2° Práctica de Cálculo por Elementos Finitos - MC516

Josue Huaroto Villavicencio - 20174070I
Sección: E

5 de julio de 2020

1 Diagrama de flujo

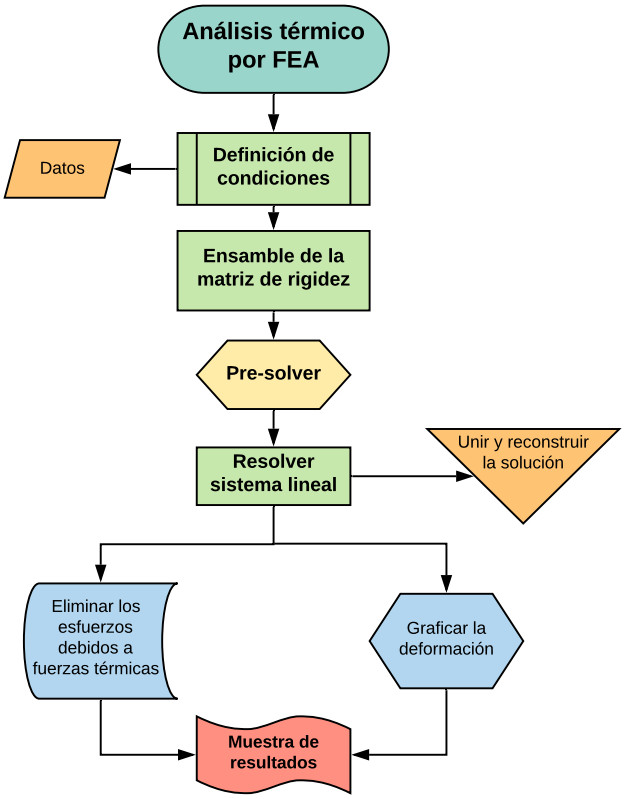

Figura 1: Diagrama de flujo

2 Ejecución del código

El solver principal se describió a detalle en el laboratorio anterior; por lo que, para este nuevo problema solo es necesario modificar las nuevas condiciones del problema.

Código 1: Condiciones del problema

```
NodesCondition = []
Nodes = 4 + 1
E = 2*1e5 #MPA
L = 1500 #mm
b0 = 1000 #mm
bf = 0 #mm
e = 120 #mm
P = 50000 #N
K = []
y = 0.0784532e-3 #N/mm3
```

```
A = []
alpha = 11e-6
DeltaT = 85
dL = L/(Nodes-1)

for i in range(0,Nodes):
    b = b0-i*((b0-bf)/(Nodes-1))
    nb = b0-(i+1)*(b0-bf)/(Nodes-1)
    A.append((b+nb)*e/2)
for i in range(0,Nodes-1):
    K.append([E*A[i]/(dL),i,i+1])

NumberOfElement = len(K)
StiffnessMatrix = np.zeros((Nodes,Nodes))
U = np.zeros(Nodes).reshape(Nodes,1)
F = np.zeros(Nodes).reshape(Nodes,1)
Initialize(StiffnessMatrix,U,F)
UBoundaryCondition(U,0,0)

for i in range(1,Nodes):
    W_T = E*(A[i-1])*alpha*DeltaT
    W = y*0.5*(A[i]+A[i-1])*dL
    if(i == 1):
        W += y*0.5*(A[i-1])*dL
    FBoundaryCondition(F,W+W_T,i)
    FBoundaryCondition(F,-W_T,i-1)

FBoundaryCondition(F,y*0.5*A[Nodes-1]*dL,Nodes-1)
FBoundaryCondition(F,P,NumberOfElement//2)

U,F=Solve(StiffnessMatrix,U,F)

print("Stiffness Matrix:\n",StiffnessMatrix,'\n')
print("Displacements:\n",U,'\n')
print("Forces:\n",F)
```

Como puede observar, el código cambio realmente poco. Las modificaciones más significativas en el código recaen en el añadido de *fuerzas térmicas*. También, se hizo una pequeña modificación a la formulación del modelo, para mejorar el tiempo de ejecución, haciendo que este sea 2 veces más rápido que el anterior:

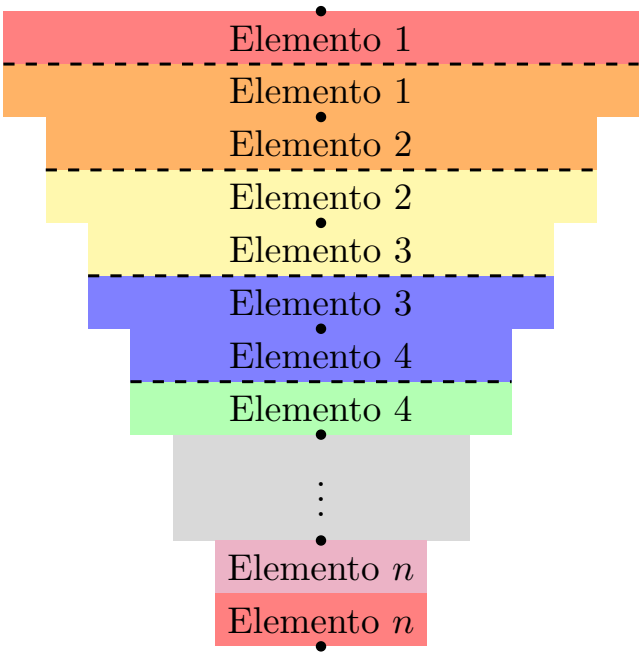

Figura 2: Representación esquemática del mallado

Al hacer el mallado de la geometría, notamos que no es posible añadir el peso en el centro de gravedad de un elemento porque dicho nodo no existe para tal elemento. En la figura 2 puede observar que si imaginariamente partimos un elemento en dos mitades y lo uniéramos a la mitad inferior del elemento siguiente, entonces existe un nodo en el centro de dicha unión; podemos aprovechar este nodo para colocar el peso de la mitad del elemento i y la mitad del elemento $i+1$.

1. Mallado con n elementos y $n+1$ nodos
2. Iterar sobre elementos del 1 al $n-1$
3. Para un elemento i, colocamos el peso equivalente a la mitad del peso del elemento i y $i+1$ juntos en el nodo $i+1$.
4. Colocamos la mitad faltante del elemento 1 en el nodo 2 y la mitad faltante del elemento n en el nodo n.

3 Resultados del problema

Al finalizar la ejecución del solver, obtenemos las siguientes gráficas:

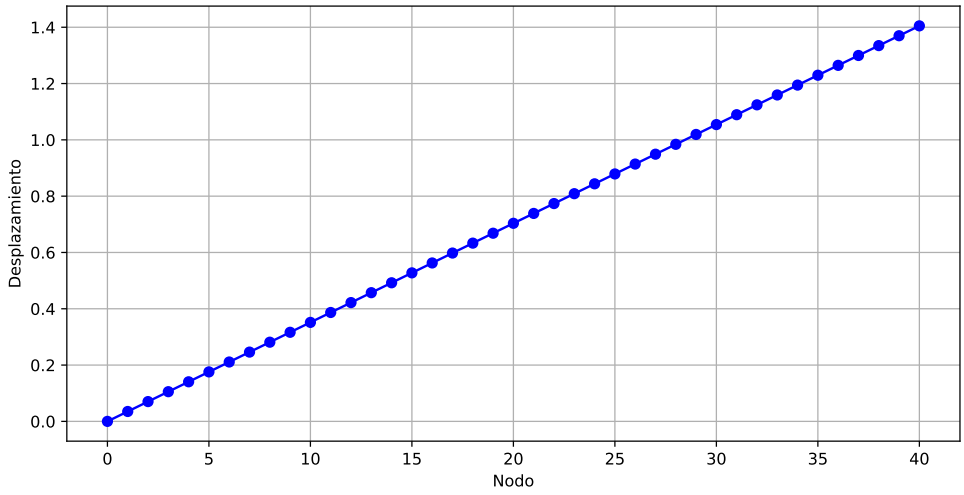

Figura 3: Desplazamientos con 40 elementos

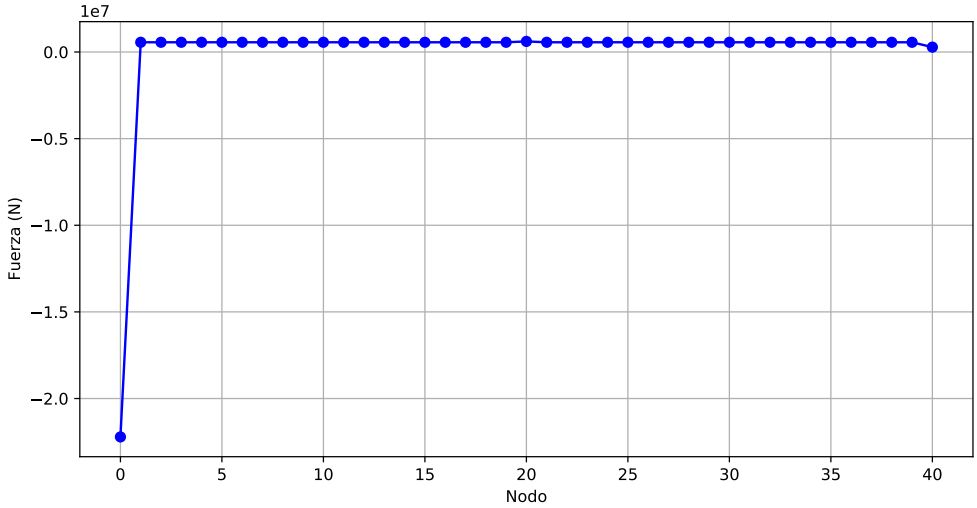

Figura 4: Fuerzas con 40 elementos

Sin embargo; la gráfica de las fuerzas no es correcta, pues nos dice en realidad la reacción considerando la *fuerza térmica*, que se incluyó solo por motivos de deformación. Como tal, dicha fuerza debe ser removida para la gráfica de esfuerzos:

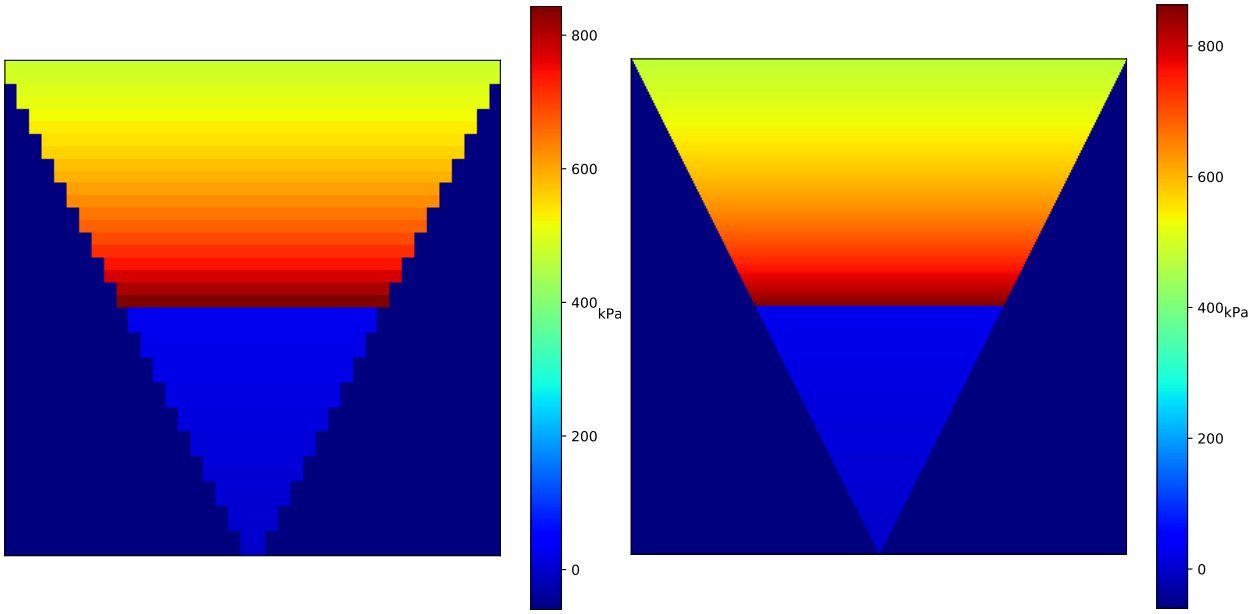

Figura 5: Esfuerzos para 40 y 5000 elementos

Se observa, que luego de rectificar los esfuerzos para eliminar el componente térmico añadido, la gráfica de esfuerzos es la misma que para el caso no térmico.

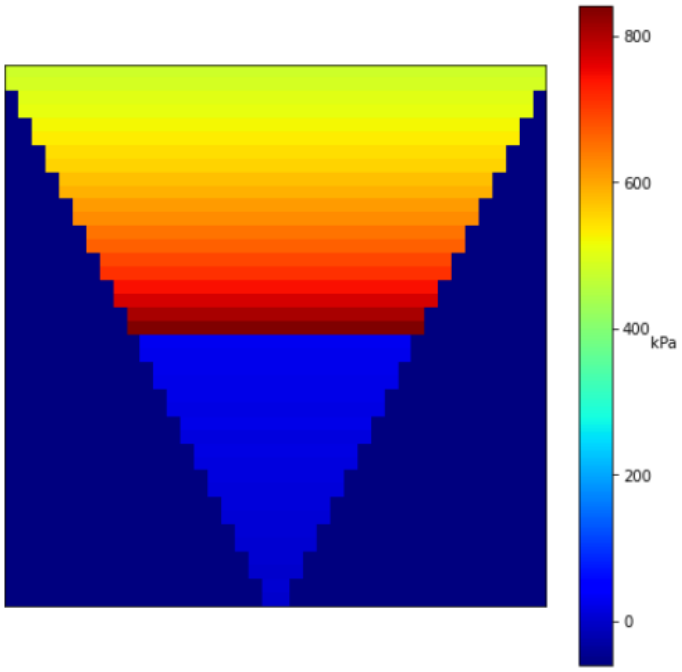

Figura 6: Esfuerzos con 40 elementos para el caso no térmico

Esto comprueba con la teoría que los esfuerzos térmicos no son considerados debido a que su naturaleza no es mecánica, y solo deben tomarse en cuenta para las deformaciones como se detalla a continuación:

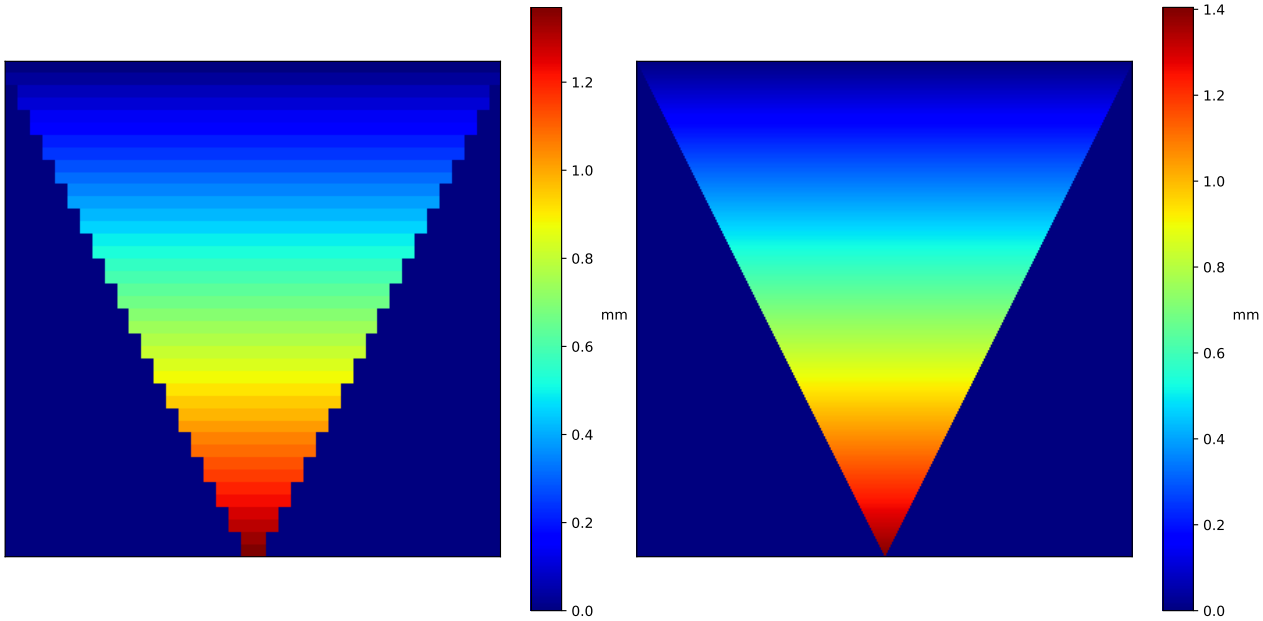

Figura 7: Deformaciones para 40 y 5000 elementos

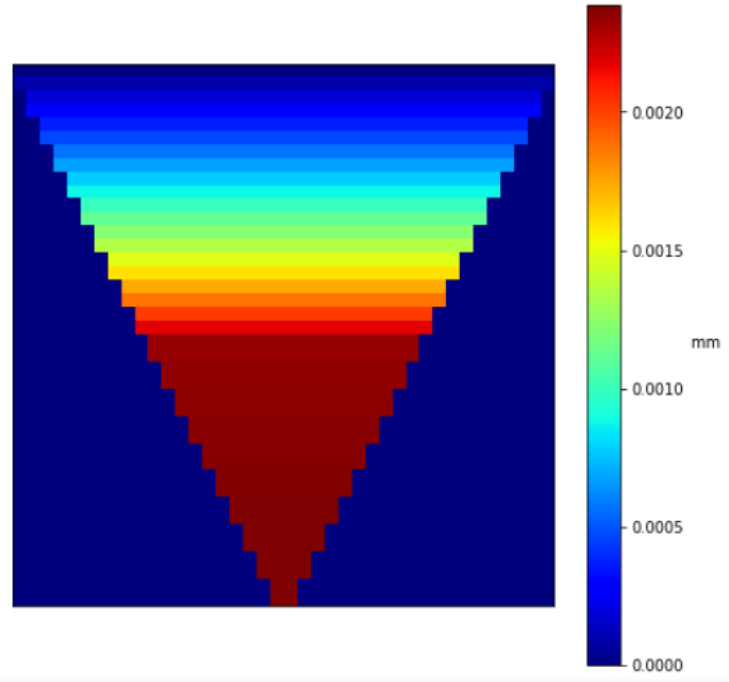

Figura 8: Deformaciones con 40 elementos para el caso no térmico

Como tal, es de esperarse que la reacción en el apoyo sea la misma que para el caso no térmico:
La fuerza de reacción en el apoyo corresponde a la fuerza en el nodo 0: $\boxed{\text{-57060.788 N}}$

4 Problema generalizado para n nodos

Al incrementar la cantidad de elementos y nodos, podemos observar que la reacción en el nodo 0 aumenta en magnitud; esto puede ser explicado considerando que la fuerza térmica añadida no depende de la longitud del elemento, y esta fuerza se aplicará sobre todos los elementos, por tanto, al incrementar la cantidad de elementos esta fuerza aparecerá mas veces haciendo que la reacción sea más grande. No obstante; estos valores convergen a un valor exacto teórico:

$$F_0 = P + EA_0\alpha\Delta T + \sum_{i\in\text{Nodes}} W_i = P + W + E(b \times e)\alpha\Delta T = 22497060.788\text{N} \tag{1}$$

Donde, la componente térmica es igual a 22440000 N; si restamos ambos resultados obtenemos el valor real de la reacción = 57060.7879 N.

5 Verificación de resultados

Para la verificación de los cálculos se utilizó el software Fusion 360.

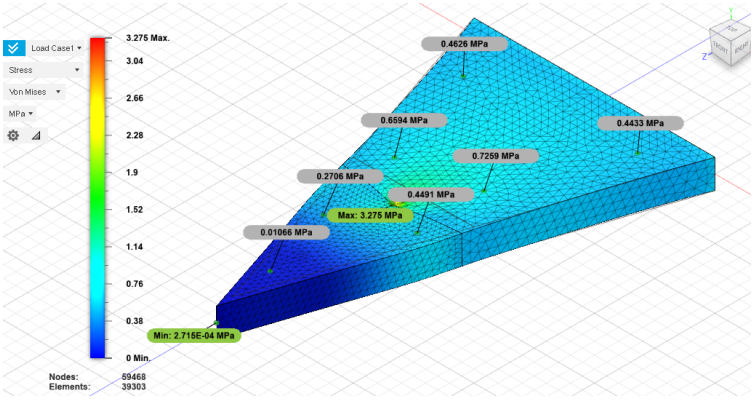

Figura 9: Esfuerzos para el caso estático

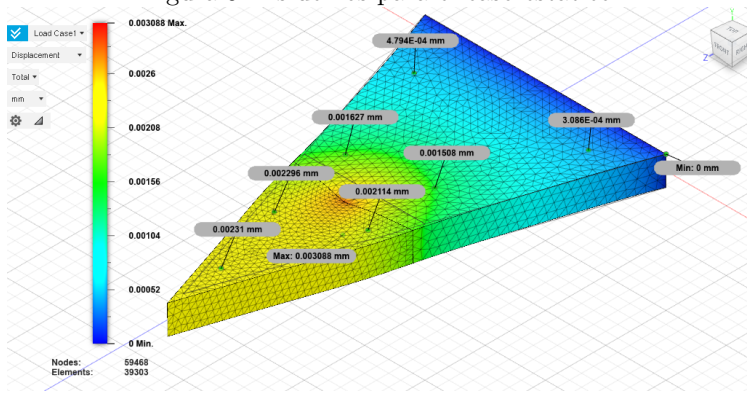

Figura 10: Deformaciones para el caso estático

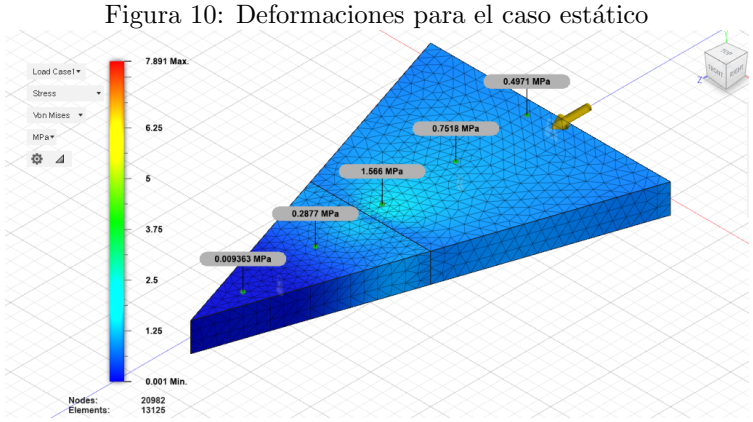

Figura 11: Esfuerzos para el caso térmico

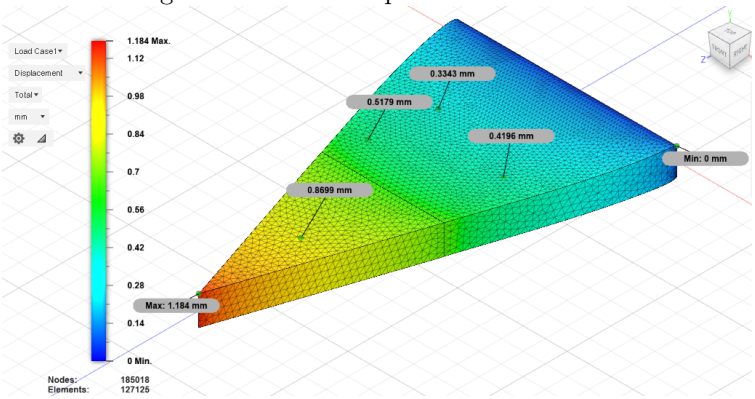

Figura 12: Deformaciones para el caso térmico

6 Conclusiones

1. Se sugiere usar un elemento de mayor dimensión para poder modelar de forma más precisa los esfuerzos y deformaciones.

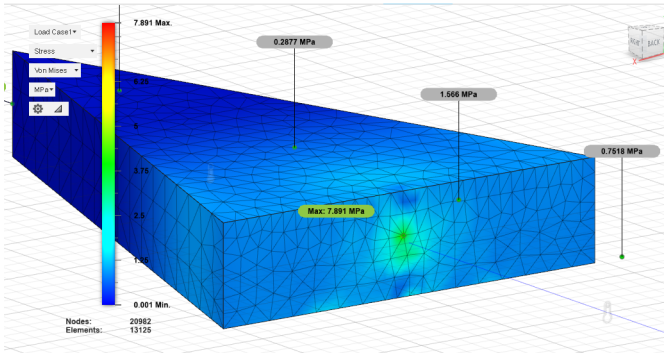

Figura 13: Esfuerzo máximo

   El esfuerzo máximo es distinto al calculado debido a que la fuerza se aplica de forma puntual y no de forma distribuida sobre un área; tan bien, sabemos que la reacción en el apoyo no se distribuye de forma regular sobre el área, sino que hay un mayor esfuerzo en las esquinas.

2. La deformación total es mayormente debida a la deformación térmica y no debido a las cargas mecánicas y por tanto la deformación tiene una tendencia lineal.
3. Se logra verificar que efectivamente, la presencia de una diferencia de temperatura no genera esfuerzos de origen mecánico al no encontrar un tope en su libre desplazamiento.
4. Es posible despreciar el peso de la placa porque es muy pequeña la fuerza en comparación con la carga central.
5. La cantidad de elementos para la convergencia de la solución es de aproximadamente 50.
6. El modelo de colocar el peso en un nodo compartido por elementos distintos nos ayuda a mejorar el tiempo de ejecución y con la misma precisión.
7. El tiempo esperado por solución es de a lo mucho 3 segundos para 10000 elementos en C++.
8. La implementación del código en MATLAB es más simple y corta pero demasiado lenta; tardando varios minutos para ejecutar 1000 elementos.
9. Debido a limitaciones de memoria no es posible usar más elementos, siendo el límite de 20000 elementos; usando 3.2 GB de RAM.

Lenguaje/Cantidad de elementos	Tiempo de ejecución (s)
C++/5000	1.2
Python/5000	4.34
MATLAB/5000	1954.7

Referencias

[1] Optimized methods in FEM:
https://www.sciencedirect.com/topics/engineering/gauss-seidel-method

[2] Strassen Algorithm:
https://www.sciencedirect.com/science/article/pii/0898122195002162

[3] Sparse Matrix:
https://en.wikipedia.org/wiki/Sparse_matrix

[4] Sparse Matrix Library:
https://github.com/uestla/Sparse-Matrix

[5] Mailman algorithm:
http://www.cs.yale.edu/homes/el327/papers/matrixVectorApp.pdf

[6] Fast Algorithms with Preprocessing for Matrix-Vector Multiplication Problems:
https://www.sciencedirect.com/science/article/pii/S0885064X84710211