```
%*** assembly and resolution of 3D linear elasto-statics (without temperature)
%     -div(Tau) = f  in the given 3D domain (long beam)
%             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
%-------------------------------------------------------------------------------


%*** 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)

%*** make 3D mesh:
  [XYZ,Elm] = mkcellhex3D(5,100,5);
  Fac = getbfaces4(Elm);
  XYZ(:,1) = XYZ(:,1)-max(XYZ(:,1))/2;
  XYZ(:,2) = XYZ(:,2)-max(XYZ(:,2))/2;
  XYZ(:,3) = XYZ(:,3)-max(XYZ(:,3))/2;
  XYZ = XYZ / max(XYZ(:,1));

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

%*** make the face reference numbers:
  FRef = ones(nfac,1);
  X = XYZ(:,1);
  Y = XYZ(:,2);
  Z = XYZ(:,3);
  ZFac = Z(Fac);
  XFac = X(Fac);
  f = find(max(ZFac,[],2)<min(Z)+1e-6);
  FRef(f,:) = 4;
  f = find(max(XFac,[],2)<min(X)+1e-6);
  FRef(f,:) = 3;

%*** define mechanical properties (const per element):
  EYoung   = 3e+9 * ones(nelm,1);       % <-- Young modulus 3GPa (nylon)
  EPoisson = 0.3 * ones(nelm,1);        % <-- Poisson ratio 0.3
  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 = 1e6*((FRef==3)*[1 0 0]);

%*** visualize the 3D mesh:
if 0
  figure(1); clf
  ts1=trisurf(Fac,XYZ(:,1),XYZ(:,2),XYZ(:,3),FRef);
  axis equal;
  set(ts1,'EdgeAlpha',0.1);
  colorbar('horiz');
  set(gca,'FontSize',16);
  xlabel('x');
  ylabel('y');
  zlabel('z');
  title('boundary faces and their reference numbers');
end;

%*** 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:
if 0
  figure(2); clf
  spy(K);
  set(gca,'FontSize',16);
  title('sparsity pattern of the stiffness matrix');
end;
```

```
%*** solution of the linear system: K*U = RHS:
  fprintf('*** solve: might take some time (~1min)\n');
%--- exact solver (fast for small problems, memory-consuming):
  U = K\RHS;
%--- conjugate gradients (for bigger problems, memory efficient but slow):
%  dK = spdiags(diag(K,0),0,3*nnod,3*nnod);
%  [U,pcgflag] = pcg(K,RHS,1e-6,10000,dK);

%*** get Cauchy strain and Euler stress:
  [EEps,ETau] = getstrainstress3D(XYZ,Elm,U,EYoung,EPoisson);
%--- interpolate strain and stress onto nodes:
  Eps = interpelm2nod(EEps,Elm,XYZ);
  Tau = interpelm2nod(ETau,Elm,XYZ);
%*** hydrostatic tension:
  Tension = (Tau(:,1)+Tau(:,2)+Tau(:,3))/3;

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

%*** exaggerated deformation:
  UXYZ = XYZ + 50*U;

%*** visualize the solution: deformed configuration (exaggerated 50x)
if 1
  figure(3); clf
  ts3a=trimesh(Fac,XYZ(:,1),XYZ(:,2),XYZ(:,3));
  set(gca,'FontSize',16)
  hold on
%  ts3b=trisurf(Fac,UXYZ(:,1),UXYZ(:,2),UXYZ(:,3),U(:,2));
  normUxz = sqrt(U(:,1).^2+U(:,3).^3);
  RelUy = U(:,2)./max(normUxz);
  ts3b=trisurf(Fac,UXYZ(:,1),UXYZ(:,2),UXYZ(:,3),RelUy);
  axis equal
  shading interp                % <-- color interpolation
%  caxis([min(U(:,2)),max(U(:,2))]);
  caxis([min(RelUy),max(RelUy)]);
  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('horiz');
  set(cb3,'FontSize',14)
  view([-45 20]);
  xl3 = xlabel('X');
  yl3 = ylabel('Y');
  zl3 = zlabel('Z');
  ti3 = title('U_y relative to max. displacement in X-Z plane');
end;


%*** visu solution at mid-plane: hydrostatic tension:
if 1
  Y = XYZ(:,2);
  YElm = Y(Elm);
  e = find(max(abs(YElm),[],2)<=0.8);
  Fac2 = getbfaces4(Elm(e,:));
  p = unique(Fac2);
  figure(4); clf
  ts4a=trisurf(Fac2,UXYZ(:,1),UXYZ(:,2),UXYZ(:,3),Tension/1e6);
  set(gca,'FontSize',16)
```



```
  axis equal
  shading interp                % <-- color interpolation
  set(ts4a,'EdgeColor','k');     % <-- edge color
  set(ts4a,'EdgeAlpha',0.1);     % <-- edge transparency
  set(ts4a,'FaceAlpha',0.5);    % <-- face transparency
  caxis([-1.5 1.5])
  cb4 = colorbar;
  set(cb4,'FontSize',14)
  view([-45 20]);
  xl4 = xlabel('X');
  yl4 = ylabel('Y');
  zl4 = zlabel('Z');
  ti4 = title('hydrostatic tension [MPa], deformation exaggerated 50x');
end;
```

```
%*** assembly of a plane strain problem:
%    instead of solving the whole (very) long beam in 3D, we assume that
%    displacements/deformations along the long axis are negligible
%    especially at the mid-section region far from both ends
%    hence, we take eps_3k = eps_k3 = 0, k=1..3, and we make a reduced
%    model in 2D only.
%--- this model can then be compared with the full 3D calculations:
%    a) run "program1" --> fig.3 and 4. with the results
%    b) run "program2" --> results of the reduced model are
%       superposed on the fig.4
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

%*** get 2D mesh of the mid-section:
  [XY,Elm] = getdrapeau3(10,10);
  [G,Border,Fac] = incidence2(Elm);

%*** sizes:
  nnod = size(XY,1);
  nelm = size(Elm,1);
  nfac = size(Fac,1);

%*** get border reference:
  FRef = ones(nfac,1);
  X = XY(:,1);
  Z = XY(:,2);
  XFac = X(Fac);
  ZFac = Z(Fac);
  f3 = find(max(XFac,[],2)<min(X)+1e-6);
  f4 = find(max(ZFac,[],2)<min(Z)+1e-6);
  FRef(f3) = 3;
  FRef(f4) = 4;
  p4 = unique(Fac(f4,:));       % <-- nodal indices on faces FRef==4

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

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

%*** body force density due to gravity, in [N/m^3]:
  EForce = zeros(nelm,2);
  EForce(:,2) = -density * gravity;

%*** surfacic force density traction on surface with reference no. 3, in [N/m^2]
:
  FForce = 1e6*((FRef==3)*[1 0]);

%*** assemble the 2D plane-strain problem with homog. Neumann everywhere:
  [K,RHS] = mkplanestrain(XY,Elm,EYoung,EPoisson,EForce);

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



```
        sparse(2*Fac(:,2)-1,1,F(:,1),2*nnod,1) + ...
        sparse(2*Fac(:,1)  ,1,F(:,2),2*nnod,1) + ...
        sparse(2*Fac(:,2)  ,1,F(:,2),2*nnod,1);

%*** impose Dirichlet boundary conditions:
  dI = ones(2*nnod,1);
  dI(pBC) = 0;
  dI = spdiags(dI,0,2*nnod,2*nnod);
  RHS = dI'*RHS;
  K = dI'*K*dI + (speye(2*nnod)-dI);

%*** solution:
  U = K\RHS;
  U = reshape(U,2,nnod)';

%*** get strain and stress tensors (3D tensors!):
  [EEps,ETau] = getstrainstress2D(XY,Elm,U,EYoung,EPoisson);
%*** interpolate on nodes:
  Eps = interpelm2nod(EEps,Elm,XY);
  Tau = interpelm2nod(ETau,Elm,XY);
%*** hydrostatic tension:
  Tension = (Tau(:,1)+Tau(:,2)+Tau(:,3))/3;

%*** visu: displacements (exaggerated 50x):
  figure(4); hold on
  UXY = XY + 50*U;
  ts4b = trisurf(Elm,UXY(:,1),0*UXY(:,1),UXY(:,2),Tension/1e6);
  shading interp
  set(ts4b,'EdgeColor','k');     % <-- edge color
```

```
function [K,RHS] = mkplanestrain(XY,Elm,EYoung,EPoisson,EForce);
% function [K,RHS] = mkplanestrain(XY,Elm,EYoung,EPoisson,EForce);
%-------------------------------------------------------------------------
% computes the stiffnes matrix for plane-strain, small deformations
% isotropic linear compressible material (Hooke)
%*** parameters:
% XY(nnod,2)                  -in-    nodal coordinates
% Elm(nelm,3)                 -in-    element connectivity (triangles)
% EYoung(nelm,1)              -in-    young modulus on each triangle
% EPoisson                    -in-    poisson ratio on each triangle
% EForce(nelm,2)or(2*nelm,1)  -in-    force density on each element
% K(2*nnod,2*nnod)            -out-   sparse stiffness matrix
% RHS(2*nnod,1)               -out-   forces due to initial pre-strain
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

  ndof = 2;
  nnod = size(XY,1);
  nelm = size(Elm,1);
  Kelm = zeros(ndof*3,ndof*3,nelm);
  RHS = zeros(ndof*nnod,1);
  t0 = clock;

  EForce = reshape(EForce',2,nelm)';

%*** loop over all elements, pre-store elementary matrices in Kelm():
  for el = 1:nelm
    if (el==ceil(nelm/100))
      t = ceil(100*etime(clock,t0));
      fprintf('*** mkplanestrain: time to finish: %d min %d s\n',...
        floor(t/60),rem(t,60));
    end;
    EXY = XY(Elm(el,:),:);
%--- dofs: [ui vi , uj vj , uk vk]:
    [EK,ERHS] = elmplanestrain(EXY,EYoung(el),EPoisson(el),EForce(el,:)');
    Kelm(1:3*ndof,1:3*ndof,el) = EK;
    for i=1:ndof
      RHS(ndof*(Elm(el,:)-1)+i,:) = RHS(ndof*(Elm(el,:)-1)+i,:) + ...
                                    ERHS(i:ndof:3*ndof,:);
    end;
  end;

%*** assemble the global sparse stiffness matrix from Kelm:
  ntot = ndof*nnod;
  K = sparse([],[],[],ntot,ntot);
  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;
return;
```



```


%-------------------------------------------------------------------------
% subfunction elmplanestrain:
%-------------------------------------------------------------------------

function [EK,ERHS] = elmplanestrain(XY,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:
  A = [ones(3,1), [XY(1,:);XY(2,:);XY(3,:)]];
  X = inv(A);
  bi = X(2,1);  bj = X(2,2);  bk = X(2,3);
  ci = X(3,1);  cj = X(3,2);  ck = X(3,3);
  Area = det(A) / 2;

%*** Cauchy deformation tensor Eps(U) = Eps*U:
%    dofs in U: (x,y) displacement components == (u,v), arranged like:
%       [ ui vi , uj vj , uk vk ]
%--- in Voigt notation (2nd order tensors in a vector):
  Eps = [   bi     0   ,  bj     0   ,   bk    0  ;     % <-- eps_11
            0    ci   ,  0     cj   ,   0    ck  ;     % <-- eps_22
          ci/2  bi/2  , cj/2  bj/2  , ck/2  bk/2 ];    % <-- eps_12

  I = diag([1 1 2]',0);

%--- average operator: Uave*U = average displacement (uT,vT,wT), const on triang
le:
  Uave = [1 0, 1 0, 1 0;
          0 1, 0 1, 0 1]/3;

%--- elasticity modulus E_{ijkl} for 2D plane-strain in Voigt notation:
  D = [mu2x+lambda  lambda       0     ;
       lambda       mu2x+lambda  0     ;
       0            0            mu2x  ];

%--- elementary stiffness matrix and right-hand side:
  EK = Area * Eps'*I*D*Eps;
  ERHS = Area * Uave'*Fvol;

return;
```

```
function [Eps,Tau] = getstrainstress2D(XY,Elm,U,EYoung,EPoisson);
% function [Eps,Tau] = getstrainstress2D(XY,Elm,U,EYoung,EPoisson);
%-------------------------------------------------------------------------
% plane strain, linear isotropic material (Hooke)
% computes strain and stress (all components, as if in 3D!)
%*** parameters:
% XY(nnod,2)            -in-    nodal coords
% Elm(nelm,3)           -in-    element connectivity
% U(2*nnod,1)           -in-    nodal displacements, U(1:2)=1st node in X and Y
%  or U(nnod,2)
% EYoung(nelm,1)        -in-    Young modulus on each triangle
% EPoisson(nelm,1)      -in-    Poisson ratio on each triangle
% Eps(nelm,6)           -out-   eps11,22,33,12,13,23, strain tensor components
% Tau(nelm,6)           -out-   tau11,22,33,12,13,23, stress tensor components
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch            Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

  ndof = 2;

  nnod = size(XY,1);
  nelm = size(Elm,1);
  Eps = zeros(nelm,6);
  Tau = zeros(nelm,6);

  t0 = clock;

%*** reshape U so that it is (nnod x ndof):
  U = reshape(U',ndof,nnod)';

%*** loop over all elements:
  for el = 1:nelm
    if (el==ceil(nelm/100))
      t = ceil(100*etime(clock,t0));
      fprintf('*** getstrainstress2D: time to finish: %d min %d s\n',…
        floor(t/60),rem(t,60));
    end;
    EXY = XY(Elm(el,:),:);
    EU = U(Elm(el,:),:);
    EU = reshape(EU', ndof*3,1);
%*** dofs: [ui vi , uj vj , uk vk]:
    [EEps,ETau] = elmgetstrain2D(EXY,EU,EYoung(el),EPoisson(el));
    Eps(el,:) = EEps';
    Tau(el,:) = ETau';
  end;
return;


%-------------------------------------------------------------------------
% subfunction elmgetstrain2D
%-------------------------------------------------------------------------

function [EpsU,TauU] = elmgetstrain2D(XY,U,Eyoung,nu);

%--- 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:
  A = [ones(3,1), [XY(1,:);XY(2,:);XY(3,:)]];
  X = inv(A);
```



```
  bi = X(2,1); bj = X(2,2); bk = X(2,3);
  ci = X(3,1); cj = X(3,2); ck = X(3,3);

%--- dof: [ ui vi , uj vj , uk vk] :

%*** Cauchy deformation tensor Eps(U) = Eps*U:
%    dofs in U: (x,y) displacement components == (u,v), arranged like:
%       [ ui vi , uj vj , uk vk ]
%--- Eps is a 3D tensor! in Voigt notation (2nd order tensors in a vector):
  Eps = [ bi    0  ,   bj   0  ,  bk    0 ;            % <-- eps_11
            0   ci  ,    0  cj  ,   0   ck ;            % <-- eps_22
            0    0  ,    0   0  ,   0    0 ;            % <-- eps_33
          ci/2 bi/2 , cj/2 bj/2 , ck/2 bk/2;            % <-- eps_12
            0    0  ,    0   0  ,   0    0 ;            % <-- eps_13
            0    0  ,    0   0  ,   0    0 ];           % <-- 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];

%--- calculate 3D stress tensor:
  TauU = D*EpsU;
return;
```