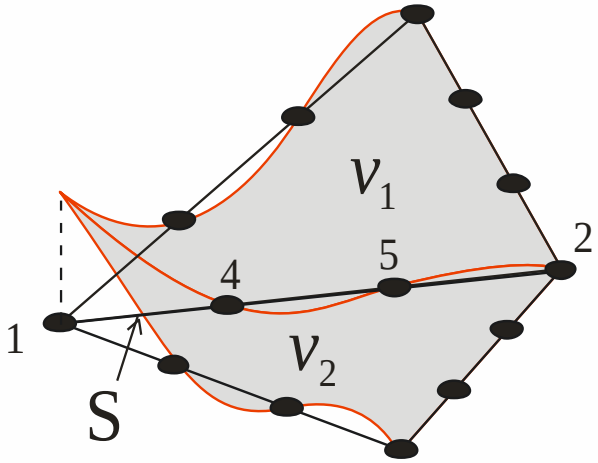


Introducción al Método de los Elementos Finitos

Parte 2.1

Programa MatFEM

Alberto Cardona

Cimec-Intec (UNL/Conicet), Santa Fe, Argentina

11-Sept-2011

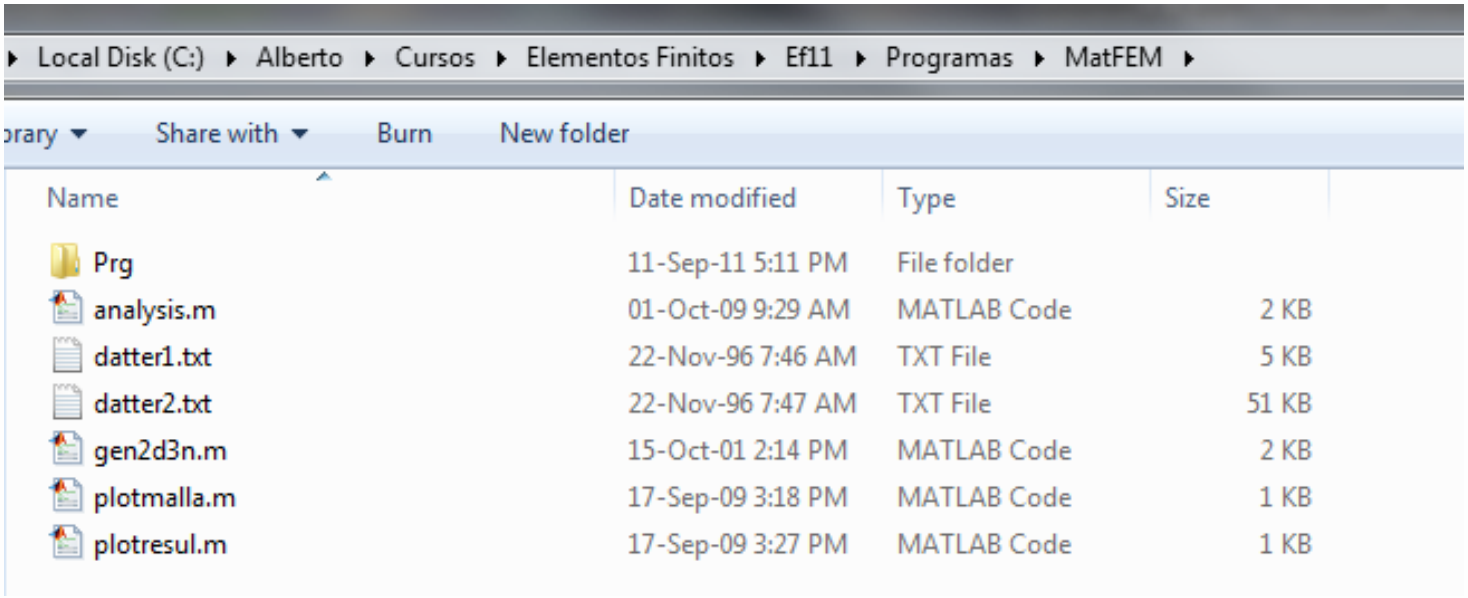
Programa de elementos finitos MatFEM

- En los ejemplos de programas vistos podemos diferenciar cuatro partes:

  1. Entrada datos
  2. Lazo sobre elementos
     - Calculo matriz rigidez para cada elemento
     - Calculo vector termino derecho para cada elemento
     - Ensamble matriz rigidez y vector carga en matriz y vector global
  3. Solución sistema de ecuaciones
  4. Postprocesamiento

- Veremos un programa general que permita reutilizar partes de forma independiente del ejemplo
- Servirá como ejemplo de la estructura típica de un programa

Carpeta principal



analysis.m : script principal

gen2d3n.m : genera datos 2d con triángulos

plotmalla.m : muestra malla de EF

plotresul.m : muestra resultados

datter1.txt, datter2.txt : ejemplos de archivos de datos

Prg: carpeta donde se ubican las funciones auxiliares

Uso del programa

Nos ubicamos en la carpeta principal. Ingresando el comando “help” seguido del nombre del programa obtenemos una ayuda para cada uno de ellos.

```
>> help analysis
   Archivo script : secuencia de llamados para resolucion
                    de un problema de elementos finitos
    Datos:
       file: nombre del archivo de datos a leer (ej. 'toto.txt')

   Lista de variables:
    in:       lista de numeros de nodos
   xx:        tabla de coordenadas
   iel:       lista de numeros de elementos
   conec: tabla de conectividades nodales
   locel:   tabla de vectores de localizacion
   fixa:     lista de grados de libertad fijos
   vfix:     valores de las fijaciones
   f:         vector de cargas nodales
   [row,col,sk]: lista de contribuciones a la matriz de rigidez
                 organizada en forma sparse (fila,columna,termino)
   u:         vector solucion
```

```
>> help gen2d3n
   function [in,xx,iel,conec]=gen2d3n(lx,numx,ly,numy,file)
   genera una malla rectangular 2d de triangulos

   lx:    Longitud rectangulo en X
   numx:  Numero de elementos sobre lado X
   ly:    Longitud rectangulo en Y
   numy:  Numero de elementos sobre lado Y
   file:  Nombre del archivo a generar
```

```
>> help plotmalla
   Llamada a la visualizacion de la malla
```

```
>> help plotresul
   Llamada a la visualizacion de resultados
```

El script principal nos muestra una lista de las principales variables del programa.

Ejemplo de uso

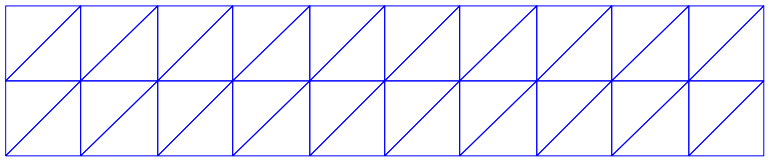
Pasos a seguir:

1. Definir una variable “file” con el nombre del juego de datos a usar.
2. Llamar a “analysis”

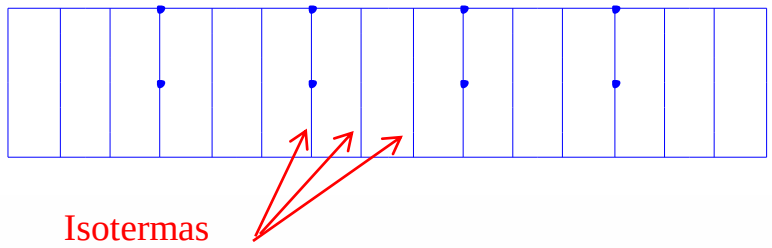
```
>> file = 'datter1.txt‘
file =
datter1.txt

>> analysis
```

3. Llamando a “plotmalla” vemos la malla de elementos finitos. Mediante “plotresul” vemos los resultados..

```
>> plotmalla
```

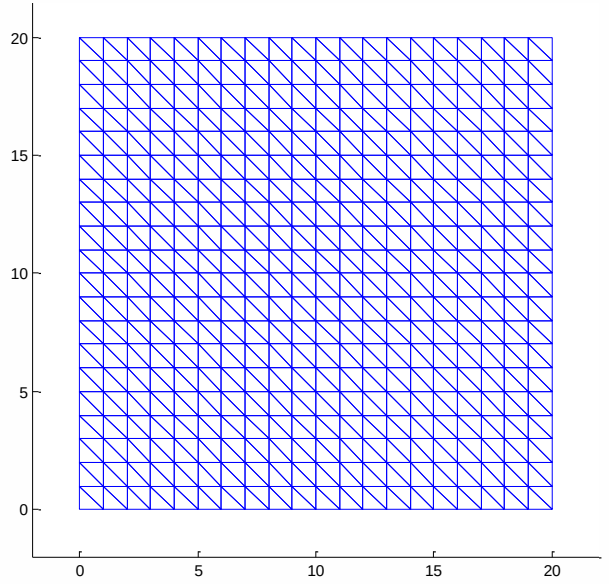


```
>> plotresul
```

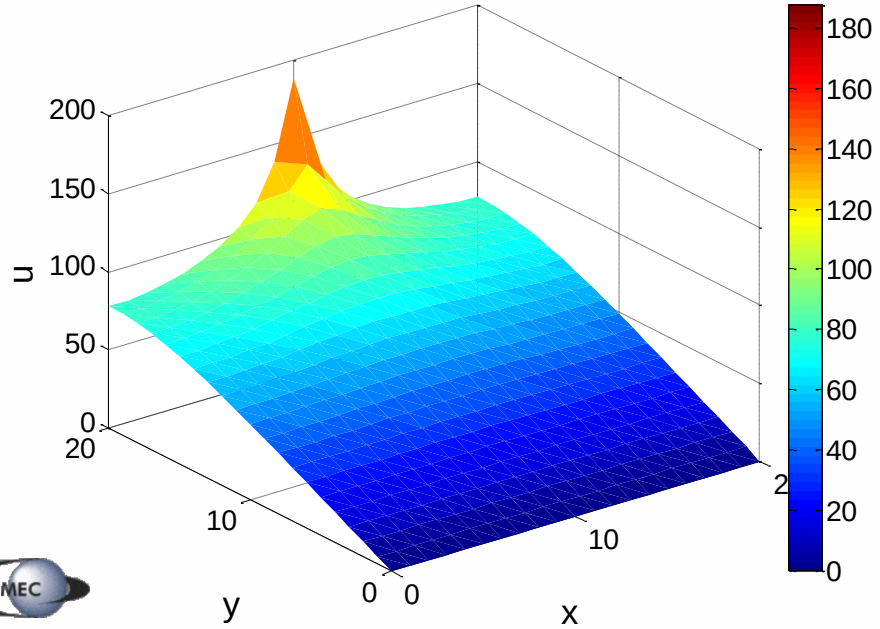


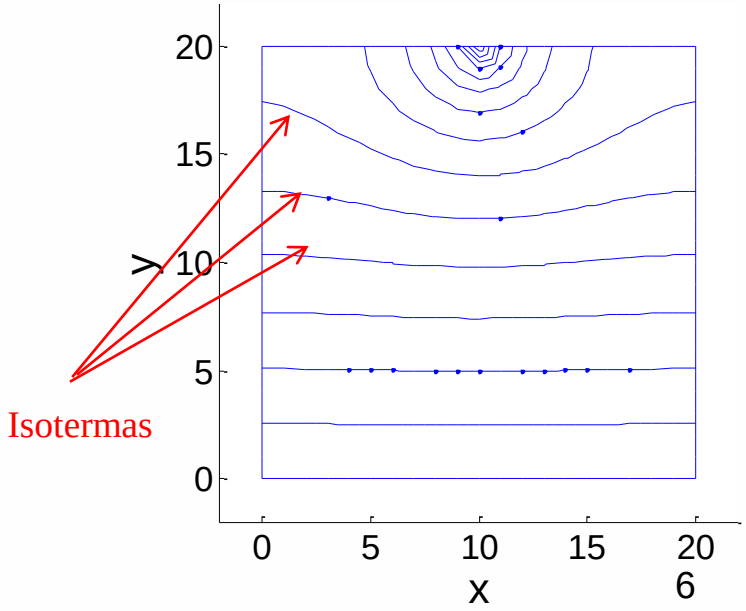
Ejemplo de uso (2)

```
>> file = 'datter2.txt‘
file =
datter2.txt
>> analysis
>> plotmalla
```




```
>> plotresul
```





Sintaxis juego de datos

nnodos

```
nnodos
    i    x  y  z
    i    x  y  z
  .....
    i    x  y  z
```

Definición nodos y coordenadas de cada nodo

Tipo de elemento (triang lineal ec Poisson, triang cuadratico ec. Poisson, cuadrangulo bilineal elasticidad, etc).

nelem

```
nelem  itype
    i  n1  n2  n3 …
    i  n1  n2  n3 …
  …
    i  n1  n2  n3 …
```

Definición elementos y conectividades de cada elemento

nfix

```
nfix
    i  cmp  val
    i  cmp  val
  …
    i  cmp  val
```

Definición nodos y componente fijos, con valor de fijacion

nloa

```
nloa
    i  cmp  val
    i  cmp  val
  …
    i  cmp  val
```

Definición nodos y componente con carga , con valor de carga

Ejemplo “datter1.txt”

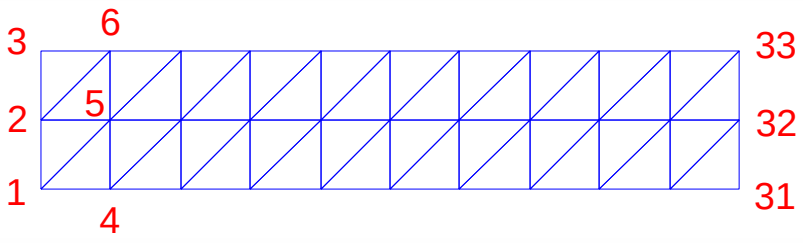
```
33
 1  0.0000000e+000  0.0000000e+000  0.0000000e+000
 2  0.0000000e+000  3.0000000e+000  0.0000000e+000
 3  0.0000000e+000  6.0000000e+000  0.0000000e+000
 4  3.0000000e+000  0.0000000e+000  0.0000000e+000
 5  3.0000000e+000  3.0000000e+000  0.0000000e+000
 6  3.0000000e+000  6.0000000e+000  0.0000000e+000
 7  6.0000000e+000  0.0000000e+000  0.0000000e+000
 8  6.0000000e+000  3.0000000e+000  0.0000000e+000
 9  6.0000000e+000  6.0000000e+000  0.0000000e+000
10  9.0000000e+000  0.0000000e+000  0.0000000e+000
11  9.0000000e+000  3.0000000e+000  0.0000000e+000
12  9.0000000e+000  6.0000000e+000  0.0000000e+000
13  1.2000000e+001  0.0000000e+000  0.0000000e+000
14  1.2000000e+001  3.0000000e+000  0.0000000e+000
15  1.2000000e+001  6.0000000e+000  0.0000000e+000
16  1.5000000e+001  0.0000000e+000  0.0000000e+000
17  1.5000000e+001  3.0000000e+000  0.0000000e+000
18  1.5000000e+001  6.0000000e+000  0.0000000e+000
19  1.8000000e+001  0.0000000e+000  0.0000000e+000
20  1.8000000e+001  3.0000000e+000  0.0000000e+000
21  1.8000000e+001  6.0000000e+000  0.0000000e+000
22  2.1000000e+001  0.0000000e+000  0.0000000e+000
23  2.1000000e+001  3.0000000e+000  0.0000000e+000
24  2.1000000e+001  6.0000000e+000  0.0000000e+000
25  2.4000000e+001  0.0000000e+000  0.0000000e+000
26  2.4000000e+001  3.0000000e+000  0.0000000e+000
27  2.4000000e+001  6.0000000e+000  0.0000000e+000
28  2.7000000e+001  0.0000000e+000  0.0000000e+000
29  2.7000000e+001  3.0000000e+000  0.0000000e+000
30  2.7000000e+001  6.0000000e+000  0.0000000e+000
31  3.0000000e+001  0.0000000e+000  0.0000000e+000
32  3.0000000e+001  3.0000000e+000  0.0000000e+000
33  3.0000000e+001  6.0000000e+000  0.0000000e+000
```

```
40  1
 1  1  4  5
 2  1  5  2
 3  2  5  6
 4  2  6  3
 5  4  7  8
 6  4  8  5
 7  5  8  9
 8  5  9  6
 9  7 10 11
10  7 11  8
11  8 11 12
12  8 12  9
13 10 13 14
14 10 14 11
15 11 14 15
16 11 15 12
17 13 16 17
18 13 17 14
19 14 17 18
20 14 18 15
21 16 19 20
22 16 20 17
23 17 20 21
24 17 21 18
25 19 22 23
26 19 23 20
27 20 23 24
28 20 24 21
29 22 25 26
30 22 26 23
31 23 26 27
32 23 27 24
33 25 28 29
```

```
34  25  29  26
35  26  29  30
36  26  30  27
37  28  31  32
38  28  32  29
39  29  32  33
40  29  33  30
```

```
3
1  1  10.
2  1  10.
3  1  10.
```

```
3
31 1  0.5
32 1  1
33 1  0.5
```



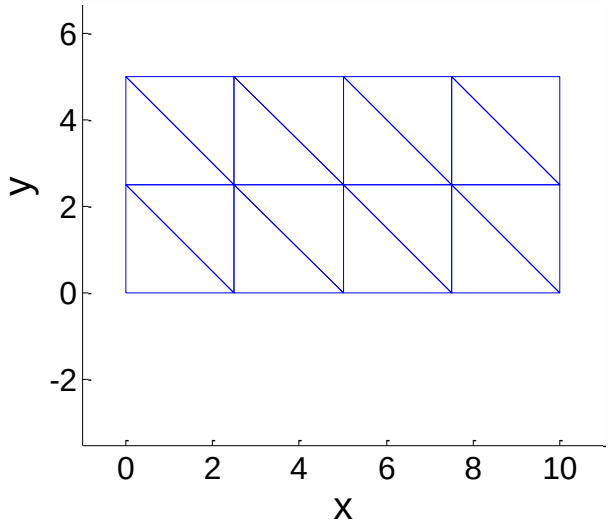
Programa auxiliar “gen2d3n.m”

```
function [in,xx,iel,conec]=gen2d3n(lx,numx,ly,numy,file)
%  function [in,xx,iel,conec]=gen2d3n(lx,numx,ly,numy,file)
%  genera una malla rectangular 2d de triangulos
%
%  lx:    Longitud rectangulo en X
%  numx:  Numero de elementos sobre lado X
%  ly:    Longitud rectangulo en Y
%  numy:  Numero de elementos sobre lado Y
%  file:  Nombre del archivo a generar
%
  DX = lx/numx;
  DY = ly/numy;
  nodo=0;
  for k=1:numy+1
    for j=1:numx+1
      nodo      = nodo+1;
      in(nodo)   = nodo;
      xx(nodo,1) = (j-1)*DX;
      xx(nodo,2) = (k-1)*DY;
      xx(nodo,3) = 0;
    end
  end

iele=0;
id1 =[1   2 numx+2];
id2 =[2 numx+3 numx+2];

for k=1:numy
   for j=1:numx
     iele=iele+1;
     iel(iele) = iele;
     conec(iele,1:3)=((k-1)*(numx+1)+j-1)+id1;
     iele=iele+1;
     iel(iele) = iele;
     conec(iele,1:3)=((k-1)*(numx+1)+j-1)+id2;
   end
end

fid = fopen(file,'wt');
fprintf(fid,'    %d \n', length(in));
fprintf(fid,'    %d    %f    %f    %f \n', [in(:) xx(:,1:3)]' );
fprintf(fid,'    %d \n', length(iel));
fprintf(fid,'    %d    %d    %d    %d \n', [iel(:) conec(:,1:3)]' );
fclose (fid);
```

Usando gen2d3n, genero la primera parte de los datos (faltan la carga y las fijaciones). Falta también el tipo de elemento !!!!

```
>> [in,xx,iel,conec] = gen2d3n(10,4,5,2,'toto.txt')

in =
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15

xx =
         0         0         0
    2.5000         0         0
……..
```

```
>> plotmalla
```



Archivo generado ‘toto.txt’

```
15
 1   0.000000    0.000000    0.000000
 2   2.500000    0.000000    0.000000
 3   5.000000    0.000000    0.000000
 4   7.500000    0.000000    0.000000
 5   10.000000   0.000000    0.000000
 6   0.000000    2.500000    0.000000
 7   2.500000    2.500000    0.000000
 8   5.000000    2.500000    0.000000
 9   7.500000    2.500000    0.000000
10   10.000000   2.500000    0.000000
11   0.000000    5.000000    0.000000
12   2.500000    5.000000    0.000000
13   5.000000    5.000000    0.000000
14   7.500000    5.000000    0.000000
15   10.000000   5.000000    0.000000
16
 1   1    2    6
 2   2    7    6
 3   2    3    7
 4   3    8    7
 5   3    4    8
 6   4    9    8
 7   4    5    9
 8   5    10   9
 9   6    7    11
10   7    12   11
11   7    8    12
12   8    13   12
13   8    9    13
14   9    14   13
15   9    10   14
16   10   15   14
```

Programa principal: “analysis.m”

```
%
%  Archivo script : secuencia de llamados para resolucion
%                   de un problema de elementos finitos
%
%  Datos:
%
%     file: nombre del archivo de datos a leer (ej. 'toto.txt')
%
%  Lista de variables:
%
%  in:    lista de numeros de nodos
%  xx:    tabla de coordenadas
%  iel:   lista de numeros de elementos
%  conec: tabla de conectividades nodales
%  locel: tabla de vectores de localizacion
%  fixa:  lista de grados de libertad fijos
%  vfix:  valores de las fijaciones
%  f:     vector de cargas nodales
%  [row,col,sk]: lista de contribuciones a la matriz de rigidez
%                organizada en forma sparse (fila,columna,termino)
%  u:     vector solucion
%
addpath 'Prg‘;
```

```
%
%  1. Lectura de datos
%
  if exist('file')
     [in,xx,iel,conec,fixa,vfix,f,locel,ndn,eltype] = input1(file);
  else
     display('   Defina el problema a correr ingresando el ');
     display('   nombre del archivo de datos en la variable "file" ');
     return
  end
%
%  2. Calculo de la matriz de rigidez
%
  if    eltype==1,    %  elemento triangulo lineal conduccion calor
    [row,col,sk] = stiffcur(in,xx,iel,conec,locel);
  end
%
%  3. Imposicion de condiciones de borde Dirichlet y solucion
%
  u = getsol(row,col,sk,fixa,vfix,f);
```

Programa auxiliar “input1.m”

```
function [in,xx,iel,conec,fixa,vfix,f,locel,ndn,eltype] = input1 (file)
%  function [in,xx,iel,conec,fixa,vfix,f,locel,ndn,eltype] = input1 (file)
%
%  Lectura, generación e impresión de datos nodales
%
%  in:     Números de nodo
%  xx:     Tabla de coordenadas
%  iel:   Números de elemento
%  conec:  Tabla de conectividades
%  fixa:   Lista de nodos fijos
%  vfix:   Valores de fijaciones
%  f:      Vector de cargas
%  locel:  Tabla de vectores de localización
%  eltype: Tipo de elemento
%  npe:    Número de nodos por elemento
%  ndn:    Número de grados de libertad por nodo

   fid = fopen(file,'rt');

%    fprintf('  Datos del análisis \n ================== \n \n')
%    fprintf('  Lista de nodos y coordenadas\n \n')

   numnp    = fscanf(fid,'%d',1);
   A = fscanf(fid,'%f',[4,numnp]);
   in = A(1,:)';
   xx = A(2:4,:)';

%    fprintf('  Lista de elementos y conectividades\n \n');

   numel  = fscanf(fid,'%f',1);
   eltype = fscanf(fid,'%f',1);
   if     (eltype==1)
     ndn = 1; npe = 3;
   elseif (eltype==2)
     ndn = 2; npe = 3;
   elseif (eltype==3)
     ndn = 2; npe = 4;
   elseif (eltype==4)
     ndn = 2; npe = 4;
   end

   A       = fscanf(fid,'%f',[npe+1,numel]);
   iel     = A(1,:)';
   conec = A(2:npe+1,:)';

   for i=1:ndn
    locel(:,i:ndn:npe*ndn) = (conec-1)*ndn + i;
   end

%    fprintf('  Lista de nodos fijos\n \n');

   nfix  = fscanf(fid,'%f',1);
   A     = fscanf(fid,'%f',[3,nfix]);
   fixa  = (A(1,:)'-1)*ndn + A(2,:)';
   vfix  =  A(3,:);
```

Programa auxiliar “input1.m” (cont)

```
%    fprintf('  Lista de cargas nodales\n \n');


    f = zeros(numnp*ndn,1);
    nload = fscanf(fid,'%f',1);
    A = fscanf(fid,'%f',[3,nload]);
    f((A(1,:)'-1)*ndn+A(2,:)') = A(3,:)';

return
```

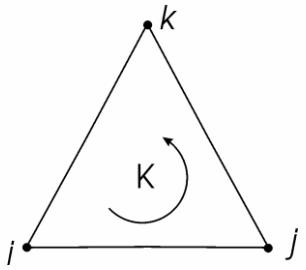
Programa auxiliar “stiffcur”

```
function [row,col,sk] = stiffcur (in,xx,iel,conec,locel)
%
%  Generacion de la matriz de rigidez
%  Elemento triangulo lineal p/problema conduccion de calor
%  in:    Numeros de nodo
%  xx:    Tabla de coordenadas
%  iel:   Numeros de elemento
%  conec: Tabla de conectividades
%  locel: Tabla de vectores de localizacion
%
nel = length (iel);

%   Genera vector inn cuya componente "i" da la posicion donde se
%   almacenan las coordenadas del nodo "i" en la tabla "xx“
inn  = zeros(max(in),1);
for i=1:length(in)
   j      = in(i);
   inn(j) = i;
end

%   El vector ind es un vector auxiliar para realizar el
%   desplazamiento  ciclico de indice en el lazo
ind  = [ 1, 2, 3, 1, 2];

%   row, col, sk dan los indices de fila, columna y contenido de
%   la matriz de rigidez en almacenamiento sparse
row  = zeros(9*nel,1);
col  = zeros(9*nel,1);
sk = zeros(9*nel,1);

in1 = 0;
for iel = 1:nel

   for k=1:3
      X1(k) = xx(inn(conec(iel,k)),1);
      Y1(k) = xx(inn(conec(iel,k)),2);
   end

   dosdelta  = (X1(2)*Y1(3) - X1(3)*Y1(2)) + ...
               (X1(3)*Y1(1) - X1(1)*Y1(3)) + ...
               (X1(1)*Y1(2) - X1(2)*Y1(1)) ;

   for i=1:3
      beta(i)  =   (Y1( ind(i+1) ) - Y1( ind(i+2) ))/dosdelta;
      gamma(i) =  -(X1( ind(i+1) ) - X1( ind(i+2) ))/dosdelta;
   end

   for i = 1:3
      for j=1:3
         in1 = in1 + 1;
         row(in1) = inn(locel(iel,i));
         col(in1)   = inn(locel(iel,j));
         sk(in1) = (beta(i) * beta(j) + gamma(i)* gamma(j) )*dosdelta/2;
      end
   end
end
```

Detalle cálculo de la matriz de rigidez elemental

$$\mathbf{A}_K \triangleq \Delta \begin{bmatrix} \left(\beta_i^2+\gamma_i^2\right) & \left(\beta_i\beta_j+\gamma_i\gamma_j\right) & \left(\beta_i\beta_k+\gamma_i\gamma_k\right) \\ & \left(\beta_j^2+\gamma_j^2\right) & \left(\beta_j\beta_k+\gamma_j\gamma_k\right) \\ \text{sim.} & & \left(\beta_k^2+\gamma_k^2\right) \end{bmatrix}$$

```
ind  = [ 1, 2, 3, 1, 2];
. . . . . .

dosdelta  = (X1(2)*Y1(3) - X1(3)*Y1(2)) + ...
            (X1(3)*Y1(1) - X1(1)*Y1(3)) + ...
            (X1(1)*Y1(2) - X1(2)*Y1(1)) ;

   for i=1:3
      beta(i)  =       (Y1( ind(i+1) ) - Y1( ind(i+2) ))/dosdelta;
      gamma(i) =  -(X1( ind(i+1) ) - X1( ind(i+2) ))/dosdelta;
   end

   for i = 1:3
      for j=1:3
         in1 = in1 + 1;
         row(in1) = inn(locel(iel,i));
         col(in1)   = inn(locel(iel,j));
         sk(in1) = (beta(i) * beta(j) + gamma(i)* gamma(j))*dosdelta/2;
      end
   end
```



$$\beta_i = \frac{y_j - y_k}{2\Delta}$$

$$\gamma_i = \frac{x_k - x_j}{2\Delta}$$

$$\Delta = \frac{1}{2}\det\begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix}$$

Programa auxiliar “getsol.m”

```
function u = getsol (row,col,sk,fixa,vfix,f)
% funcion u = getsol (S,fixa,vfix,f)
%    row,col,sk : filas, columnas, contenido de matriz de rigidez "rala"
%    fixa:  Lista de grados de libertad fijos
%    vfix:  Valores de fijaciones
%    f:     Vector de cargas
%

   S = sparse(row,col,sk);

   if size(vfix,1)==1
      vfix = vfix';
   end

   f1         = full(S(:,fixa))*vfix;
   f1         = f - f1;
   f1(fixa) = vfix;

   S(:,fixa)      = zeros(size(S,1),length(fixa));
   S(fixa,:)      = zeros(length(fixa),size(S,1));
   S(fixa,fixa) = diag(ones(length(fixa),1));

   u  =  (S\f)';

return
```