FORMULACIÓN TOTAL DE LAGRANGE

TRABAJO PRÁCTICO 2

Fredy Andrés Mercado Navarro
DNI: 94.872.342
Maestría en Simulación Numérica y Control
Docentes: Eduardo Dvorkin y José Gabriel Hasbani.
Cuatrimestre: I-2013.
Buenos Aires, Argentina

September 7, 2013

Abstract

El desarrollo de este trabajo práctico consiste en la elaboración de un algoritmo que calcule desplazamientos, deformaciones y tensiones en un elemento isoparamtrico de 4 nodos, aplicando la Formulación Total de Lagrange (Total Lagrangian Formulation - TLF), para la solución no lineal del modelo de elementos finitos.

1 DATOS DEL PROBLEMA

Para el desarrollo del modelo asumiremos grandes desplazamientos y pequeñas deformaciones, y un material elástico lineal, cuya ley constitutiva entre el tensor de Piola-Kirchhoff y el tensor de deformaciones de Green-Lagrange es:

$$ {}^{t}_{o}S_{IJ} = \left(K\delta_{IJ}\delta_{MN} + 2GI^{D}_{IJMN}\right) {}^{t}_{o}\epsilon_{MN} $$

$$ I^{D}_{IJMN} = \frac{1}{2}\left(\delta_{IM}\delta_{JN} + \delta_{IN}\delta_{JM} - \frac{2}{3}\delta_{IJ}\delta_{MN}\right) $$

Con:

$$ K = \frac{E}{3(1-2\nu)} $$

$$ G = \frac{E}{2(1+\nu)} $$

$$ E = 2.1e6\frac{kgf}{cm^2} $$

$$ \nu = 0.30 $$

2 RESUMEN DE TEORÍA

Las ecuaciones a resolver para el análisis estático no lineal son:

$$({}^{t}_{o}K_L + {}^{t}_{o}K_{NL})U = {}^{t+\Delta t}R - {}^{t}_{o}F \tag{1}$$

$$\left(\int_{{}_oV} {}^{t}_{o}B^T_L \; {}_oC \; {}^{t}_{o}B_L \; d^oV + \int_{{}_oV} {}^{t}_{o}B^T_{NL} \; {}^{t}_{o}S \; {}^{t}_{o}B_{NL} \; d^oV\right) U = {}^{t+\Delta t}R - \int_{{}_oV} {}^{t}_{o}B^T_L \; {}^{t}_{o}\hat{S} \; d^oV \tag{2}$$

Donde ${}^{t}_{0}B_L$ y ${}^{t}_{0}B_{NL}$ son las matrices de deformaciones-desplazamientos lineal y no lineal respectivamente, ${}_oC$ es la matriz constitutiva elástica lineal para el problema plano de deformaciones, que equivale a la relación constitutiva dada en la sección Datos del Problema, ${}^{t}_{o}S$ es la matriz del segundo tensor de tensiones de Piola-Kirchhoff, U es el vector de incrementos de desplazamientos, ${}^{t+\Delta t}R$ es el vector de cargas correspondiente a cada paso a resolver y ${}^{t}_{o}\hat{S}$ son las componentes del segundo tensor de Piola-Kirchhoff arreglado como vector.

La definición de los términos ${}^{t}_{o}B_L$, ${}^{t}_{o}B_{NL}$, ${}^{t}_{o}S$ y ${}^{t}_{o}\hat{S}$ puede hallarse en las páginas 552 y 553 de la Referencia [1].

Como alternativa, para el cálculo de la matriz ${}^{t}_{o}K_{NL}$ se puede emplear la expresión de la Sección 2.3. Para el calculo de la matriz constitutiva ${}_oC$ se emplea la teoría dada en la Sección 2.4.

2.1 Segundo tensor de tensiones de Piola-Kirchhoff

Para el cálculo de este tensor se emplea el tensor de deformaciones de Green-Lagrange, el cual fue hallado teniendo en cuenta las deformaciones lineales y no lineales a través del cálculo con el tensor gradiente de deformaciones, así:

$$\left[{}^{t}_{o}\epsilon\right] = \begin{bmatrix} {}^{t}_{o}\epsilon_{11} & {}^{t}_{0}\epsilon_{12} \\ {}^{t}_{o}\epsilon_{21} & {}^{t}_{0}\epsilon_{22} \end{bmatrix} = \frac{1}{2}\left[{}^{t}_{o}X^T \, {}^{t}_{o}X - I\right] = \frac{1}{2}\left[{}^{t}_{o}X^T \, {}^{t}_{o}X - [1,0;0,1]\right]$$

Como vector:

$$\left[{}^{t}_{o}\epsilon\right] = \begin{bmatrix} {}^{t}_{o}\epsilon_{11} \\ {}^{t}_{o}\epsilon_{22} \\ 2{}^{t}_{o}\epsilon_{12} \end{bmatrix}$$

Finalmente, se obtiene ${}^{t}_{o}S$ como:

$${}^{t}_{o}S = {}_oC \; {}^{t}_{o}\epsilon$$

2.2 Tensor de tensiones de Cauchy

Se debe realizar una operación de push-forward a las componentes del Segundo tensor de tensiones de Piola-Kirchhoff para obtener tensiones de Cauchy. Teniendo en cuenta la sumatoria de Einstein tenemos:

$$ {}^t\sigma_{mn} = \frac{{}^t\rho}{{}_o\rho}\ {}^t_o x_{m,i}\ {}^t_o x_{n,j}\ {}^t_o S_{i,j} = \frac{1}{det\ {}^t_o X}\ {}^t_o x_{m,i}\ {}^t_o x_{n,j}\ {}^t_o S_{i,j} $$

Matricialmente, también sería:

$$ {}^t\sigma = \frac{{}^t\rho}{{}_o\rho}\ {}^t_o X\ {}^t_o S\ {}^t_o X^T = \frac{1}{det\ {}^t_o X}\ {}^t_o X\ {}^t_o S\ {}^t_o X^T $$

2.3 Matriz de Rigidez tangente No Lineal

Una expresión equivalente para el término

$$ {}^t_o G = {}^t_o B^T_{NL}\ {}^t_o S\ {}^t_o B_{NL} $$

es, para un estado plano de deformaciones:

$$ \left[{}^t_o G\right]_{[2(p-1)+\gamma][2(q-1)+\gamma]} = \left({}_o h_{p,1}\, {}_o h_{q,1}\, {}^t_o S_{11} + {}_o h_{p,2}\, {}_o h_{q,2}\, {}^t_o S_{22}\right) + {}_o h_{p,1}\, {}_o h_{q,2}\, {}^t_o S_{12} + {}_o h_{p,2}\, {}_o h_{q,1}\, {}^t_o S_{12} $$

Esta expresión fue finalmente la empleada en la integración de la matriz de rigidez tangente no lineal del trabajo práctico.

2.4 Estado Plano de Deformaciones

Las deformaciones se encuentran en un plano, mas no las tensiones:

$$ {}^t_o\epsilon_{33} = {}^t_o\epsilon_{13} = {}^t_o\epsilon_{23} = 0 $$
$$ {}^t_o S_{13} = {}^t_o S_{23} = 0 $$

$$ \begin{bmatrix} {}^t_o S_{11} \\ {}^t_o S_{22} \\ {}^t_o S_{33} \\ {}^t_o S_{12} \\ 0 \\ 0 \end{bmatrix} = [C] \begin{bmatrix} {}^t_o\epsilon_{11} \\ {}^t_o\epsilon_{22} \\ 0 \\ {}^t_o\epsilon_{12} \\ 0 \\ 0 \end{bmatrix} $$

$$ \begin{bmatrix} {}^t_o S_{11} \\ {}^t_o S_{22} \\ {}^t_o S_{12} \end{bmatrix} = \frac{E}{(1+v)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \begin{bmatrix} {}^t_o\epsilon_{11} \\ {}^t_o\epsilon_{22} \\ 2\,{}^t_o\epsilon_{12} \end{bmatrix} $$

$$ \begin{bmatrix} {}^t_o\epsilon_{11} \\ {}^t_o\epsilon_{22} \\ 2\,{}^t_o\epsilon_{12} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1-\nu^2 & -\nu(1+\nu) & 0 \\ -\nu(1+\nu) & 1-\nu^2 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} \begin{bmatrix} {}^t_o S_{11} \\ {}^t_o S_{22} \\ {}^t_o S_{12} \end{bmatrix} $$

De ${}^t_o S = C\, {}^t_o\epsilon$ (3D) tengo:

$$ {}^{t}_{o}S_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}({}^{t}_{o}\epsilon_{11} + {}^{t}_{o}\epsilon_{22}) + \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}{}^{t}_{o}\epsilon_{33} $$

${}^{t}_{o}\epsilon_{33} = 0$, luego:

$$ {}^{t}_{o}S_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}({}^{t}_{o}\epsilon_{11} + {}^{t}_{o}\epsilon_{22}) $$

2.5 Esquema de iteración

Si se emplea un esquema de iteración Full Newton-Raphson tendriamos:

$$ \left({}^{t+\Delta t}_{0}K_{L}^{(i-1)} + {}^{t+\Delta t}_{0}K_{NL}^{(i-1)}\right)\Delta U^{(i)} = {}^{t+\Delta t}R - {}^{t+\Delta t}_{0}F^{(i-1)} \tag{3} $$

$$ {}^{t+\Delta t}U^{(i)} = {}^{t+\Delta t}U^{(i-1)} + \Delta U^{(i)} \tag{4} $$

El esquema de iteración de las ecuaciones (3) y (4) se puede resumir en la Figura 1. Para el trabajo práctico se empleó un esquema iterativo Newton-Raphson Modificado, donde la matriz de rigidez tangente (suma de la lineal y la no lineal) es calculada una vez por cada paso de carga (step), en lugar de una vez por cada iteración. Esto para facilitar en un principio la implementación computacional, sin embargo, por la forma en que fue diseñado, es posible modificarlo a Full Newton-Raphson fácilmente.

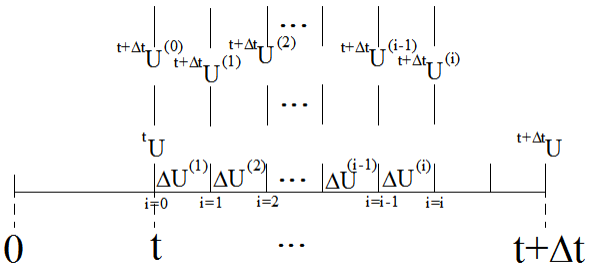

Figure 1: Esquema de iteraciones

2.6 Criterios de Convergencia

2.6.1 Desplazamientos

El criterio para finalizar iteraciones comparando desplazamientos es:

$$ \|\Delta U^{(i)}\|_2 \leq D_{tol}\ \|{}^{t+\Delta t}U^{(i)}\|_2 $$

Donde D_{tol} es la tolerancia del incremento de desplazamientos. La norma 2 de un vector, $\|v\|_2$ esta definida como:

$$ \|v\|_2 = \sqrt{v_1^2 + v_2^2 + ... + v_n^2} $$

Siendo n la dimensión del vector.

2.6.2 Fuerzas desbalanceadas

El criterio para finalizar iteraciones comparando fuerzas desbalanceadas es:

$$\|^{t+\Delta t}R - {}^{t+\Delta t}F^{(i)}\|_2 \leq F_{tol}\ \|^{t+\Delta t}R - {}^{t}F\|_2$$

Donde F_{tol} es la tolerancia de las fuerzas desbalanceadas.

2.6.3 Energía Interna

En este criterio se compara el incremento de energía interna con el incremento de energa interna inicial.

$$\left[\Delta U^{(i)^T}\left({}^{t+\Delta t}R - {}^{t+\Delta t}F^{(i-1)}\right)\right] \leq E_{tol}\left[\Delta U^{(i)^T}\left({}^{t+\Delta t}R - {}^{t}F\right)\right]$$

Donde E_{tol} es la tolerancia del incremento de energía interna.

En el trabajo práctico se emplearon los criterios de convergencia combinados. Se programó para salir del ciclo iterativo cuando se cumpliera el criterio de energías internas con alguno de los otros dos criterios, o en el mejor de los casos, con ambos.

3 ALGORITMO

A continuación se presentan las funciones que integran el algoritmo desarrollado. Sólo se presentan las más importantes. En la Sección 3.1 se presenta la función principal del programa. En la Sección 3.2 se encuentra la función que realiza las integrales del algoritmo, mientra que en Sección 3.3 la función que evalúa la convergencia de la solución:

3.1 Función Principal: Main6.m

Los datos de entrada están almacenados en tablas, que son cargadas en el programa dependiendo del caso de carga elegido. El algoritmo tiene en cuenta los pasos de carga deseados, incrementando la carga con cada step:

```
1  %Elementos Finitos Avanzados - Slidos
2  %Trabajo Prctico 2
3  %Total Lagrangian Formulation
4
5  %FUNCION PRINCIPAL
6
7  clear all
8  clc
9
10 %USUARIO
11     Caso=1;
12     maxite=10;
13     TOL=1e-6;
14     Utol=TOL;
```

```
    Ftol=TOL;
    Etol=TOL;
    NPGauss=2;
    Nsteps=1;
%USUARIO

    if Caso==1
        load 1ECaso1 %Cargan Rultimo,Ui,bcU2D,conec2D,coord2D,material
    elseif Caso==2
        load 1ECaso2
    elseif Caso==3
        load 1ECaso3
    end

E=material(1,1);
nu=material(2,1);
[Uactivos]=vecUactivos(bcU2D);
NGLactivos=sum(Uactivos);

%MEMORIA
    K=zeros(8,8,maxite);
    o_tdt_F=zeros(8,maxite);
    tdt_U=zeros(8,maxite);
    deltaU=zeros(8,maxite);
    deltaR=zeros(8,maxite);
    Kred=zeros(NGLactivos,NGLactivos,maxite);
    deltaR_red=zeros(NGLactivos,maxite);
    deltaU_red=zeros(NGLactivos,maxite);

%DATOS INICIALES
        [oXk,oYk]=coordenadas2D(coord2D,conec2D); %Geometra(X,Y) en t=0
        [C,Ctensor]=evaluaC(material(1,1),material(2,1));

LoadIncrement=Rultimo/Nsteps; %Incremento de carga constante
tdt_R = zeros(8,1);
t_U = [0;0;0;0;0;0;0;0];

for paso=1:Nsteps

    tdt_R = tdt_R + LoadIncrement;

    %Compute ot_K and ot_F
    [ot_K,ot_F,ot_E3x4,Cauchy4x4]=evalua_otKotF(t_U,oXk,oYk,C,NPGauss,E,nu);
                [ot_K_red]=reduceMatriz(ot_K,bcU2D,Uactivos);

    %ITERATION CERO
    o_tdt_Fcero = ot_F;
    tdt_Ucero = t_U;

    for i=1:maxite
        if i==1
            deltaRcero = tdt_R - o_tdt_Fcero;
                    %REDUCCIONES Ki y deltaRi
                    [deltaRcero_red]=reduceVector(deltaRcero,Uactivos);
```

```
                    deltaU_red(:,i) = ot_K_red \ deltaRcero_red;
            [deltaU(:,i)]=expandeVector(deltaU_red(:,i),Uactivos,conec2D,2,4);
                %EXPANSION de deltaUred

            tdt_U(:,i) = tdt_Ucero + deltaU(:,i);
        else
            deltaR(:,i-1) = tdt_R - o_tdt_F(:,i-1);
                    %REDUCCIONES Ki y deltaRi
                    [deltaR_red(:,i-1)]=reduceVector(deltaR(:,i-1),Uactivos);
                    deltaU_red(:,i) = ot_K_red \ deltaR_red(:,i-1);
            [deltaU(:,i)]=expandeVector(deltaU_red(:,i),Uactivos,conec2D,2,4);
                %EXPANSION de deltaUred

            tdt_U(:,i) = tdt_U(:,i-1) + deltaU(:,i);
        end
        [ot_KiNoUsada,o_tdt_F(:,i),o_tdt_E3x4,Cauchy4x4]=evalua_otKotF(tdt_U
            (:,i),oXk,oYk,C,NPGauss,E,nu);

        %CONVERGENCIA
        [SIoNO]=convergencia(Utol,Ftol,Etol,deltaU,tdt_U(:,i),tdt_R,o_tdt_F(:,
            i),ot_F,o_tdt_F,i);
                if SIoNO>0
                    break;
                end
    end

    %Al final de este ciclo de iteraciones s lo
    %necesito el DESPLAZAMIENTO al que se lleg .
    %Luego lo voy a actualizar como tU para volver
    %a iterar.

    t_U = tdt_U(:,i);
end
```

3.2 Función de Integraciones: evaluaotkotF.m

Esta función se encarga de integrar numéricamente las matrices de rigidez tangentes, el vector de fuerzas equivalentes a tensiones y calcular las deformaciones y tensiones en cada punto de Gauss.

```
function [ot_K,ot_F,ot_EvecPrint,Cauchy4x4]=evaluaotKotF(t_U,oXk,oYk,C,NPGauss
    ,E,nu)

%Evalua ot_K y ot_F

ot_K=zeros(8,8);
ot_KL=zeros(8,8);
ot_KNL=zeros(8,8);

ot_F=zeros(8,1);
Cauchy4x4=zeros(4,4);
ot_EvecPrint=zeros(4,3);
```

```
12
13 if NPGauss==2
14     puntogauss=[-1/sqrt(3),1/sqrt(3)];
15     peso=[1,1];
16 end
17
18     %PuntoGauss1(-r,-s)
19     %PuntoGauss2(-r,+s)     2   4
20     %PuntoGauss3(+r,-s)     1   3
21     %PuntoGauss4(+r,+s)
22
23 puntoG=0;
24 for jj=1:NPGauss
25     for ii=1:NPGauss
26         puntoG=puntoG+1;
27
28         r=puntogauss(jj);
29         s=puntogauss(ii);
30
31         alpha_r=peso(ii);
32         alpha_s=peso(jj);
33
34         [ohkx,ohky,oJ]=evalua0Hkx(r,s,oXk,oYk);
35
36         [ot_BL0]=evaluaBL0(ohkx,ohky);
37         [ot_BL1]=evaluaBL1(ohkx,ohky,t_U); %OK, es tU - Bathe.
38             ot_BL = ot_BL0 + ot_BL1;
39
40         %KL INTEGRAL
41         FL = alpha_r * alpha_s * ( ot_BL' * C * ot_BL * det(oJ) );
42         ot_KL(:,:) = ot_KL(:,:) + FL;
43
44         %GREEN-LAGRANGE
45         tu = [t_U(1),t_U(3),t_U(5),t_U(7)];
46         tv = [t_U(2),t_U(4),t_U(6),t_U(8)];
47         [ot_X]=calcGradiente(ohkx,ohky,tu,tv);
48         ot_E = (1/2)*(ot_X'*ot_X-[1,0;0,1]);
49         ot_Evec=[ot_E(1,1);ot_E(2,2);2*ot_E(1,2)];
50         ot_EvecPrint(puntoG,:)=ot_Evec';
51
52         %SECOND PIOLA-KIRCHHOFF
53         ot_S1a3=C*ot_Evec;
54
55         %CAUCHY 4x4
56         [ot_X3x3]=calcGradiente2(ohkx,ohky,tu,tv);
57         [ot_S33]=evaluaS33(E,nu,ot_Evec);
58         ot_Smatriz3x3 = [ot_S1a3(1,1),ot_S1a3(3,1),0;...
59                          ot_S1a3(3,1),ot_S1a3(2,1),0;...
60                          0,0,ot_S33];
61         [CauchyVecFila1x4]=calcCauchy3(ot_Smatriz3x3,ot_X3x3);
62         Cauchy4x4(puntoG,:)=CauchyVecFila1x4;
63
64         %KNL INTEGRAL
65         [evKNL]=evaluaKNL(ohkx,ohky,ot_S1a3);
```

```
66          FNL = alpha_r * alpha_s * ( evKNL * det(oJ) );
67          ot_KNL(:,:) = ot_KNL(:,:) + FNL;
68
69          %F INTEGRAL
70          FF = alpha_r * alpha_s * ( ot_BL' * ot_S1a3 * det(oJ) );
71          ot_F = ot_F + FF;
72      end
73  end
74
75  ot_K(:,:) = ot_KL(:,:) + ot_KNL(:,:);
```

3.3 Función de Convergencia: convergencia.m

Esta función tiene en cuenta las tolerancias y evalúa la convergencia o no de la solución.

```
1  function [SIoNO]=convergencia(Utol,Ftol,Etol,deltaU,tdt_Ui,tdt_R,tdt_Fi,t_F,
       tdt_F,i)
2
3  SIoNO=0;
4
5  if i==1
6      tdt_Fim1 = t_F;
7  else
8      tdt_Fim1 = tdt_F(:,i-1);
9  end
10
11 ULEFT = norm(deltaU(:,i),2);
12 URIGHT = norm(tdt_Ui,2);
13
14 FLEFT = norm((tdt_R - tdt_Fi),2);
15 FRIGHT = norm((tdt_R - t_F),2);
16
17 ELEFT = deltaU(:,i)'*(tdt_R - tdt_Fim1);
18 ERIGHT = Etol*(deltaU(:,1)'*(tdt_R - t_F));
19
20 if (ULEFT <= Utol*URIGHT) && (ELEFT <= Etol*ERIGHT)
21     SIoNO=SIoNO+1;
22 end
23
24 if (FLEFT <= Ftol*FRIGHT) && (ELEFT <= Etol*ERIGHT)
25     SIoNO=SIoNO+2;
26 end
```

4 RESULTADOS

El algoritmo calcula desplazamientos, deformaciones de Green-Lagrange y tensiones de Cauchy para cada uno de los tres estados de carga propuestos por el enunciado del trabajo práctico. Los resultados de las corridas fueron comparados con modelos realizados en el software de elementos finitos ADINA. Para todos los casos se calcularon resultados con

una tolerancia de 1e-6 para desplazamientos, fuerzas y energía.

Los resultados encontrados corriendo el programa de Matlab coinciden exactamente con los obtenidos mediante los modelos de ADINA a excepción de aquellos números con ordenes de magnitud muy pequeños (1e-19).

4.1 Caso de cargas 1

Para este caso el elemento está sometido a tensión en los nodos superiores. La tensión en +Y es igual a 3000 kgf en cada nodo.

Los desplazamientos obtenidos fueron:

P. Gauss	Desp. X	Desp. Y
11	-1.1135e-03	2.5966e-03
12	-3.7847e-19	2.5966e-03
21	0	0
22	-1.1135e-03	0

Las deformaciones obtenidas fueron:

P. Gauss	Defor. XX	Defor. YY	Defor. XY	Defor. ZZ
11	-3.7111e-04	8.6592e-04	-9.4799e-20	0
12	-3.7111e-04	8.6592e-04	-9.4799e-20	0
21	-3.7111e-04	8.6592e-04	-9.4799e-20	0
22	-3.7111e-04	8.6592e-04	-9.4799e-20	0

Las tensiones obtenidas fueron:

P. Gauss	Tension XX	Tension YY	Tension XY	Tension ZZ
11	-9.7398e-09	2.0007e+03	-7.6812e-14	599.1851
12	-9.7398e-09	2.0007e+03	-1.1078e-14	599.1851
21	-9.7398e-09	2.0007e+03	-6.5843e-14	599.1851
22	-9.7398e-09	2.0007e+03	-1.0843e-16	599.1851

4.2 Caso de cargas 2

Para este caso el elemento está sometido a cortante puro con una fuerza en +X en el nodo superior derecho. La magnitud de esta fuerza son 3000 kgf.

Los desplazamientos obtenidos fueron:

P. Gauss	Desp. X	Desp. Y
11	7.4775e-03	-3.0354e-03
12	6.1794e-03	1.9104e-03
21	0	0
22	1.2921e-03	0

Figure 2: Esquema de solución del Caso de cargas 1

Las deformaciones obtenidas fueron:

P. Gauss	Defor. XX	Defor. YY	Defor. XY	Defor. ZZ
11	4.3127e-04	2.9056e-04	1.7126e-03	0
12	4.3323e-04	2.9056e-04	7.6054e-04	0
21	4.3127e-04	-6.6107e-04	1.7141e-03	0
22	4.3323e-04	-6.6107e-04	7.6295e-04	0

Las tensiones obtenidas fueron:

P. Gauss	Tension XX	Tension YY	Tension XY	Tension ZZ
11	1.5771e+03	1.3427e+03	1.3855e+03	8.7390e+02
12	1.5771e+03	1.3445e+03	6.1500e+02	8.7627e+02
21	4.2444e+02	-1.3458e+03	1.3816e+03	-2.7847e+02
22	4.2681e+02	-1.3440e+03	6.1291e+02	-2.7690e+02

4.3 Caso de cargas 3

Para este caso el elemento está sometido a tensión en +Y con 3000 kgf en cada nodo superior y 3000 kgf en +X en el nodo superior derecho.

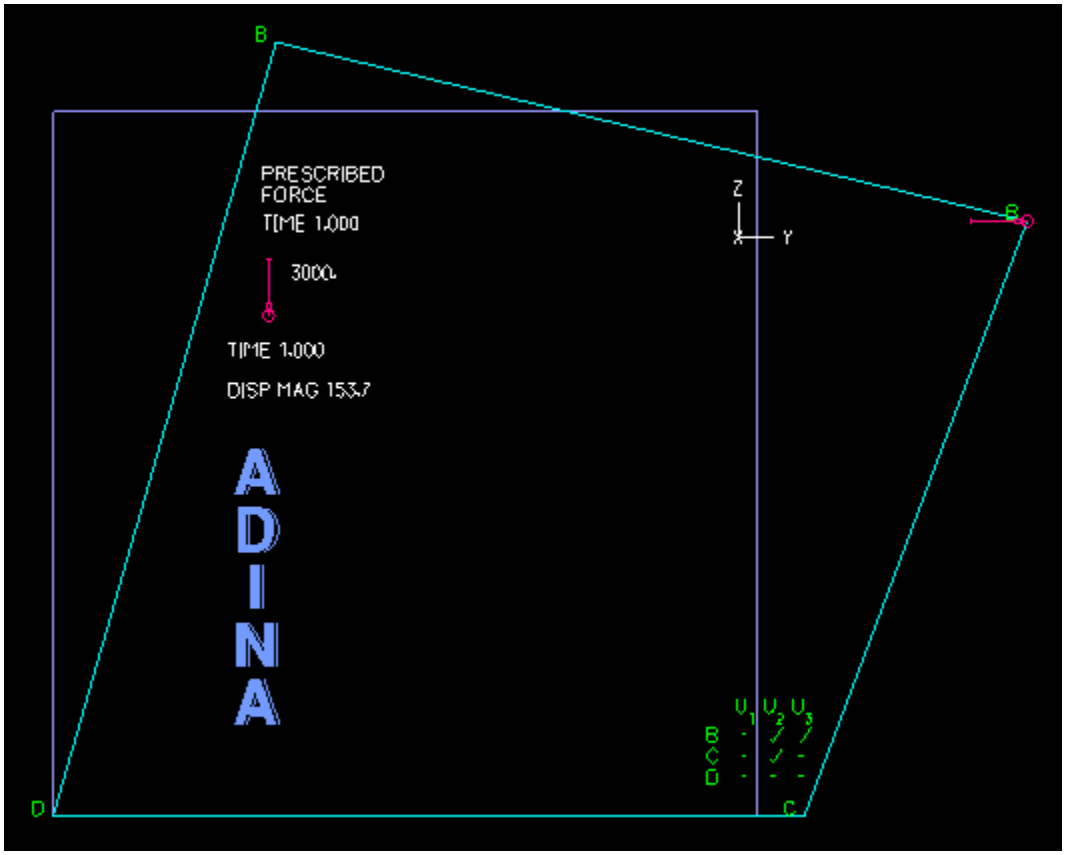

Figure 3: Esquema de solución del Caso de cargas 2

Los desplazamientos obtenidos fueron:

P. Gauss	Desp. X	Desp. Y
11	6.3427e-03	-4.2524e-04
12	6.1572e-03	4.4951e-03
21	0	0
22	1.7935e-04	0

Las deformaciones obtenidas fueron:

P. Gauss	Defor. XX	Defor. YY	Defor. XY	Defor. ZZ
11	6.0272e-05	1.1545e-03	1.7060e-03	0
12	6.2216e-05	1.1545e-03	7.5797e-04	0
21	6.0272e-05	2.0698e-04	1.7075e-03	0
22	6.2216e-05	2.0698e-04	7.6036e-04	0

Las tensiones obtenidas fueron:

P. Gauss	Tension XX	Tension YY	Tension XY	Tension ZZ
11	1.5731e+03	3.3395e+03	1.3842e+03	1.4700e+03
12	1.5754e+03	3.3412e+03	6.1702e+02	1.4723e+03
21	4.2675e+02	6.5727e+02	1.3803e+03	3.2370e+02
22	4.2911e+02	6.5900e+02	6.1494e+02	3.2605e+02

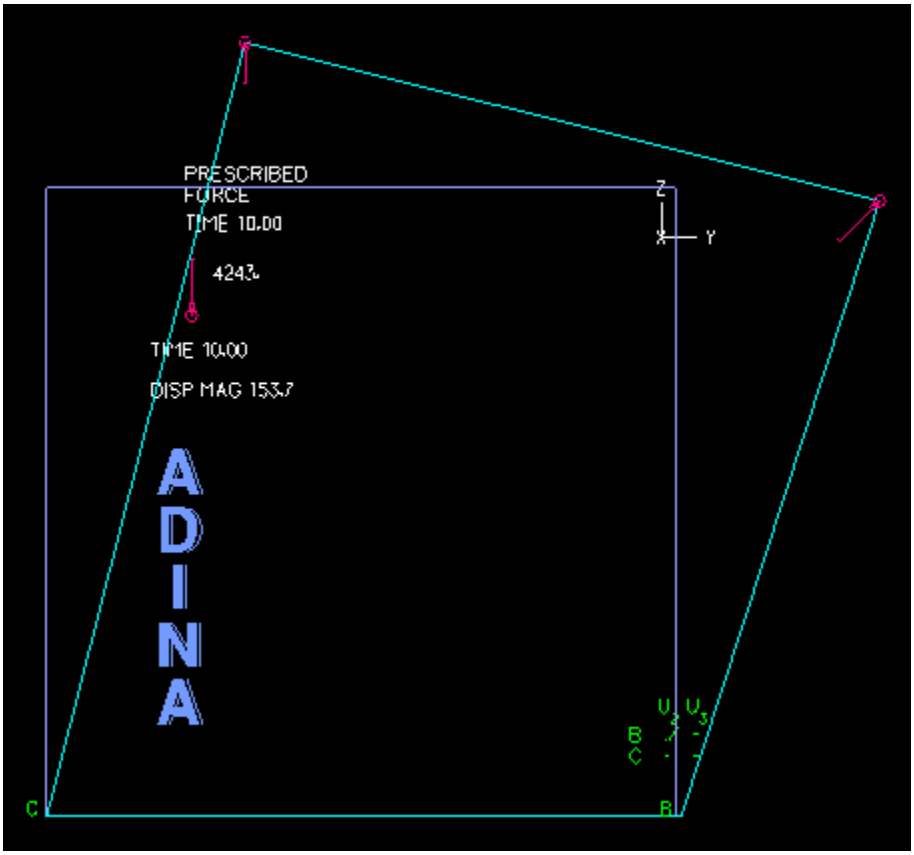

Figure 4: Esquema de solución del Caso de cargas 3

5 CONCLUSION

Se desarrolló un algoritmo que resuelve problemas estáticos con no linealidad geométrica empleando la Formulación Total de Lagrange y para problemas de estado plano de deformaciones (2D). Los resultados obtenidos de los desplazamientos, deformaciones y tensiones fueron comparados con la solución del software comercial ADINA, consiguiéndose resultados exactos en los cálculos a excepción de aquellos números con ordenes de magnitud muy pequeños.

References

[1] Klaus Jurgen Bathe. *Finite Element Procedures*. Prentice-Hall, 1996.