```
%*** assembly and resolution of
%    -div(grad(U)) = 0  in a disc in 2D
%                U = U0 on the right half of the disc border (Gamma_D)
%             dU/dN = 0  on the left half of the disc border (Gamma_N)
%    with
%      U0(x,y) = sin(pi/6*x).*cos(pi/2*y);
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch
% http://perso.unifr.ch/ales.janka

%*** get the mesh: a disc/circle: unstructured mesh
  [P,E,T]=initmesh('circleg');
  XY = P';  Elm = T';  Elm = Elm(:,1:3);
  nnod = size(XY,1);
  nelm = size(Elm,1);

%*** get the mesh: a disc/circle: structured mesh
%  [XY,Elm] = getcircle4(15,40,2.1);
%  nnod = size(XY,1);
%  nelm = size(Elm,1);

%*** get the mesh: a square (-1,1) x (-1,1):
%  [XY,Elm] = getdrapeau3(20,20);
%  nnod = size(XY,1);
%  nelm = size(Elm,1);


%*** identify the border nodes, get the node-connectivity matrix G:
  [G,Border,bfac] = incidence2(Elm);

%*** identify nodes with the Dirichlet condition --> list them in pBC():
  Mark = zeros(nnod,1);
  Mark(Border) = 1;
  pBC = find(Mark & XY(:,1)>=-1e-6);

%*** visualize the mesh and Dirichlet boundary nodes (red dots):
  figure(1); clf
  gplot(G,XY);   % <-- gplot() fonction Matlab de base, visu des graphes
  axis equal
  hold on
  plot(XY(pBC,1),XY(pBC,2),'ro');
  ti1 = title('mesh and Dirichlet boundary nodes');

%*** define U0, not only on Gamma_D but everywhere:
  U0 = sin(pi/6*XY(:,1)).*cos(pi/2*XY(:,2));

%*** assembly of the Poisson problem with homog.Neumann boundary cond.everywhere
:
  K = mkpoisson2D(XY,Elm);
  RHS = zeros(nnod,1);

%*** visu: sparsity pattern of the stiffness matrix:
  figure(2); clf;
  spy(K);
  title('sparsity pattern of the stiffness matrix');


%*** impose Dirichlet boundary conditions on Gamma_D:
%    instead of solving K*U = RHS, with U=U0 on Gamma_D, we solve
%       K*(U0+W) = RHS,
%    with U0 known everywhere on Omega (on Gamma_D with the right boundary value
s)
```



```
%    Then, we must have W=0 on Gamma_D, ie we solve   K*W = RHS - K*U0;
%    In order to impose W=0 on Gamma_D, we set to zero the
%    corresponding (pBC) rows and columns of the stiffness matrix K
%--- right-hand side RHS is also set to zero on pBC:
  RHS = RHS - K*U0;
  RHS(pBC) = 0;
  dI = ones(nnod,1);
  dI(pBC) = 0;
  dI = spdiags(dI,0,nnod,nnod);
%--- add ones on the diagonal of K for Dirichlet nodes:
  K = dI*K*dI + (speye(nnod)-dI);

%*** solution of the linear system: K*W = RHS:
  W = K\RHS;
  U = U0 + W;

%*** visualization the solution U by trisurf(Elm,X,Y,Z,Color):
  figure(3); clf;
  ts3=trisurf(Elm,XY(:,1),XY(:,2),U,U);
  shading interp                % <-- color interpolation
  set(ts3,'EdgeColor','k');     % <-- edge color
  set(ts3,'EdgeAlpha',0.1);     % <-- edge transparency
%  set(ts3,'FaceAlpha',0.1);    % <-- face transparency
  cb3 = colorbar;
  xl3 = xlabel('X');
  yl3 = ylabel('Y');
  zl3 = zlabel('U(x,y)');
  ti3 = title('solution U(x,y)');
```

```
function K = mkpoisson2D(XY,Elm);
% function K = mkpoisson2D(XY,Elm);
%-------------------------------------------------------------------------
%*** purpose:
%   assembles the system of equations for
%    -div(grad(U)) = f   in Omega
%   with homogeneous Neumann conditions on the bord of Omega
%   no Dirichlet boundary conditions (BCs) are imposed!
%   2D domains Omega are handled,
%   nnod == no. of nodes of the 2D mesh
%   nelm == no. of elements/triangles of the 2D mesh
%   ndof == no. of degrees of freedom per node (ndof=1 for Laplace,
%                                               ndof=3 for lin. elasticity)
%-------------------------------------------------------------------------
%*** parameters:
%  XY(1:nnod,1:2)       -in-    XY(i,1:2)=[x_i, y_i] nodal coords of node (i)
%  Elm(1:nelm,1:3)      -in-    Elm(k,1:3) = [i,j,k] vertices of element (k)
%  K(1:nnod,1:nnod)     -out-   sparse system matrix WITHOUT Dirichlet BCs
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch
% http://perso.unifr.ch/ales.janka

%*** number of nodes (nnod) and number of elements:
  nnod = size(XY,1);
  nelm = size(Elm,1);
  ndof = 1;

%*** storage space for all elementary stiffness-matrices:
  Kelm = zeros(ndof*3,ndof*3,nelm);
  t0 = clock;
%=== loop over all elements (el), el={1,...,nelm}:
  for el = 1:nelm
%*** time estimation of finish: (not important):
    if (el==ceil(nelm/100))
      t = ceil(100*etime(clock,t0));
      fprintf('*** mkpoisson2D: time to finish: %d min %d s\n',...
          floor(t/60),rem(t,60));
    end;
%*** get the elementary stiffness matrix (for element Elm(el,:)):
    EXY = XY(Elm(el,:),:);
    EK = elmpoisson2D(EXY);
%*** store the elementary stiffness-matrix in the storage space Kelm():
    Kelm(1:3*ndof,1:3*ndof,el) = EK;
  end;
%=== loop over all elements done

%*** assemble the global stiffness-matrix from the local ones in Kelm:
%    elements of local stiffness matrix (ndof*3 x ndof*3) on the element (el)
%    correspond to nodes Elm(el,1), Elm(el,2), Elm(el,3), in the order:
%                     Elm(el,1) Elm(el,2) Elm(el,3)
%     Elm(el,1):   |  EK(1,1)   EK(1,2)   EK(1,3) |
%     Elm(el,2):   |  EK(2,1)   EK(2,2)   EK(2,3) |
%     Elm(el,3):   |  EK(3,1)   EK(3,2)   EK(3,3) |
%--- Add these contribution to the global matrix K (nnod*ndof x nnod*ndof)
  ntot = ndof*nnod;

%*** allocate the global stiffness matrix:
  K = sparse([],[],[],ntot,ntot);
%--- for each local node ei={1,2,3},ej={1,2,3} assemble the elm contributions:
  for ei=1:3
    ni = Elm(:,ei);
    for ej=1:3
```



```
      nj = Elm(:,ej);
      for i=1:ndof
        for j=1:ndof
          ki = ndof*(ni-1)+i;
          kj = ndof*(nj-1)+j;
          kv = squeeze(Kelm(ndof*(ei-1)+i,ndof*(ej-1)+j,:));
          K = K + sparse(ki,kj,kv,ntot,ntot);
        end;
      end;
    end;
  end;

  clear Kelm
return;


%-------------------------------------------------------------------------
% subfunction elmpoisson2D
%-------------------------------------------------------------------------
function KE = elmpoisson2D(XY)
%*** basis functions phi_1(x), phi_2(x), phi_3(x) are supposed in the form:
%    phi_1(x) = a1 + b1*x + c1*y,  phi_1(x_i) = 1, phi_1(x_j)=phi_1(x_k)=0
%    phi_2(x) = a2 + b2*x + c2*y,  phi_2(x_j) = 1, phi_2(x_i)=phi_1(x_k)=0
%    phi_3(x) = a3 + b3*x + c3*y   phi_3(x_k) = 1, phi_3(x_i)=phi_1(x_j)=0
%--- calculate a1..a3, b1..b3, c1..c3 as the solution of the system:
%       |1 x_i y_i|    |a1  a2  a3|    | 1 0 0 |
%       |1 x_j y_j| *  |b1  b2  b3| =  | 0 1 0 |
%       |1 x_k y_k|    |c1  c2  c3|    | 0 0 1 |
%--- in fact, for the following, only {bi} and {ci} are needed:
  A = [ones(3,1), XY];
  Ainv = inv(A);
% a1 = Ainv(1,1);  a2 = Ainv(1,2);  a3 = Ainv(1,3);
  b1 = Ainv(2,1);  b2 = Ainv(2,2);  b3 = Ainv(2,3);
  c1 = Ainv(3,1);  c2 = Ainv(3,2);  c3 = Ainv(3,3);
%*** area of the triangle element (two times):
  Area2x = det(A);

%*** on each element U(x) = Phi_i(x)*U(i) + Phi_j(x)*U(j) + Phi_k(x)*U(k),
%    U(i,j,k) are the three values of U(x) on element vertices/nodes (i,j,k)
%    grad(U(x))=grad(Phi_i(x))*U(i)+grad(Phi_j(x))*U(j)+grad(Phi_k(x))*U(k)
%    where  grad(Phi_i) = | bi | , because Phi_i(x) = ai + bi*x + ci*y
%                         | ci |
%    Hence, grad(U) = | b1 b2 b3 |   | U(n1) |
%                     | c1 c2 c3 | * | U(n2) | = Grad*U|elm
%---                                 | U(n3) |
  Grad = [ b1 b2 b3
           c1 c2 c3];

%*** elementary stiffness matrix KE on element K=Elm(el,:) in tau(Omega):
%    a_K(U,V) = int_K  (Grad*V)' * (Grad*U) dx
%    but for piecewise LINEAR finite elements (P1), (Grad*U)(x) does not
%    depend on (x). Hence:
%                                              | U(i) |
%       a_K(U,V) = [V(i) V(j) V(k)] * [KE] * | U(j) |
%---                                           | U(k) |
  KE = Area2x/2 *Grad'*Grad;

return;
```

```
%*** assembly and resolution of
%    -div(grad(U)) = 0  in the given 3D domain (engrenage)
%                U = 1  on boundary faces with reference number 4
%                U = 2  on boundary faces with reference number 3
%            dU/dN = 0  on all other boundary faces
%--------------------------------------------------------------------------


%*** load 3D mesh:
% XYZ(nnod,3)   ... nodal coordinates (x,y,z)
% Elm(nelm,4)   ... element connectivity (4 indices of nodes per tetrahedron)
% Fac(nfac,3)   ... boundary faces connectivity (3 indices of nodes per face)
% FRef(nfac,1)  ... face reference number (to tell which face is where, roughly)

  load engrenage.mat

%*** mesh size:
  nnod = size(XYZ,1);
  nelm = size(Elm,1);
  nfac = size(Fac,1);

%*** visualize the 3D mesh:
  figure(1); clf
  ts1=trisurf(Fac,XYZ(:,1),XYZ(:,2),XYZ(:,3),FRef);
  axis equal;
  set(ts1,'EdgeAlpha',0.1);
  colorbar
  xlabel('x');
  ylabel('y');
  zlabel('z');
  title('boundary faces and their reference numbers');

%*** identify nodes on faces with ref. numbers 3 and 4:
  f3 = find(FRef==3);           % <-- indices of faces with ref. no. 3
  p3 = unique(Fac(f3,:));       % <-- nodal indices on those faces
  f4 = find(FRef==4);           % <-- indices of faces with ref. no. 4
  p4 = unique(Fac(f4,:));       % <-- nodal indices on those faces

%*** set of nodes with Dirichlet boundary conditions applied:
  pBC = [p3;p4];

%*** prepare particular U0 satisfying given Dirichlet boundary conditions:
  U0 = zeros(nnod,1);
  U0(p4) = 1;
  U0(p3) = 2;

%*** assembly of the Poisson problem with homog.Neumann boundary cond.everywhere
%  K = mkpoisson3D(XYZ,Elm);
  K = speye(nnod);
  RHS = zeros(nnod,1);

%*** impose Dirichlet boundary conditions on Gamma_D:
%    instead of solving K*U = RHS, with U=U0 on Gamma_D, we solve
%       K*(U0+W) = RHS,
%    with U0 known everywhere on Omega (on Gamma_D with the right boundary value
%    Then, we must have W=0 on Gamma_D, ie we solve   K*W = RHS - K*U0;
%    In order to impose W=0 on Gamma_D, we set to zero the
%    corresponding (pBC) rows and columns of the stiffness matrix K
%--- right-hand side RHS is also set to zero on pBC:
  RHS = RHS - K*U0;
  RHS(pBC) = 0;
  dI = ones(nnod,1);
```



```
  dI(pBC) = 0;
  dI = spdiags(dI,0,nnod,nnod);
%--- add ones on the diagonal of K for Dirichlet nodes:
  K = dI*K*dI + (speye(nnod)-dI);

%*** visu: sparsity pattern of the stiffness matrix:
  figure(2); clf
  spy(K);
  title('sparsity pattern of the stiffness matrix');

%*** solution of the linear system: K*W = RHS:
  W = K\RHS;
  U = U0 + W;

%*** visualize the solution:
  figure(3); clf
  ts3=trisurf(Fac,XYZ(:,1),XYZ(:,2),XYZ(:,3),U);
  axis equal
  shading interp                % <-- color interpolation
  set(ts3,'EdgeColor','k');     % <-- edge color
  set(ts3,'EdgeAlpha',0.1);     % <-- edge transparency
%  set(ts3,'FaceAlpha',0.1);    % <-- face transparency
  cb3 = colorbar;
  xl3 = xlabel('X');
  yl3 = ylabel('Y');
  zl3 = zlabel('Z');
  ti3 = title('solution U(x,y,z)');
```