```
%*** assembly and resolution of linear elasto-statics (without temperature)
%     -div(Tau) = f  in the given 3D domain (engrenage)
%             U = 0  on Dirichlet boundary (reference number 4)
%         Tau.n = g  on Neuman boundary  (traction force)
% where
%    Tau = 2*mu*Eps + lambda*trace(Eps)*Id
%    Eps(U)_{ij} = 1/2 * (dU_i/dx_j + dU_j/dx_i)
%    mu = 0.5*E/(1+nu)
%    lambda = E*nu/((1+nu)(1-2*nu))
% (mu,lambda) are the Lame coefficients,
% (E,nu) is the Young modulus and Poisson ratio
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------


%*** 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 geometry / mesh:
  load engrenage.mat
%*** turn by pi/6 in X-Z plane:
  al = pi/6;
  XYZ = XYZ*[cos(al) 0 sin(al); 0 1 0; -sin(al) 0 cos(al)];
%*** zoom 10x:
  XYZ = 10*XYZ;

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

%*** define mechanical properties (const per element):
  EYoung   = 3e+9 * ones(nelm,1);       % <-- Young modulus 3GPa (nylon)
  EPoisson = 0.3 * ones(nelm,1);        % <-- Poisson ratio 0.35
  density = 1500;
  gravity = 9.8;

%*** body force density due to gravity, in [N/m^3]:
  EForce = zeros(nelm,3);
  EForce(:,3) = -density * gravity;

%*** barycenters of faces:
  FXYZ = (XYZ(Fac(:,1),:)+XYZ(Fac(:,2),:)+XYZ(Fac(:,3),:))/3;

%*** surfacic force density traction on surface with reference no. 3, in [N/m^2]
:
  FForce = 200*((FRef==3)*[1 1 1]) .* [-FXYZ(:,3) FXYZ(:,2) FXYZ(:,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
  set(gca,'FontSize',16);
  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

%*** Dirichlet boundary conds: x+y+z displacement of nodes with ref.no.4:
  pBC = [3*p4-2;        % <-- x-displacements of nodes listed in (p4)
         3*p4-1;        % <-- y-displacements of nodes listed in (p4)
         3*p4];         % <-- z-displacements of nodes listed in (p4)

%*** assemble linear elasto-statics with homogeneous Neumann everywhere:
  [K,RHS] = mklinelast3D(XYZ,Elm,EYoung,EPoisson,EForce);

%*** add the surface integral "int_{Gamma_N} g^i . v_i dGamma" into RHS:
  FArea =  elmvol(XYZ,Fac);
  F = (FArea*[1/3 1/3 1/3]).*FForce;
  RHS = RHS + ...
        sparse(3*Fac(:,1)-2,1,F(:,1),3*nnod,1) + ...
        sparse(3*Fac(:,2)-2,1,F(:,1),3*nnod,1) + ...
        sparse(3*Fac(:,3)-2,1,F(:,1),3*nnod,1) + ...
        sparse(3*Fac(:,1)-1,1,F(:,2),3*nnod,1) + ...
        sparse(3*Fac(:,2)-1,1,F(:,2),3*nnod,1) + ...
        sparse(3*Fac(:,3)-1,1,F(:,2),3*nnod,1) + ...
        sparse(3*Fac(:,1)  ,1,F(:,3),3*nnod,1) + ...
        sparse(3*Fac(:,2)  ,1,F(:,3),3*nnod,1) + ...
        sparse(3*Fac(:,3)  ,1,F(:,3),3*nnod,1);


%*** impose Dirichlet boundary conditions on Gamma_D:
  RHS(pBC) = 0;
  dI = ones(3*nnod,1);
  dI(pBC) = 0;
  dI = spdiags(dI,0,3*nnod,3*nnod);
%--- add ones on the diagonal of K for Dirichlet nodes:
  K = dI*K*dI + (speye(3*nnod)-dI);

%*** visu: sparsity pattern of the stiffness matrix:
  figure(2); clf
  spy(K);
  set(gca,'FontSize',16);
  title('sparsity pattern of the stiffness matrix');

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

%*** get Cauchy strain and Euler stress:
  [EEps,ETau] = getstrainstress3D(XYZ,Elm,U,EYoung,EPoisson);

%*** calculate "von Mieses stress" criterion for plastic yield on tetras:
  EVonMieses = sqrt(((ETau(:,1)-ETau(:,2)).^2 + (ETau(:,2)-ETau(:,3)).^2 + ...
                     (ETau(:,1)-ETau(:,3)).^2 + 6*sum(ETau(:,4:6).^2,2))/2);
%--- interpolate on nodes:
  VonMieses = interpelm2nod(EVonMieses,Elm,XYZ);

%*** rearrange U(3*nnod,1) into U(nnod,3), with x+y+z displacement per line:
  U = reshape(U,3,nnod)';

%*** visualize the solution: deformed configuration (exaggerated 500x)
```

```
  UXYZ = XYZ + 500*U;
  figure(3); clf
  ts3a=trimesh(Fac,XYZ(:,1),XYZ(:,2),XYZ(:,3));
  hold on
  ts3b=trisurf(Fac,UXYZ(:,1),UXYZ(:,2),UXYZ(:,3),VonMieses);
  axis equal
  set(gca,'FontSize',16)
  shading interp                % <-- color interpolation
  set(ts3a,'EdgeColor','k');    % <-- edge color
  set(ts3a,'EdgeAlpha',0.1);    % <-- edge transparency
  set(ts3b,'EdgeColor','k');    % <-- edge color
  set(ts3b,'EdgeAlpha',0.1);    % <-- edge transparency
%  set(ts3b,'FaceAlpha',0.9);   % <-- face transparency
  hidden off
  cb3 = colorbar;
  set(cb3,'FontSize',16)
  view([0 0]);
  xl3 = xlabel('X');
  yl3 = ylabel('Y');
  zl3 = zlabel('Z');
  ti3 = title('deformed state (exaggerated 500x): von Mieses stress');
```



```
function [K,RHS] = mklinelast3D(XYZ,Elm,EYoung,EPoisson,EForce);
% function [K,RHS] = mklinelast3D(XYZ,Elm,EYoung,EPoisson,EForce);
%-------------------------------------------------------------------------
% assembles linear elasto-statics problem in 3D, small deformations
% homogeneous Neumann boundary conditions
%*** parameters:
% XYZ(nnod,3)                   -in-    nodal coordinates
% Elm(nelm,4)                   -in-    element connectivity (tetrahedron)
% EYoung(nelm,1)                -in-    Young modulus on each element (tetrahedr
on)
% EPoisson(nelm,1)              -in-    Poisson ratio on each element
% EForce(nelm,3)or(3*nelm,1)    -in-    volumic force density
% K(3*nnod,3*nnod)              -out-   sparse stiffness matrix
% RHS(3*nnod,1)                 -out-   right-hand side (volumic+surface forces)
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka/
%-------------------------------------------------------------------------

%--- no. of degrees of freedom per node (ndof), no. of nodes / elements:
  ndof = 3;
  nnod = size(XYZ,1);
  nelm = size(Elm,1);

%--- check shape of input arrays:
  EForce = reshape(EForce',3,nelm)';

%--- allocate new arrays:
  Kelm = zeros(ndof*4,ndof*4,nelm);
  RHS = zeros(ndof*nnod,1);

%*** loop over all elements, pre-store elementary matrices in Kelm():
  t0 = clock;
  for el = 1:nelm
    if (el==ceil(nelm/100))
      t = ceil(100*etime(clock,t0));
      fprintf('*** mklinelast3D: time to finish: %d min %d s\n',...
        floor(t/60),rem(t,60));
    end;
    EXYZ = XYZ(Elm(el,:),:);

%--- dofs: [ui vi , uj vj , uk vk , ul, vl]:
    [EK,ERHS] = elmlinelast3D(EXYZ,EYoung(el,:),EPoisson(el,:),EForce(el,:)');
    Kelm(1:4*ndof,1:4*ndof,el) = EK;
    for i=1:ndof
      RHS(ndof*(Elm(el,:)-1)+i,:) = RHS(ndof*(Elm(el,:)-1)+i,:) + ...
                                    ERHS(i:ndof:4*ndof,:);
    end;
  end;

%*** assemble the global sparse stiffness matrix from Kelm:
  ntot = ndof*nnod;
  K = sparse([],[],[],ntot,ntot);
  for ei=1:4
    ni = Elm(:,ei);
    for ej=1:4
      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;
return;


%-------------------------------------------------------------------------------
% subfunction elmlinelast3D:
%-------------------------------------------------------------------------------

function [EK,ERHS] = elmlinelast3D(XYZ,Eyoung,nu,Fvol)

%--- transform (E,nu) to (lambda, mu):
  mu2x   =  Eyoung/(1+nu);
  lambda =  Eyoung*nu/((1+nu)*(1-2*nu));

%--- basis functions phi_i(x,y,z) = ai + bi*x + ci*y + di*z:
  A = [ones(4,1), XYZ];
  X = inv(A);
  bi = X(2,1);  bj = X(2,2);  bk = X(2,3);  bl = X(2,4);
  ci = X(3,1);  cj = X(3,2);  ck = X(3,3);  cl = X(3,4);
  di = X(4,1);  dj = X(4,2);  dk = X(4,3);  dl = X(4,4);
  Vol = det(A)/6;


%*** Cauchy deformation tensor Eps(U) = Eps*U:
%    dofs in U: (x,y,z) displacement components == (u,v,w), arranged like:
%       [ui vi wi , uj vj wj , uk vk wk , ul vl wl]
%--- in Voigt notation (2nd order tensors in a vector):
  Eps = [bi    0    0 , bj    0    0 , bk    0    0 , bl    0    0;     % eps_11
           0   ci    0 ,  0   cj    0 ,  0   ck    0 ,  0   cl    0;     % eps_22
           0    0   di ,  0    0   dj ,  0    0   dk ,  0    0   dl;     % eps_33
          ci/2 bi/2  0 , cj/2 bj/2  0 , ck/2 bk/2  0 , cl/2 bl/2  0;     % eps_12
          di/2  0 bi/2 , dj/2  0 bj/2 , dk/2  0 bk/2 , dl/2  0 bl/2;     % eps_13
           0 di/2 ci/2 ,  0 dj/2 cj/2 ,  0 dk/2 ck/2 ,  0 dl/2 cl/2];    % eps_23
  I = diag([1 1 1 2 2 2]',0);

%--- average operator: Uave*U = average displacement (uT,vT,wT), const on tetra:
  Uave = [1  0  0 , 1  0  0 , 1  0  0 , 1  0  0;
          0  1  0 , 0  1  0 , 0  1  0 , 0  1  0;
          0  0  1 , 0  0  1 , 0  0  1 , 0  0  1]/4;

%--- elasticity modulus E_{ijkl} in Voigt notation:
  D = [mu2x+lambda  lambda      lambda      0     0     0    ;
       lambda       mu2x+lambda  lambda      0     0     0    ;
       lambda       lambda       mu2x+lambda  0     0     0    ;
       0            0            0           mu2x  0     0    ;
       0            0            0           0     mu2x  0    ;
       0            0            0           0     0     mu2x];

%--- hence, stress tensor Tau = D*Eps*U:
  EK   = Vol * Eps'*I*D*Eps;
  ERHS = Vol * Uave'*Fvol;

return;
```



```
function [EEps,ETau] = getstrainstress3D(XYZ,Elm,U,EYoung,EPoisson);
% function [EEps,ETau] = getstrainstress3D(XYZ,Elm,U,EYoung,EPoisson);
%-------------------------------------------------------------------------------
% calculates Cauchy deformation tensor and Euler stress for
% small deformations and linear material properties (Hooke)
% corresonding to the displacement field U
%*** parameters:
% XYZ(nnod,3)                  -in-    nodal coordinates
% Elm(nelm,4)                  -in-    element connectivity (tetrahedron)
% U(3*nnod,1) or (nnod,3)      -in-    displacement field
% EYoung(nelm,1)               -in-    Young modulus per tetra
% EPoisson(nelm,1)             -in-    Poisson ratio per tetra
% EEps(nelm,6)                 -in-    Cauchy strain (eps11,22,33,12,13,23) per
 tetra
% ETau(nelm,6)                 -in-    Euler stress (tau11,22,33,12,13,23) per
tetra
%-------------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka/
%-------------------------------------------------------------------------------

%--- no. of degrees of freedom per node (ndof), no. of nodes / elements:
  ndof = 3;
  nnod = size(XYZ,1);
  nelm = size(Elm,1);

%--- check shape of input arrays:
  U = reshape(U',3,nnod)';

%--- allocate new arrays:
  EEps = zeros(nelm,6);
  ETau = zeros(nelm,6);

%*** loop over all elements, pre-store elementary matrices in Kelm():
  t0 = clock;
  for el = 1:nelm
    if (el==ceil(nelm/100))
      t = ceil(100*etime(clock,t0));
      fprintf('*** getstrainstress3D: time to finish: %d min %d s\n',...
        floor(t/60),rem(t,60));
    end;
    EXYZ = XYZ(Elm(el,:),:);
    EU = U(Elm(el,:),:);

%--- dofs: [ui vi , uj vj , uk vk , ul, vl]:
    [eEps,eTau] = elmstrainstress3D(EXYZ,EU,EYoung(el,:),EPoisson(el,:));
    EEps(el,:) = eEps';
    ETau(el,:) = eTau';
  end;
return;


%-------------------------------------------------------------------------------
% subfunction elmstrainstress3D:
%-------------------------------------------------------------------------------

function [EpsU,TauU] = elmstrainstress3D(XYZ,U,Eyoung,nu)

%--- transform (E,nu) to (lambda, mu):
  mu2x   =  Eyoung/(1+nu);
  lambda =  Eyoung*nu/((1+nu)*(1-2*nu));
```

```
%--- reshape displacements in nodes, from (4x3) to (12,1) format:
  U = reshape(U',3*4,1);
%--- basis functions phi_i(x,y,z) = ai + bi*x + ci*y + di*z:
  A = [ones(4,1), XYZ];
  X = inv(A);
  bi = X(2,1);  bj = X(2,2);  bk = X(2,3);  bl = X(2,4);
  ci = X(3,1);  cj = X(3,2);  ck = X(3,3);  cl = X(3,4);
  di = X(4,1);  dj = X(4,2);  dk = X(4,3);  dl = X(4,4);
%  Vol = det(A)/6;


%*** Cauchy deformation tensor Eps(U) = Eps*U:
%    dofs in U: (x,y,z) displacement components == (u,v,w), arranged like:
%       [ui vi wi , uj vj wj , uk vk wk , ul vl wl]
%--- in Voigt notation (2nd order tensors in a vector):
  Eps = [bi    0    0 , bj    0    0 , bk    0    0 , bl    0    0;     % eps_11
           0   ci    0 ,  0   cj    0 ,  0   ck    0 ,  0   cl    0;     % eps_22
           0    0   di ,  0    0   dj ,  0    0   dk ,  0    0   dl;     % eps_33
          ci/2 bi/2  0 , cj/2 bj/2  0 , ck/2 bk/2  0 , cl/2 bl/2  0;     % eps_12
          di/2  0 bi/2 , dj/2  0 bj/2 , dk/2  0 bk/2 , dl/2  0 bl/2;     % eps_13
           0 di/2 ci/2 ,  0 dj/2 cj/2 ,  0 dk/2 ck/2 ,  0 dl/2 cl/2];    % eps_23
  EpsU = Eps*U;

%--- elasticity modulus E_{ijkl} in Voigt notation:
  D = [mu2x+lambda  lambda       lambda       0     0     0    ;
       lambda       mu2x+lambda  lambda       0     0     0    ;
       lambda       lambda       mu2x+lambda  0     0     0    ;
       0            0            0            mu2x  0     0    ;
       0            0            0            0     mu2x  0    ;
       0            0            0            0     0     mu2x];

%--- hence, stress tensor is D*EpsU:
  TauU = D*EpsU;
return;
```