



COMPLX

Advanced Engineering for Real Solutions



COMPLX

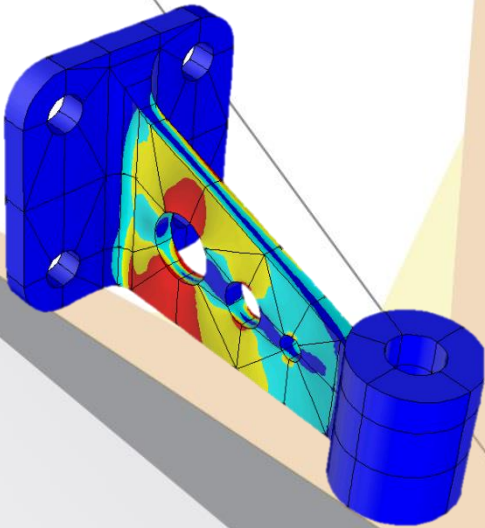
CURSO BÁSICO DE ELEMENTOS FINITOS

1.4 PROGRAMACIÓN DEL MEF EN 1D



MÓDULO MEF 1

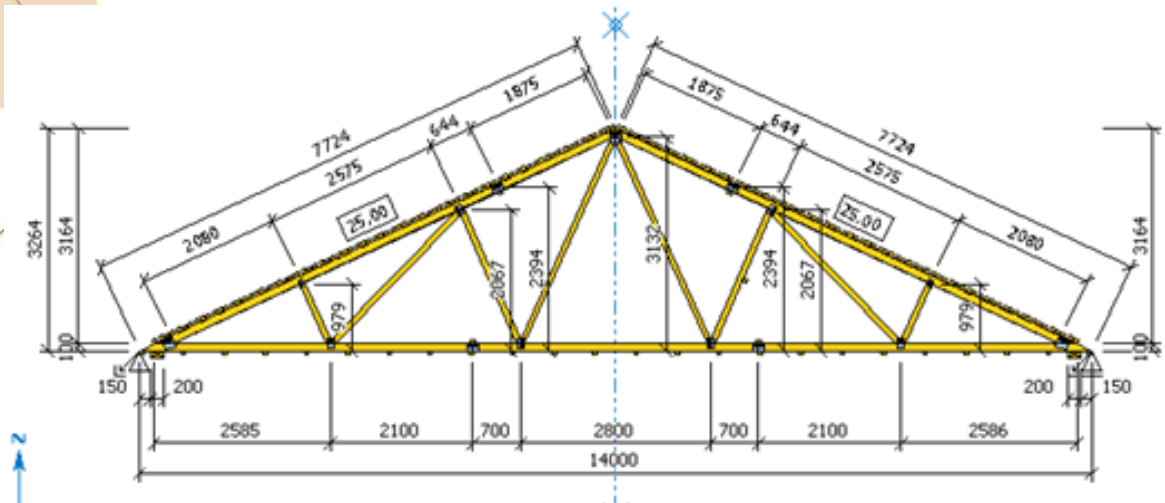
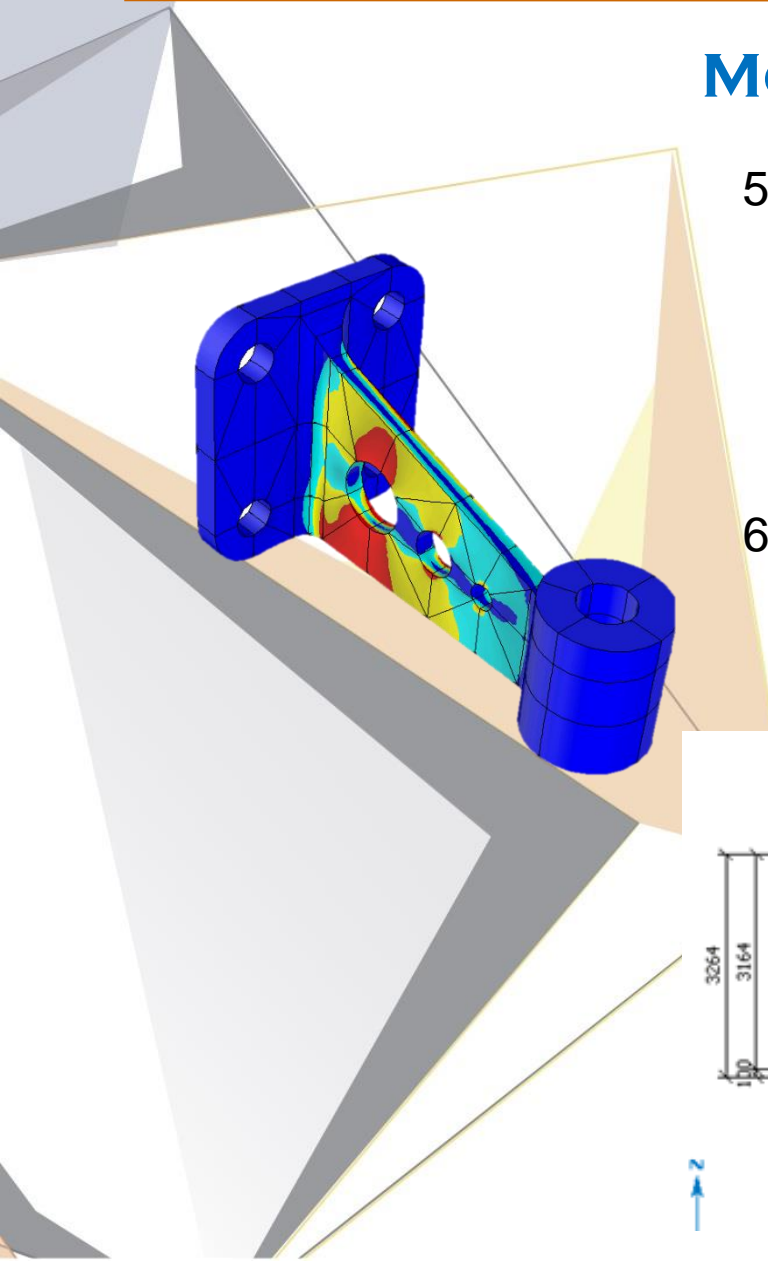
1. Introducción al Método de Elementos Finitos (MEF)
 - a) Definición
 - b) Historia
2. Conceptos básicos de álgebra lineal
 - a) Sistemas de ecuaciones simultáneas y matrices
 - b) Tipos especiales de matrices
 - c) Operaciones con matrices
 - d) Introducción a Scilab
3. Matriz de rigidez del elemento barra (1D)
 - a) Definición del elemento barra
 - b) Derivación de la matriz de rigidez
 - c) Ensamble de la matriz de rigidez global
 - d) Condiciones de Frontera homogéneas
 - e) Solución
4. Programación del MEF en 1D





MÓDULO MEF 1

5. Matriz de rigidez del elemento barra (2D)
 - a) Definición del elemento barra
 - b) Derivación de la matriz de rigidez
 - c) Ensamble de la matriz de rigidez global
 - d) Condiciones de Frontera homogéneas
 - e) Solución
6. Análisis de estructuras reticulares utilizando barras bidimensionales





FUNCIONES EN SCILAB

Gathering various steps into a reusable function is one of the most common tasks of a Scilab developer. The most simple calling sequence of a function is the following:

outvar = myfunction (invar)

where the following list presents the various variables used in the syntax:

- *myfunction* is the name of the function,
- *invar* is the name of the input arguments,
- *outvar* is the name of the output arguments.

The values of the input arguments are not modified by the function, while the values of the output arguments are actually modified by the function.

The complete syntax for a function which has a fixed number of arguments is the following:

[o1 , ... , on] = myfunction (i1 , ... , in)



DEFINICIÓN DE FUNCIONES

To define a new function, we use the *function* and *endfunction* Scilab keywords.

In the following example, we define the function *myfunction*, which takes the input argument *x*, multiplies it by 2, and returns the value in the output argument *y*.

```
function y = myfunction ( x )  
y = 2 * x  
endfunction
```

The statement *function y = myfunction (x)* is the header of the function, while the body of the function is made of the statement *y = 2 * x*. The body of a function may contain one, two or more statements.



EJEMPLO: DEFINICIÓN DE FUNCIONES

```
function y = myfunction ( x )  
y = 2 * x  
endfunction
```

Copie la función anterior en una nueva sesión del editor de Scilab, guardela en su directorio de trabajo y ejecútela con el botón **Execute**

Pruebe la función introduciendo cualquier número como argumento de entrada. Repita el procedimiento con:

```
function y = myfunction2 ( x )  
z = 2 * x  
endfunction
```

The screenshot shows the Scilab 5.5.2 Console window. The console output is as follows:

```
Scilab 5.5.2 Console  
File Edit Control Applications ?  
Scilab 5.5.2 Console  
--> |  
function y = myfunction ( x )  
y = 2 * x  
endfunction  
coord = [0;  
3;  
1;  
2];  
ndim(coord)  
size(coord)  
a=size(1)  
a=size(coord)  
a=size(coord,1)
```

The console output shows the function definition and the result of the execution. The function definition is: `function y = myfunction (x)`, `y = 2 * x`, and `endfunction`. The execution result is: `coord = [0;`, `3;`, `1;`, `2];`, `ndim(coord)`, `size(coord)`, `a=size(1)`, `a=size(coord)`, and `a=size(coord,1)`.

Copie la función anterior en una nueva sesión del editor de Scilab, guardela en su directorio de trabajo y ejecútela con el botón Execute

Utilice la función introduciendo cualquier variable

Repita el procedimiento con:



MATRICES DE RIGIDEZ ELEMENTALES

De acuerdo a la definición de la matriz de rigidez,

Nuestra función podría ser de la siguiente forma:

```
function kbarra = kBar ( E,A,coord )  
L=coord(2)-coord(1)  
kbarra = (E*A/L)*[1,-1;-1,1]  
endfunction
```

$$k = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

The screenshot shows the Scilab 5.5.2 environment. The script editor displays the following code:

```
kBar1D.sci  
1 function kbarra = kBar (.E,A, coord .)  
2 L=coord(2)-coord(1)  
3 kbarra = (E*A/L)*[1,-1;-1,1]  
4 endfunction  
5
```

The Scilab 5.5.2 Console window shows the execution of the function:

```
-->exec ('C:\Users\roger\Dropbox\COMPLX\FE  
-->kBar (1000,20, [0, 3])  
ans =  
    6666.6667  - 6666.6667  
 - 6666.6667   6666.6667  
-->
```

The Variable Browser window shows the following variables:

Name	Value	Type	Visi...
ans	[6.6...	Double	local
kbarra	[1, -...	Double	local
L	1	Double	local
coord	[0, 1]	Double	local
A	1	Double	local
E	1	Double	local

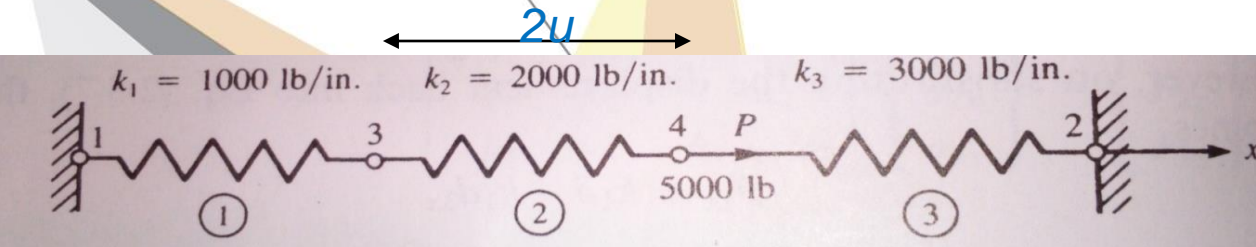
The Command History window shows the following commands:

```
-coord = [0; //N  
          3; //Nodc  
          1; //Nodc  
          2]; //Nodc  
-ndim(coord)  
-size(coord)  
-a=size(1)  
-size(coord)
```



VARIOS ELEMENTOS

Necesitamos agregar un argumento de entrada adicional (elem) a fin de permitir a la función ser utilizada con varios pares de coordenadas, referidas a un elemento por fila.



```
function kbarra = kBar ( E,A,coord,elem )
L=coord(elem,2)-coord(elem,1)
kbarra = (E*A/L)*[1,-1;-1,1]
endfunction
```

```
-->coord=[0, 1;1, 3; 3,4];
```

```
-->kBar(1000,1,coord,2)
```

ans =

$$\begin{bmatrix} 500. & -500. \\ -500. & 500. \end{bmatrix}$$

```
-->kBar(1000,1,coord,1)
```

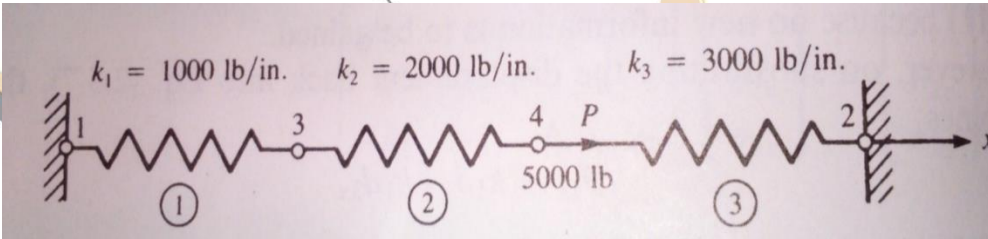
ans =

$$\begin{bmatrix} 1000. & -1000. \\ -1000. & 1000. \end{bmatrix}$$



NODOS VS. POSICIÓN

Es más conveniente separar los números de nodo de su localización, a fin de referirnos directamente al nodo n en lugar del nodo “que está en la posición x, y, z ”.



$$L = ndinfo(elinfo(elem,2),1) - ndinfo(elinfo(elem,1),1)$$

```
-->elem=1;L=ndinfo(elinfo(elem,2),1)-ndinfo(elinfo(elem,1),1)
```

L =
1.

```
-->elem=2;L=ndinfo(elinfo(elem,2),1)-ndinfo(elinfo(elem,1),1)
```

L =
2.

```
-->elem=3;L=ndinfo(elinfo(elem,2),1)-ndinfo(elinfo(elem,1),1)
```

L =
3.

```
-->ndinfo=[0,0,0,0;6,0,0,0;1,0,0,1;3,0,0,1]
```

```
ndinfo =  
0. 0. 0. 0.  
6. 0. 0. 0.  
1. 0. 0. 1.  
3. 0. 0. 1.
```

```
-->elinfo=[1,3,1000,1;3,4,2000,1;4,2,3000,1]
```

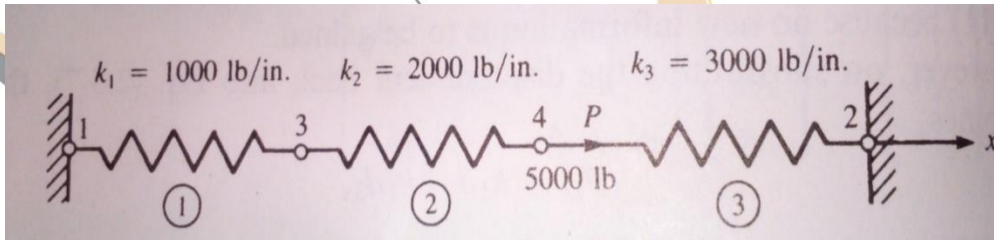
```
elinfo =  
1. 3. 1000. 1.  
3. 4. 2000. 1.  
4. 2. 3000. 1.
```

Las coordenadas utilizadas en ndinfo permiten validar el cálculo de L en elementos que miden respectivamente 1, 2 y 3 unidades.



NUEVA FUNCIÓN KBAR

Cada elemento puede tener L, E y A distinta. Combina la información de conectividad de cada elemento con la posición de cada nodo al cual está conectado



```
function kbarra = kBar (
    ndinfo,elinfo,elem )
L=ndinfo(elinfo(elem,2),1)-
ndinfo(elinfo(elem,1),1)
kbarra =
    (elinfo(elem,3)*elinfo(elem,4)/L)*
    [1,-1;-1,1]
endfunction
```

```
-->ndinfo=[0,0,0,0;3,0,0,0;1,0,0,1;2,0,0,1]
ndinfo =
```

0.	0.	0.	0.
3.	0.	0.	0.
1.	0.	0.	1.
2.	0.	0.	1.

```
-->elinfo=[1,3,1000,1;3,4,2000,1;4,2,3000,1]
elinfo =
```

1.	3.	1000.	1.
3.	4.	2000.	1.
4.	2.	3000.	1.

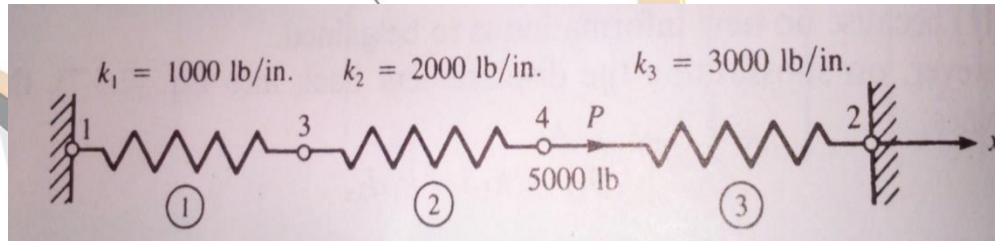
```
-->kBar(ndinfo,elinfo,2)
ans =
```

2000.	- 2000.
- 2000.	2000.



ENSAMBLE KBAR

Para obtener la matriz de rigidez requerimos una matriz nula con las dimensiones adecuadas, la matriz de cada elemento y una nueva función.



-->kelem
kelem =

$$\begin{bmatrix} 3000. & -3000. \\ -3000. & 3000. \end{bmatrix}$$

-->KEstruc
KEstruc =

$$\begin{bmatrix} 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \end{bmatrix}$$

-->KEstruc=KGBar(kelem,dimen,conec,KEstruc)
KEstruc =

$$\begin{bmatrix} 0. & 0. & 0. & 0. \\ 0. & 3000. & 0. & -3000. \\ 0. & 0. & 0. & 0. \\ 0. & -3000. & 0. & 3000. \end{bmatrix}$$

```
function kglobal = KGBar ( kbar, dimen,
conec, kglobal)
for reng=1:2*dimen;
EG=conec(reng);
for colum=1:2*dimen;
CG=conec(colum);
kglobal(EG,CG)=kglobal(EG,CG)+kbar(reng,
colum);
end
end
endfunction
```



EJEMPLO

Con las funciones descritas anteriormente, solo hace falta otro ciclo que vaya calculando las matrices elementales y vaciándolas en la matriz global

conec =

3. 4.

KEstruc =

$$\begin{bmatrix} 1000. & 0. & -1000. & 0. \\ 0. & 0. & 0. & 0. \\ -1000. & 0. & 3000. & -2000. \\ 0. & 0. & -2000. & 2000. \end{bmatrix}$$

kelem =

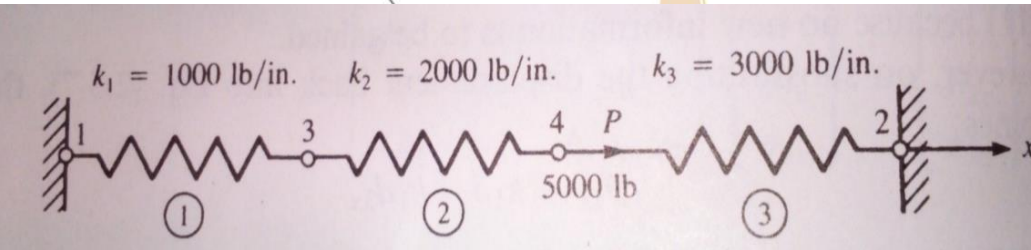
$$\begin{bmatrix} 3000. & -3000. \\ -3000. & 3000. \end{bmatrix}$$

conec =

4. 2.

KEstruc =

$$\begin{bmatrix} 1000. & 0. & -1000. & 0. \\ 0. & 3000. & 0. & -3000. \\ -1000. & 0. & 3000. & -2000. \\ 0. & -3000. & -2000. & 5000. \end{bmatrix}$$



```
KEstruc = zeros(nGDLTot,nGDLTot);
```

```
for elem = 1:elementos
```

```
kelem=kBar(ndinfo,elinfo,elem)
```

```
conec=elinfo(elem,1:2)
```

```
KEstruc=KGBar(kelem,dimen,conec,KEstruc)
```

```
end
```



CARACTERÍSTICAS DE K

Aprovechemos para repasar las características de la matriz de rigidez global:

- Simétrica
- Diagonal principal positiva
- Singular (no tiene inversa)

```
-->KEstruc=[1000.,0.,- 1000.,0.; 0.,3000.,0.,-  
3000.;- 1000.,0.,3000.,- 2000.; 0.,- 3000.,-  
2000.,5000]
```

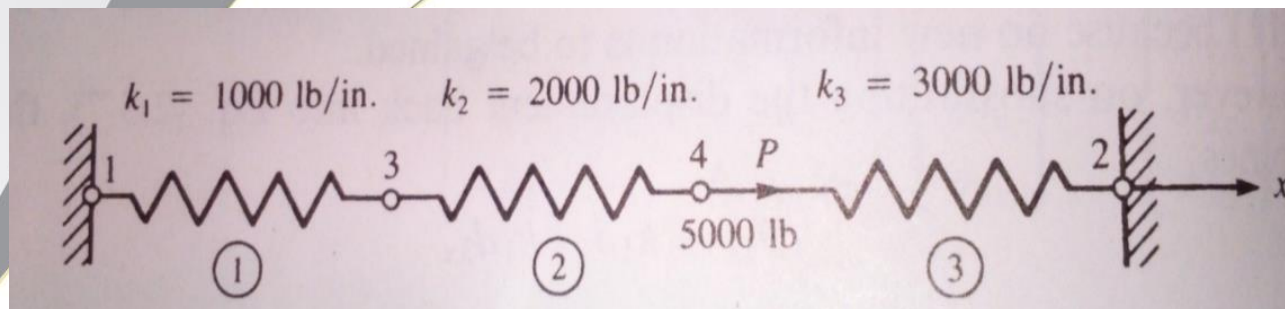
KEstruc =

```
1000.  0.  -1000.  0.  
0.    3000.  0.  -3000.  
-1000.  0.  3000.  -2000.  
0.    -3000.  -2000.  5000.
```

```
-->inv(KEstruc)
```

```
!--error 19
```

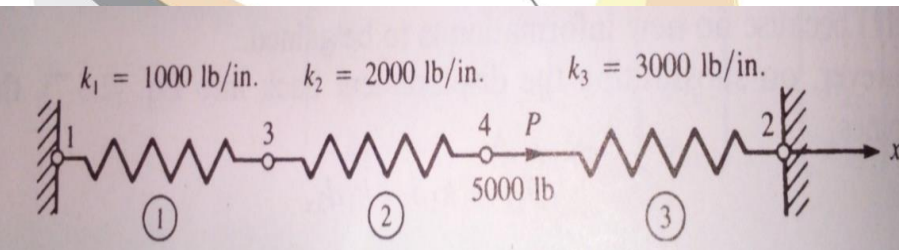
```
Problem is singular.
```





MATRIZ DE RIGIDEZ REDUCIDA

Con las funciones descritas anteriormente, solo hace falta otro ciclo que vaya calculando las matrices elementales y vaciándolas en la matriz global



KEstruc =

$$\begin{bmatrix} 1000. & 0. & -1000. & 0. \\ 0. & 3000. & 0. & -3000. \\ -1000. & 0. & 3000. & -2000. \\ 0. & -3000. & -2000. & 5000. \end{bmatrix}$$

```
freenodes=sum(ndinfo(:,4)) // Nodos
libres
```

```
nGDL=0
```

```
for nodo = 1:nodos
```

```
    if ndinfo(nodo,4)==1
```

```
        nGDL=nGDL+1;
```

```
        Matred(nGDL) = nodo;
```

```
    end
```

```
end
```

```
FExt=zeros(freenodes,1);
```

```
for gdlreng=1:freenodes
```

```
    FExt(gdlreng)=Fext(Matred(gdlreng))
```

```
    for gdlcolumn=1:freenodes
```

```
        KReduci(gdlreng,gdlcolumn) =
```

```
        KEstruc(Matred(gdlreng),Matred(gdlcolumn));
```

```
    end
```

```
end
```



SOLUCIÓN DEL EJEMPLO

Los comandos descritos desde el slide 12 fueron agrupados en una función llamada Solucionador, la cual lee la malla FEM así como las condiciones de frontera, fuerzas externas y dimensiones del problema.

```

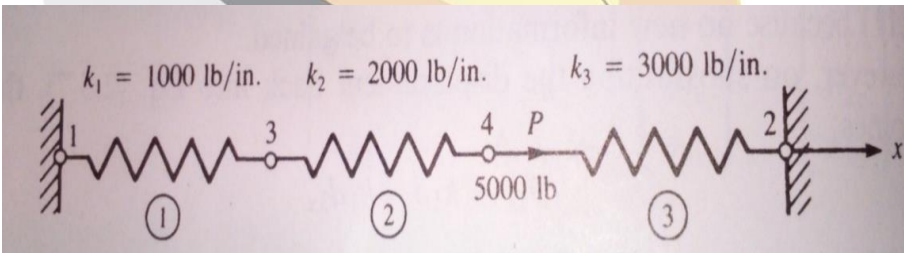
-->ndinfo=[0,0,0,0;3,0,0,0;1,0,0,1;2,0,0,1];
-->elinfo=[1,3,1000,1;3,4,2000,1;4,2,3000,1];
-->Fext=[0,0,0,5000];
-->dimen=1;
>[U,Kred,KEstruc,F]=Solucionador(ndinfo,elinfo,Fext,
dimen)
F =
    0.
    5000.

KEstruc =
    1000.    0.   -1000.    0.
    0.    3000.    0.   -3000.
   -1000.    0.    3000.   -2000.
    0.   -3000.   -2000.    5000.

Kred =
    3000.   -2000.
   -2000.    5000.

U =
    0.9090909
    1.3636364

```

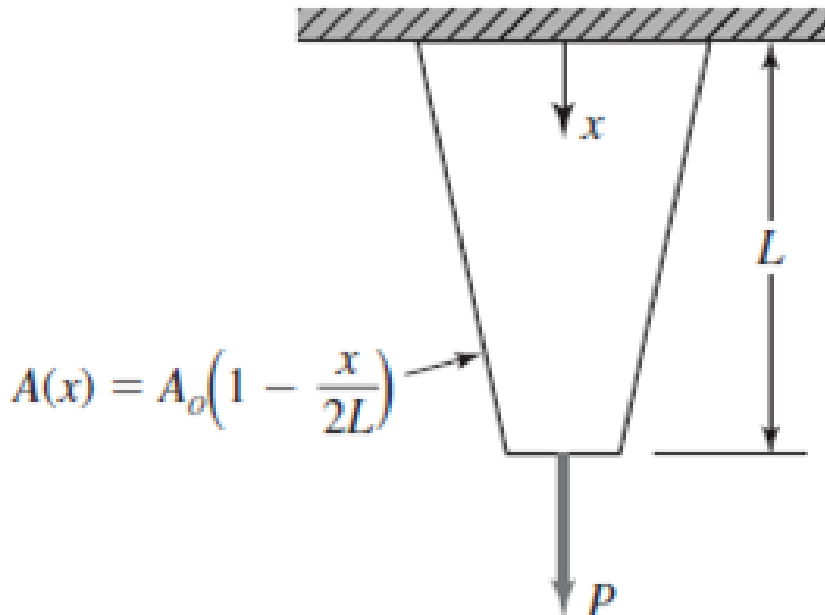




EJERCICIO 3:

En la figura se muestra una barra elástica con sección transversal variable en la cual actúa una fuerza $P = 100 \text{ N}$ en un extremo, mientras que en el otro se encuentra fija. La longitud total de la barra $L = 20 \text{ mm}$. El área transversal varía linealmente de 50 a 25 mm^2 . La fuerza $P = 10,000 \text{ N}$.

Calcule numéricamente el desplazamiento al final de la barra utilizando 1, 2, 3 y 4 elementos. Genere una gráfica de convergencia comparando los resultados del programa FEM 1D contra el resultado analítico.



$$U = 1.386 \frac{PL}{A_0 E}$$



REFERENCIAS

Zienkiewicz, O., & Taylor, R. (2004). *El método de los elementos finitos*. Barcelona: CIMNE.

Carnegie Mellon Curriculum: introduction to CAD and CAE. (2014). Obtenido de Autodesk University: <http://auworkshop.autodesk.com/library/carnegie-mellon-curriculum-introduction-cad-and-cae?language=en>

Introduction to Finite Element Methods (ASEN 5007). (22 de Diciembre de 2014). Recuperado el 17 de Junio de 2015, de Department of Aerospace Engineering Sciences University of Colorado at Boulder: <http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/>

Gallegos, S., (2006). *Notas del Curso de Elementos Finitos*. Monterrey. ITESM.