j)=0

enddo

if(neq .ge. 1) then

aux(1)= a(1)\*vector(1)

endif

if(neq .ge. 2) then

do j=2,neq

clnefet= jdiag(j) - jdiag(j-1)

aux(j)= aux(j) + a(jdiag(j)) \* vector(j)

if (clnefet .ge. 2) then

do i=1, (clnefet-1)

aux(j-i)= aux(j-i) + a(jdiag(j)-i) \* vector(j)

aux(j)= aux(j) + a(jdiag(j)-i) \* vector(j-i)

enddo

endif

enddo

endif

do j=1, neq

vector(j)=aux(j)

enddo

end

Outra rotina auxiliar é a *prodesc()*, que faz o produtos escalar entre dois vetores. Ela é apresentada a seguir.

function prodesc(b,c,neq)

real\*8 b(neq), c(neq)

integer neq

prodesc = 0.0

do i=1,neq

prodesc = prodesc + b(i) \* c(i)

enddo

return

end

A sub-rotina do método do gradiente conjugado é apresentada a seguir.

subroutine grad\_conj(a, b, jdiag, neq, noutrel)

implicit real\*8 (a-h,o-z)

parameter (imax = 400)

integer i/0/, neq, j

dimension a(\*),b(\*),jdiag(\*)

real\*8 r(neq), x(neq), vec(neq), s(neq), delta\_new

real\*8 delta\_0, delta\_old, alfa, beta, q(neq), d(neq), M1(neq)

do j= 1,neq

x(j) = 0.0

M1(j) = 1/(a(jdiag(j)))

vec(j) = 0.0

enddo

call prodmv(jdiag, a, vec, neq)

do j= 1,neq

r(j) = b(j) - vec(j)

d(j) = M1(j)\*r(j)

enddo

delta\_new = prodesc(r,d,neq)

delta\_0 = delta\_new

do while (i .lt. imax .and. delta\_new .gt. delta\_0\*10E-6\*\*2)

do j= 1,neq

vec(j) = d(j)

enddo

call prodmv(jdiag, a, vec, neq)

do j= 1,neq

q(j) = vec(j)

enddo

alfa = delta\_new / prodesc(q,d,neq)

do j= 1,neq

x(j) = x(j) + alfa \* d(j)

enddo

if (i .gt. 50 .and. mod(i,50)\*50 .eq. i) then

vec = x

call prodmv(jdiag, a, vec, neq)

do j= 1,neq

r(j) = b(j) - vec(j)

enddo

else

do j= 1,neq

r(j) = r(j) - alfa \* q(j)

enddo

endif

do j= 1,neq

s(j) = M1(j) \* r(j)

enddo

delta\_old = delta\_new

delta\_new = prodesc(s,r,neq)

beta = delta\_new / delta\_old

do j= 1,neq

d(j) = s(j) + beta \* d(j)

enddo

i = i + 1

enddo

do j= 1, neq

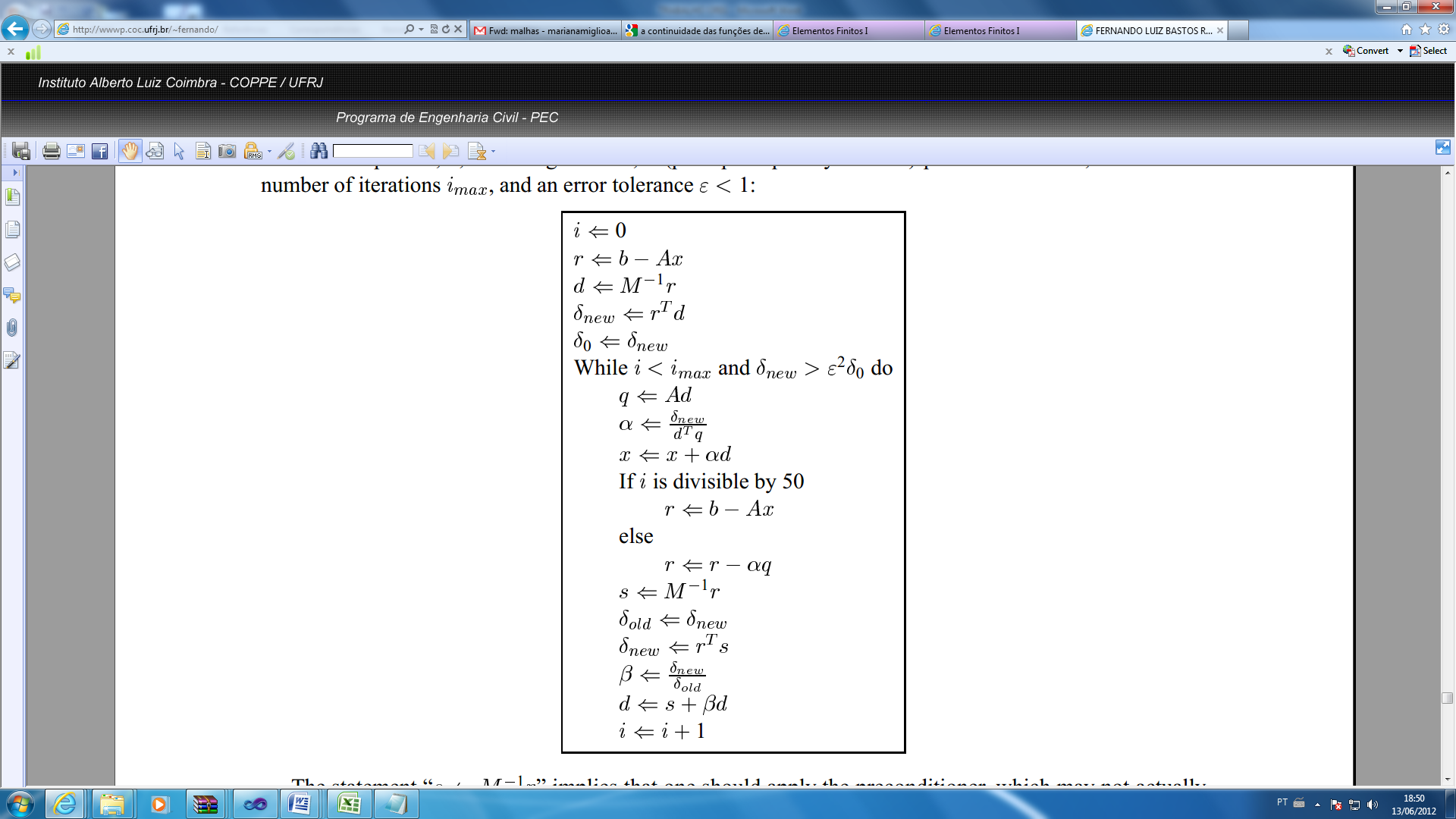
b(j) = x(j)

enddo

call energia(a,b,jdiag,neq,noutrel)

end

O procedimento segue a descrição que foi apresentada anteriormente, na metodologia, e o algoritmo apresentado em “*An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*”, na página 51, da qual foi extraída a figura abaixo.



Verifica-se que quase todas as variáveis são facilmente identificadas. Utilizou-se uma variável auxiliar *vec* (vetor de reais) para armazenar o produto de uma matriz por um vetor; chamou-se de M1 a inversa da matriz M, que é facilmente calculável, já que a matriz M é diagonal.

O processo iterativo é interrompido somente quando se atingir o número máximo de iterações (definido como 400) ou quando *delta\_new* for igual a *delta0* multiplicado por 10-6.

Após o término do método propriamente dito, chama a sub-rotina *energia()* para o cálculo da norma de energia.

Esta rotina é apresentada a seguir.

subroutine energia(a, b, jdiag, neq, noutrel)

implicit real\*8 (a-h,o-z)

integer i, neq, j

real\*8 x(neq), norma

dimension a(\*),b(\*),jdiag(\*)

do j = 1, neq

x(j) = b(j)

enddo

call prodmv(jdiag, a, x, neq)

norma = (prodesc(x,b,neq))\*\*0.5

ue = 1.379745

ee = sqrt(abs(ue\*\*2-norma\*\*2))

eta = ee/ue

write (noutrel,'(a,F8.6)') "||û|| = ", norma

write (noutrel,'(a,F8.6)') "||u||e = ", ue

write (noutrel,'(a,F8.6)') "||e||e = ", ee

write (noutrel,'(a,F8.6)') "eta = ", eta

end

A sub-rotina atribui utiliza uma variável auxiliar *x* (vetor de números reais) que recebe os valores do vetor *b,* vetor que armazena os deslocamentos. Em seguida, ela chama a função *prodmv()*, que faz o produto entre a matriz *a* (matriz de rigidez armazenada em coluna) e o vetor x, considerando o armazenamento do tipo skyline (o resultado retorna no vetor *x*). Depois, ela calcula a norma de energia como a raiz quadrada do produto escalar entre os vetores *x* e *b*.

Os comandos subsequentes foram utilizados para calcular o erro e o erro relativo, que são utilizados para comparar a convergência para as diferentes malhas estudadas.

## Adaptação da saída para o ParaView

No início do programa é perguntado ao usuário qual o arquivo de entrada, qual o nome de arquivo de saída para o ParaView e o nome do arquivo de saída do relatório. Esse trecho do código fonte é apresentado a seguir.

c

c Abertura de arquivos:

c

nin = 1

nout = 2

noutrel = 3

print\*, 'Arquivo de dados:'

read(\*,'(a)') fname

open(nin,file=fname)

print\*, 'Arquivo de saida:'

read(\*,'(a)') fname

open(nout,file=fname)

print\*, 'Arquivo de relatorio:'

read(\*,'(a)') fname

open(noutrel,file=fname)

O arquivo de relatório foi utilizado para facilitar a tomada de dados para comparação entre as malhas. O seu código pode ser visto na sub-rotina do método do gradiente conjugado.

A saída para o programa ParaView foi feita utilizando como referência o documento file-formats-vtk[yy]. Foi escolhido o formato de saída mais simples (*“Simple Legacy Format”*). Essa escolha se deve ao fato de ele ser mais fácil de implementar do que o formato XML. O formato adotado possui a desvantagem de não ter todas as funcionalidades da saída em XML, porém ele atende às necessidades do programa do curso.

O arquivo de saída pode ser dividido nas seguintes partes.

1. Cabeçalho

No cabeçalho, são especificados: a versão do arquivo *vtk* para leitura, o título do arquivo, o formato dos dados (ASCII) e o tipo de geometria/topologia adotada.

1. Representação dos nós

O comando começa com a palavra-chave “POINTS”, em seguida informa o número total de coordenadas que serão lidas (número de pontos x 3 coordenadas espaciais) e depois indica o tipo do dado que será lido (no caso, “float”).

Nas linhas seguintes, são dadas as coordenadas dos nós em relação aos eixos X, Y e Z. Como o problema estudado é de estado plano, todas as coordenadas Z são iguais (e valem 0.0).

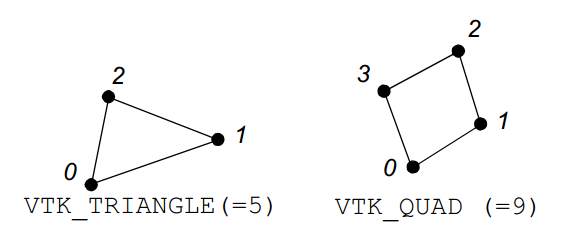
1. Lista de elementos

A primeira linha começa com a palavra-chave “CELLS”, em seguida indica o número de elementos e depois indica a quantidade de campos que serão lidos (que vale o número de elementos vezes (número de nós + 1)), onde o campo adicional serve para indicar o número de nós de cada elemento.

Nas linhas subsequentes são apresentadas, na ordem, o número de nós do elemento e as conectividades dos elementos, conforme definidas no arquivo de entrada do programa.

1. Tipos de elementos

Essa parte começa com a palavra-chave “CELL\_TYPES” e em seguida indica o número de elementos. Foi utilizado o tipo 5 para elementos triangulares e o tipo 9 para elementos quadriláteros, conforme a figura abaixo.



1. Resultados

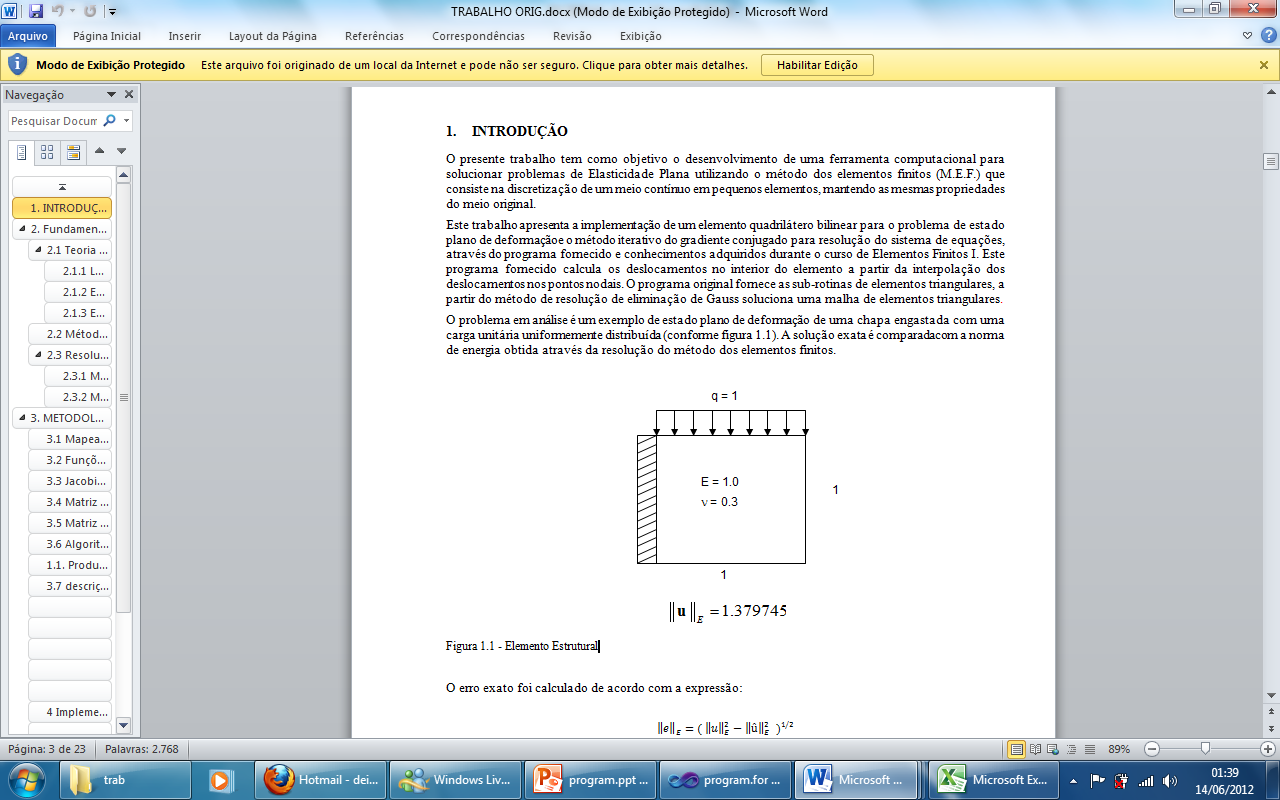
Por fim, são escritos os resultados (deslocamentos). Essa parte começa com a palavra-chave “POINT\_DATA” e em seguida indica o número de pontos (*nnode*). A linha seguinte começa com a palavra-chave “VECTORS”, para indicar que os resultados serão apresentados como vetores, e, em seguida, é indicado o título do resultado (no caso, “Deslocamentos”) e, depois, o tipo de dado “*float*”.

Os deslocamentos são escritos como vetores, com componentes nas direções X e Y obtidas no processamento do programa e 0.0 na direção Z.

Os deslocamentos podem ser lidos no ParaView clicando com o botão direito sobre a malha e escolhendo a opção “Deslocamentos” e, em seguida, a direção desejada.

# Exemplos numéricos

O caso analisado é uma chapa quadrada engastada com uma carga unitária vertical uniformemente distribuída, conforme a figura abaixo.





Foram utilizadas 8 malhas diferentes: 4 malhas com elementos quadriláteros (4x4, 8x8, 16x16 e 32x32) e 4 malhas com elementos triangulares, tomando por base as malhas de quadriláteros e dividindo-se cada quadrilátero em dois triângulos.

## Resultados

O quadro a seguir apresenta o resumo dos resultados encontrados para as 8 malhas analisadas pelo método do gradiente conjugado.



- nel: número de elementos;

- neq: número de equações;

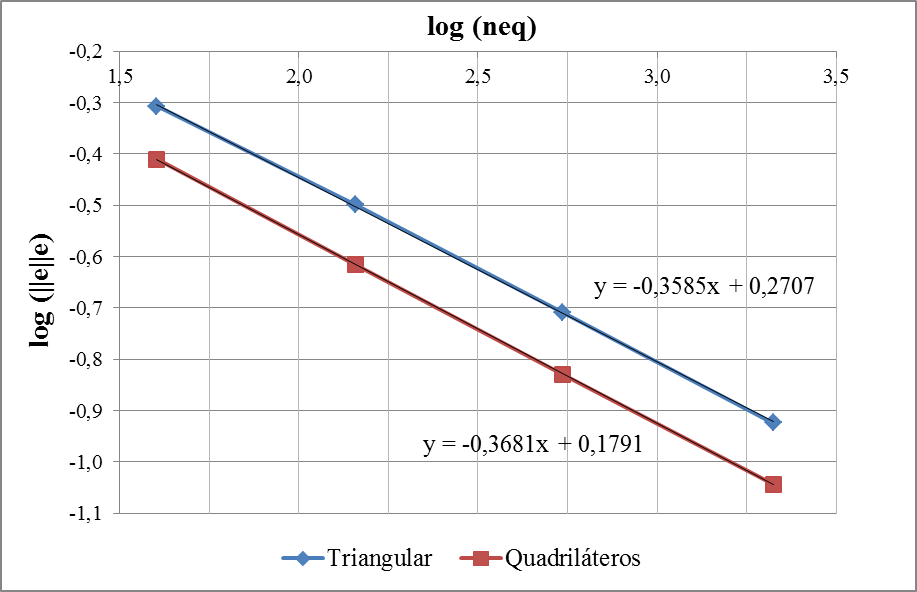
-||u||e: solução exata;

- ||û||e: solução aproximada, obtida pelo programa;

- ||e||e: erro exato (||e||e2 = ||u||e2 - ||û||e2);

- η: Erro relativo, em porcentagem (η = ||e||e/ ||u||e ).

Para determinação da convergência das malhas triangulares e de quadriláteros, foi utilizada a ferramenta da aproximação linear do programa Excel, os valores encontrados para os coeficientes angulares foram **α = -0.3585rad** para a malha triangular e **α = -0.3681rad** para a malha de quadriláteros. As taxas de convergência (**μ = -2α**) encontradas foram **μ = 0.717** para a malha triangular e **μ = 0.7362** para a malha de quadriláteros. O gráfico **log (neq) x log (||e||e)** é apresentado a seguir.

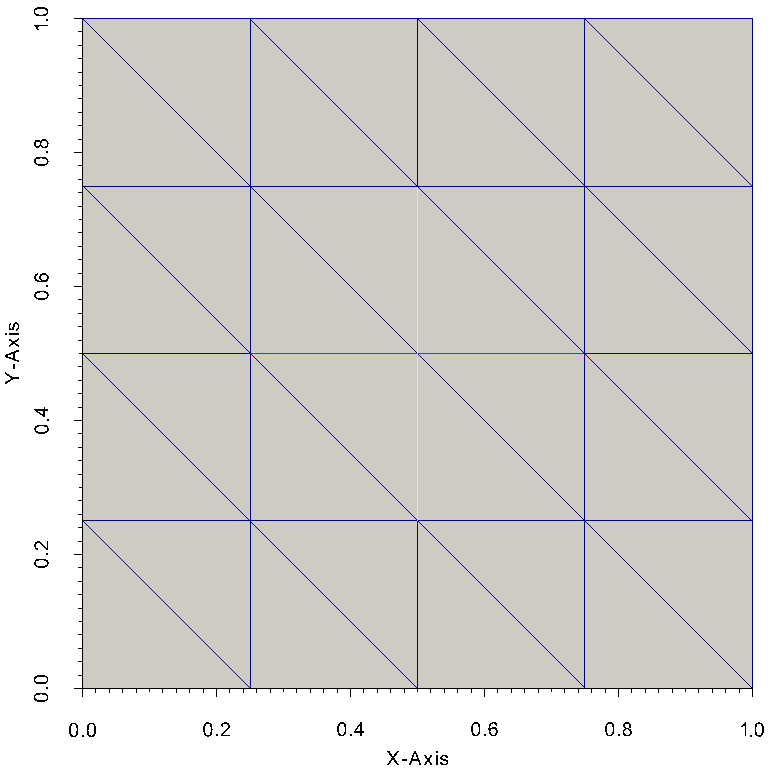


O quadro a seguir relaciona os deslocamentos máximos obtidos na direção Y, para cada malha analisada:

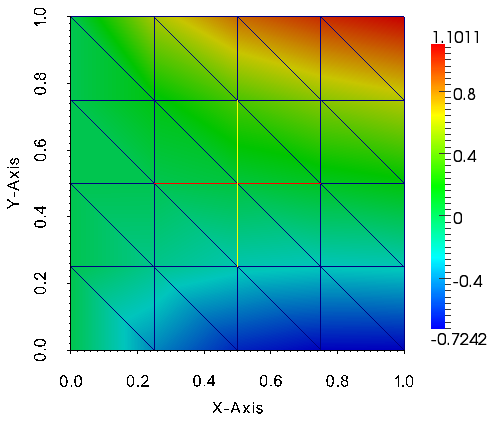
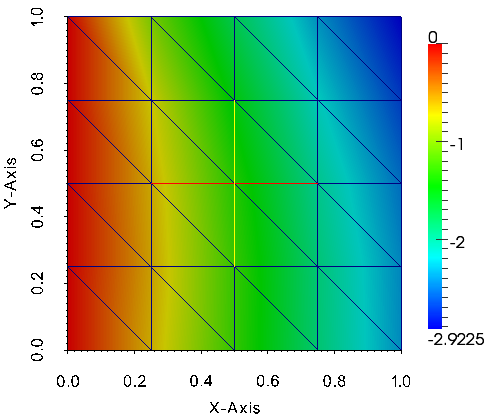


A seguir é apresentada a visualização gráfica dos resultados (deslocamentos nas direções x e y), para obtenção destes gráficos foi utilizado o software livre ParaView.

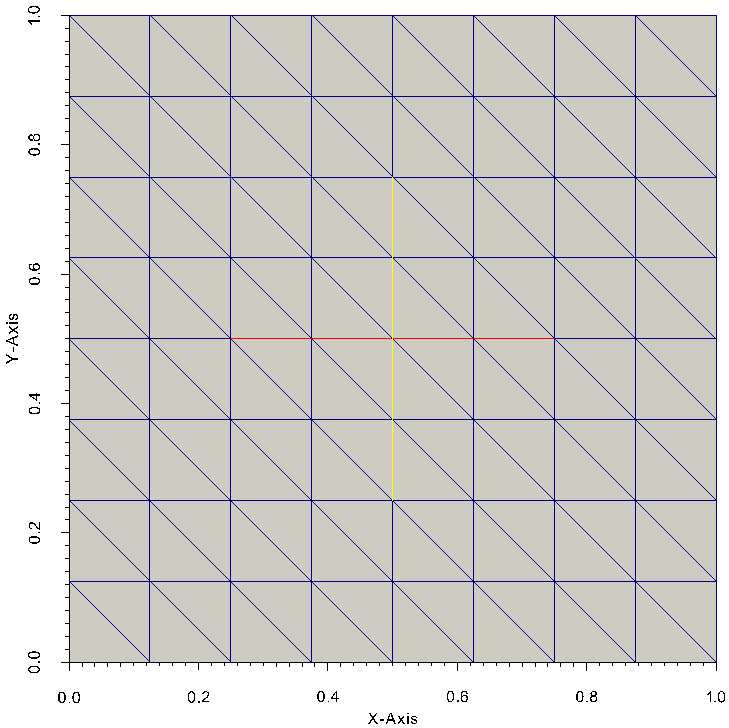
**Elemento triangular malha 4x4: 25 nós, 32 elementos**



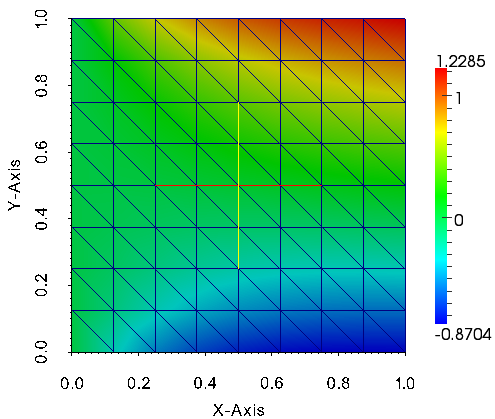
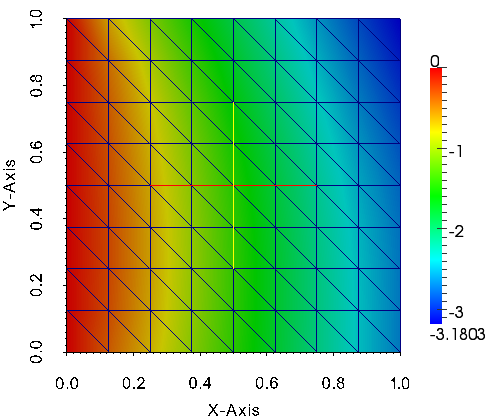
Deslocamentos em X: Deslocamentos em Y:

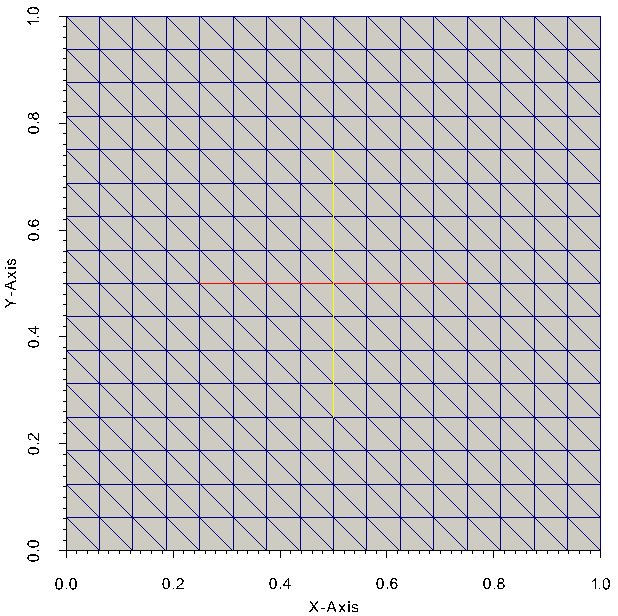
**Elemento triangular malha 8x8: 81 nós, 128 elementos**



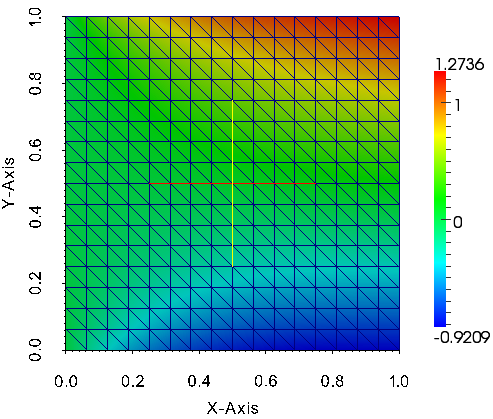
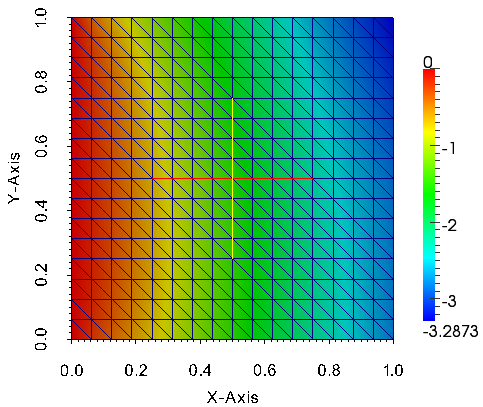
Deslocamentos em X: Deslocamentos em Y:

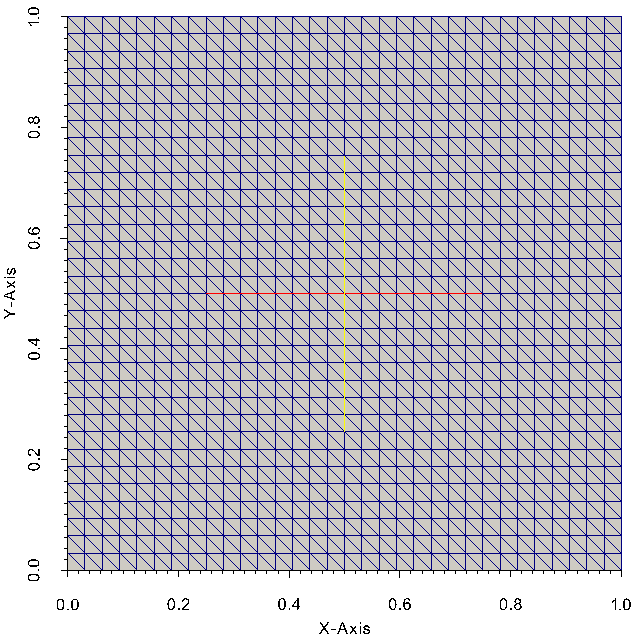
**Elemento triangular malha 16x16: 289 nós, 512 elementos**



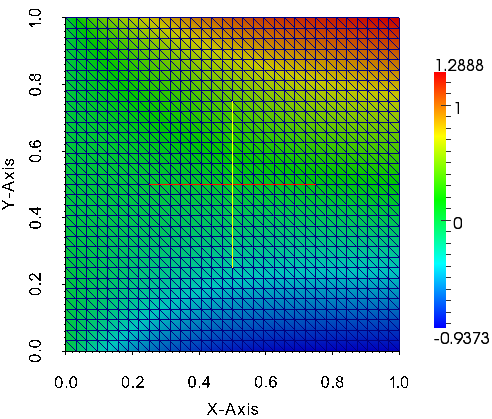
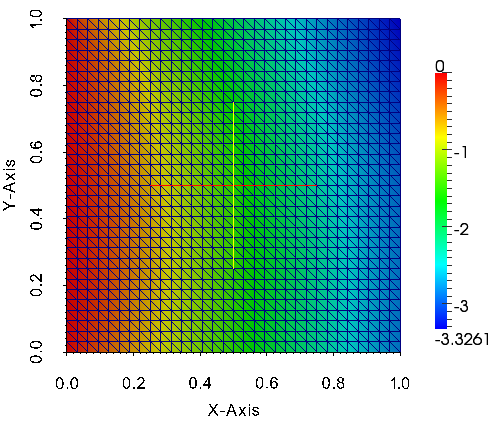
Deslocamentos em X: Deslocamentos em Y:

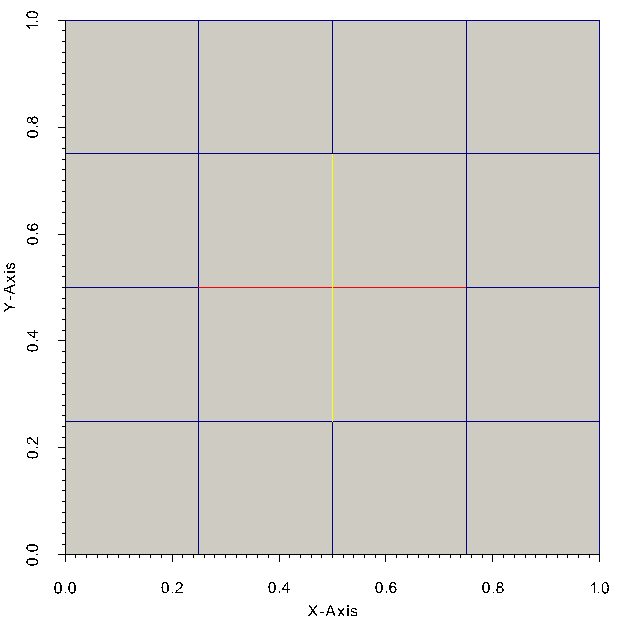
**Elemento triangular malha 32x32: 1089 nós, 2048 elementos**



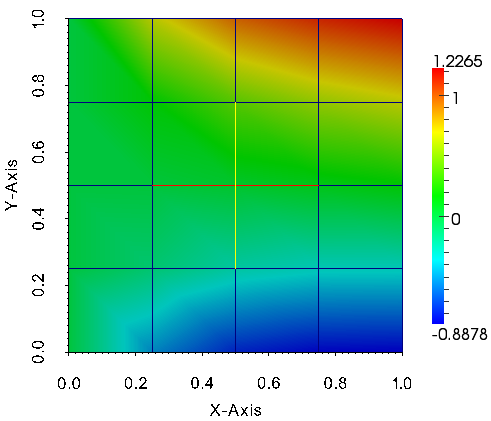
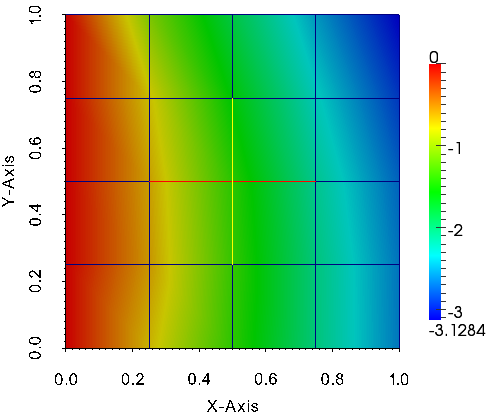
Deslocamentos em X: Deslocamentos em Y:

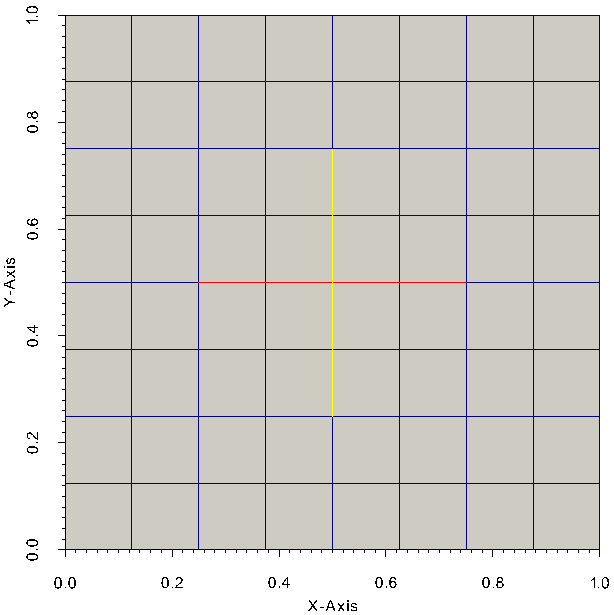
**Elemento quadrilátero malha 4x4: 25 nós, 16 elementos**



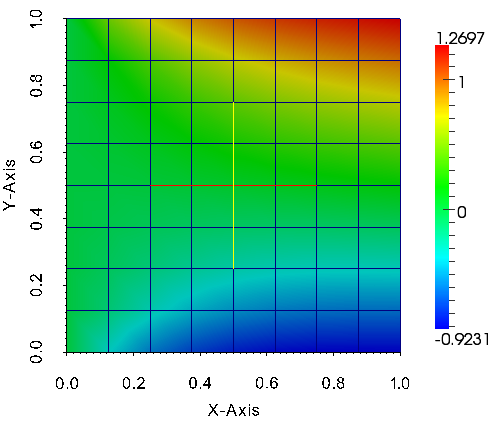
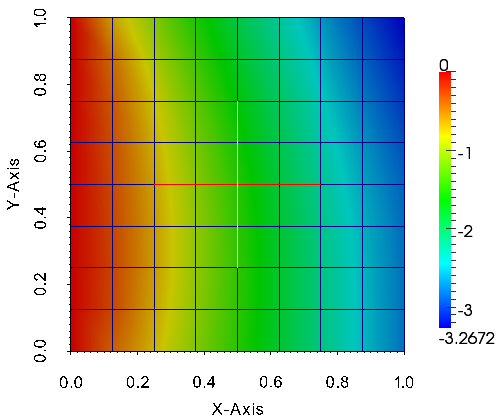
Deslocamentos em X: Deslocamentos em Y:

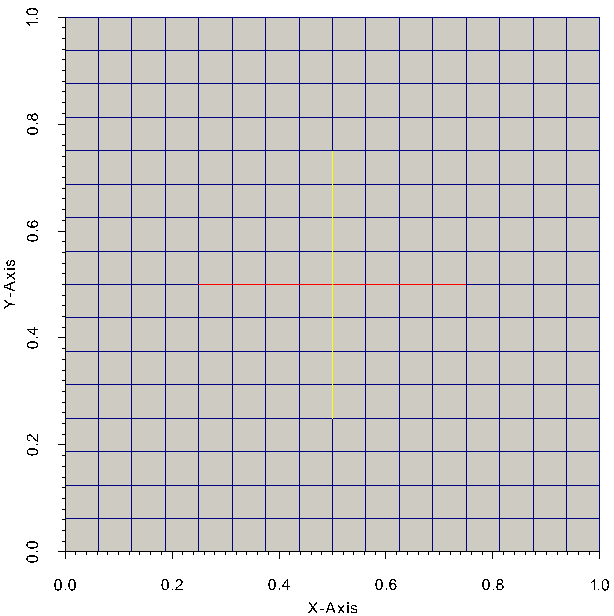
**Elemento quadrilátero malha 8x8: 81 nós, 64 elementos**



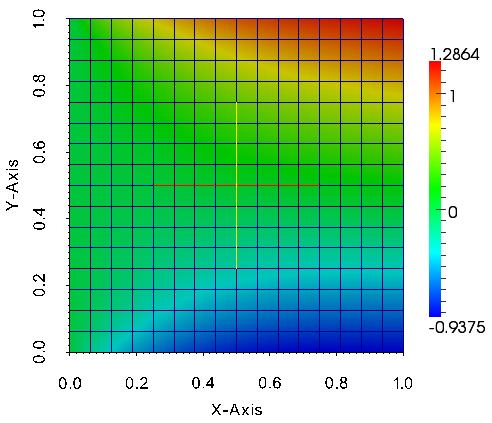
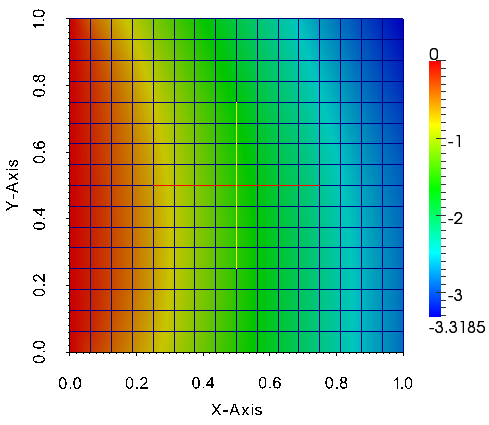
Deslocamentos em X: Deslocamentos em Y:

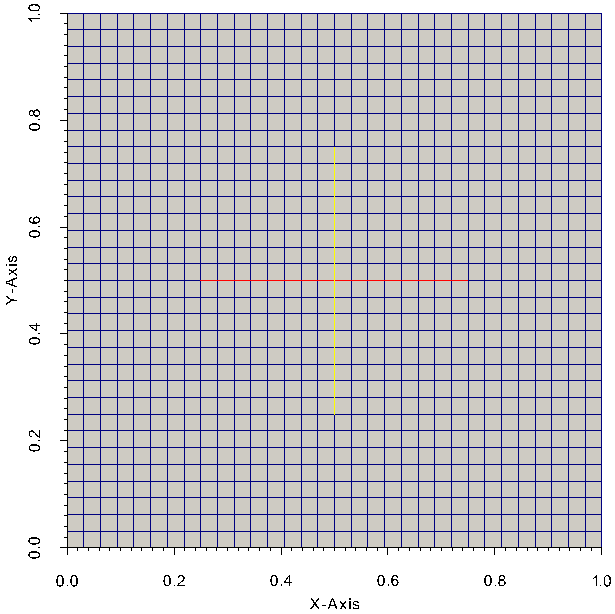
**Elemento quadrilátero malha 16x16: 289 nós, 256 elementos**



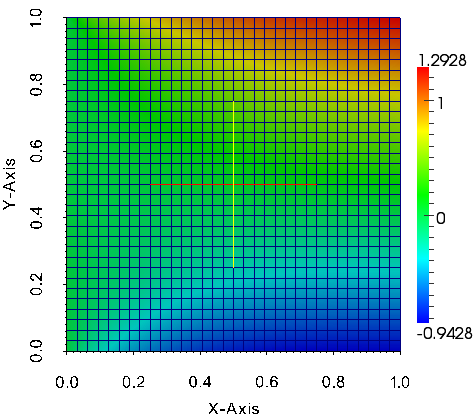
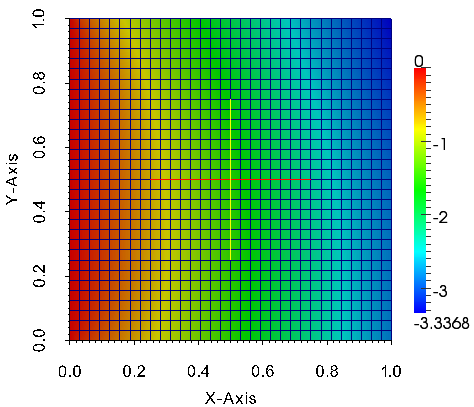
Deslocamentos em X: Deslocamentos em Y:

**Elemento quadrilátero malha 32x32: 1089 nós, 1024 elementos**



Deslocamentos em X: Deslocamentos em Y:

# Conclusões

Analisando os resultados obtidos, percebe-se uma melhor aproximação dos resultados com o refinamento da malha, aumentando-se o número de elementos. Ainda é notável, como esperado, uma melhor aproximação na malha de elementos quadriláteros, em comparação com a malha de elementos triangulares, pois as funções de interpolação utilizadas nos elementos triangulares são lineares, enquanto que nos elementos quadriláteros as funções de interpolação são bilineares, gerando melhores resultados.

O software ParaView, de distribuição gratuita, é continuamente atualizado e permite com facilidade a visualização e personalização dos resultados. Aproveitando as ferramentas disponíveis neste programa, seria válida a implementação, no programa de Elementos Finitos, do cálculo das tensões e deformações na malha, gerando relatório em formato **.vtk**, possibilitando a leitura e visualização gráfica pelo ParaView.

Ainda como sugestão de aperfeiçoamento do programa de Elementos Finitos, seria interessante a implementação de elementos quadriláteros de 8 ou 9 nós, possibilitando a utilização de funções de interpolação de graus mais elevados, melhorando os resultados das aproximações por Elementos Finitos.

# Referências bibliográficas

* Ribeiro, F. L. (2004). *Introdução ao Método dos Elementos Finitos. Notas de Aula.* Rio de Janeiro.
* Gomes, Francisco A. M. *O método dos Gradientes Conjugados*. 2008.
* The VTK User’s Guide – File Formats
* Jonathan Richard Shewchuk. *An Introduction tothe Conjugate Gradient MethodWithout the Agonizing Pain*. Edition 1 ¼. August 4, 1994.