```
% direct (displacement only) formulation of linear plane-strain problem
% P1 finite elements for displacement, no pressure
% works only for compressible materials, cf. DATA.Epoisson below
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

%*** mechanical data:
  DATA.plane_stress = 0;
  DATA.density  = 1e-4;
  DATA.Eyoung   = 1.5e+0;
  DATA.Epoisson = 0.3;
  DATA.gravity = 10;

%*** get the mesh:
  xaspect = 3;
  [XY,Elm] = getdrapeau3(xaspect*15+1,15+1);
  nnod = size(XY,1);
  nelm = size(Elm,1);
  [G,Border,bfaces] = incidence2(Elm);
  XY(:,1) = XY(:,1)*xaspect;


%*** force density:
  EForce = zeros(nelm,2);
  EForce(:,2) = -DATA.gravity * DATA.density;

%*** assemble the matrix:
  [K,RHS] = mkelasticity2D(XY,Elm,DATA,EForce);

%*** Dirichlet boundary conditions:
  nodBC = find(XY(:,1)<=(min(XY(:,1))+1e-6));
  pBC = [2*nodBC-1;
         2*nodBC];
  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);

  U = K\RHS;
  U = reshape(U,2,nnod)';

%*** visu: displacements:
  figure(1); clf
  gplot(G,XY);
  set(gca,'FontSize',16);
  hold on
  gplot(G,XY+U,'r');
  axis equal
  title('displacement');
  grid on
  drawnow


%*** calculate pressure on elements: p = -trace(Tau)/3:
  [Eps,Tau] = getstrainstress2D(XY,Elm,U,DATA.Eyoung,DATA.Epoisson);
  EP = -sum(Tau(:,1:3),2)/3;

%*** visu: pressure on elements:
```



```
  figure(2); clf
  ts2 = trisurf(Elm,XY(:,1),XY(:,2),0.*XY(:,1),EP);
  set(gca,'FontSize',16);
  axis equal;
  view(2);
  set(ts2,'EdgeColor','none');
  cb = colorbar;
  set(cb,'FontSize',16);
  title('pressure');
  drawnow
```

```
function [K,RHS] = mkelasticity2D(XY,Elm,DATA,EF,Tau0);
% function [K,RHS] = mkelasticity2D(XY,Elm,DATA,EF,Tau0);
%-------------------------------------------------------------------------
% computes the stiffnes matrix for plane-stress or plain-strain
% works only for small deformations!
%*** parameters:
% XY(nnod,2)            -in-    nodal coordinates
% Elm(nelm,3)           -in-    element connectivity (triangles)
% DATA                  -in-    mechanical data (structure)
% DATA.plane_stress     -in-    0: plane strain, 1: plane stress
% DATA.Eyoung           -in-    Young modulus
% DATA.Epoisson         -in-    Poisson ratio in [-1,0.5] (0.5==incompressible)
% EF(nelm,2)or(2*nelm,1)-in-    force density on each element
% Tau0(nelm,3)          -in-    (facultative) init.stress (tau11,22,12 per elm)
% K(2*nnod,2*nnod)      -out-   sparse stiffness matrix
% RHS(2*nnod,1)         -out-   forces due to initial pre-strain
%-------------------------------------------------------------------------

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

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

  if (nargin<5)
    Tau0 = zeros(nelm,3);
  end;

%*** loop over all elements, pre-store elementary matrices in Kelm():
  for el = 1:nelm
    EXY = XY(Elm(el,:),:);
%--- dofs: [ui vi , uj vj , uk vk]:
    [EK,ERHS] = elmelasticity2D(EXY,DATA,EF(el,:)',Tau0(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 elmelasticity2D:
%-------------------------------------------------------------------------

function [EK,ERHS] = elmelasticity2D(XY,DATA,Fvol,Tau0)
  plane_stress = DATA.plane_stress;
  Eyoung = DATA.Eyoung;
  nu = DATA.Epoisson;
  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);
  Area2x = det(A);
%--- dof: [ ui  vi ,  uj  vj , uk  vk] :
  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
  Uave = [1 0, 1 0, 1 0;
          0 1, 0 1, 0 1]/3;

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

  if (plane_stress)
%--- 2D elasticity: plain stress: watch out D(3,3) <--> I(3,3)!
    D = Eyoung / (1-nu^2) * [1   nu    0;
                             nu   1    0;
                             0    0  (1-nu)];
  else
%--- 2D elasticity: plain strain: watch out D(3,3) <--> I(3,3)!
    D = Eyoung / ((1+nu)*(1-2*nu)) * [1-nu   nu    0;
                                      nu   1-nu   0;
                                       0     0  (1-2*nu)];
    D(3,3) = Eyoung / (1+nu);
  end;

%--- elementary stiffness matrix and right-hand side:
  Vol = Area2x/2;
  EK = Vol * Eps'*D*I*Eps;
  ERHS = Vol * Uave'*Fvol + ...
        -Vol * Eps'*I*Tau0;

return;
```

```
% mixed formulation of linear plane-strain problem
% P1-P0 finite element formulation, ie. continuous piecewise linear displacement
% and discontinuous piecewise constant pressure
% works also for compressible materials, cf. DATA.Epoisson below
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

%*** mechanical data:
  DATA.density  = 1e-4;
  DATA.Eyoung   = 1.5e+0;
  DATA.Epoisson = 0.3;
  DATA.gravity = 10;

%*** get the mesh:
  xaspect = 3;
  [XY,Elm,ECell] = getdrapeau3(xaspect*15+1,15+1);
  XY(:,1) = XY(:,1)*xaspect;

%*** for stability reasons, the following 2 lines have to be INACTIVE:
  nelm = size(Elm,1);
  ECell = (1:nelm)';

%*** sizes:
  nnod = size(XY,1);
  nelm = size(Elm,1);
  ncell = max(ECell);
%*** total no.of unknowns: 2 displacements on each node+1 pressure on each cell
  ntot = 2*nnod + ncell;

%*** body force density (constant on elements):
  EForce = zeros(nelm,2);
  EForce(:,2) = -DATA.gravity * DATA.density;

%*** mass/volume change:
  CQ = zeros(ncell,1);

%*** mesh incidence (node neighbourhood, border node list, border edges):
  [G,Border,bfaces] = incidence2(Elm);

%*** Dirichlet boundary conditions:
  nodBC = find(XY(:,1)<=(min(XY(:,1))+1e-6));
  pBC = [2*nodBC-1;
         2*nodBC];
  dI = ones(ntot,1);
  dI(pBC) = 0;
  dI = spdiags(dI,0,ntot,ntot);

%*** assemble linear elasticity problem, mixed formulation P1-P0 finite elems:
  [K,RHS] = mkelasticity2Dp1p0(XY,Elm,ECell,DATA,EForce,CQ);

%*** Dirichlet in right-hand side:
  RHS = dI'*RHS;

%*** Dirichlet in the stiffness matrix:
  K = dI'*K*dI + (speye(ntot)-dI);

%*** solve:
  X = K\RHS;

%*** displacement on nodes:
```



```
  U = [X(1:2:2*nnod,1) X(2:2:2*nnod,1)];

%*** cell pressures:
  CP = X(2*nnod+1:end,1);

%*** visu: displacement
  figure(1); clf
  gplot(G,XY,'b');
  set(gca,'FontSize',16);
  hold on
  gplot(G,XY+U,'r');
  axis equal
  title('displacements')
  grid on
  drawnow

%*** visu: pressure
  figure(2); clf
  ts2 = trisurf(Elm,XY(:,1),XY(:,2),0.*XY(:,1),CP(ECell));
  set(gca,'FontSize',16);
  view(2);
  axis equal tight
  set(ts2,'EdgeColor','none');
  cb=colorbar;
  set(cb,'FontSize',16);
  title('pressure');
  drawnow;
```

```
function [K,RHS] = mkelasticity2Dp1p0(XY,Elm,ECell,DATA,EForce,CQ);
% function [K,RHS] = mkelasticity2Dp1p0(XY,Elm,ECell,DATA,EForce,CQ);
%-------------------------------------------------------------------------
% assembles linear plane-strain, small displacements
% mixed formulation (deviators + hydrostatic pressure P)
% P1-P0 finite elements (continuous piecewise linear displacements,
% discontinuous piecewise const pressure
% totoal no of degrees of freedom: ntot = 2*nnod + ncell
% DOFs are arranged like [ux1 uy1, ux2 uy2,...,ux_nnod uy_nnod, p1, p2...pncell]
% ATTENTION: be sure that cells are numbered 1..ncell without gaps!
%*** parameters:
% XY(nnod,2)            -in- nodal coordinates
% Elm(nelm,3)           -in- element connectivity (triangles)
% ECell(nelm,1)         -in- coloring of triangles into cells, ncell=max(ECell)
% DATA                  -in- mechanical data (structure)
% DATA.Eyoung           -in- Young modulus
% DATA.Epoisson         -in- Poisson ratio in [-1,0.5] (0.5==incompressible)
% EForce(nelm,2) or
% EForce(2*nelm,1)      -in- force density on each element
% CQ(ncell,1)           -in- volume change on each cell (0== no change)
% K(ntot,ntot)          -out- sparse stiffness matrix
% RHS(ntot,1)           -out- right-hand side (ux, uy, p)
%-------------------------------------------------------------------------

%*** sizes:
  nnod = size(XY,1);
  nelm = size(Elm,1);
  ncell = max(ECell);
  ntot = 2*nnod + ncell;

  if (size(DATA.Eyoung,1)==nelm)
    Eyoung = DATA.Eyoung;
  else
    Eyoung = DATA.Eyoung(1)*ones(nelm,1);
  end;

%*** check dimensions of force:
  EForce = reshape(EForce',2,nelm)';

%--- allocate new arrays:
  Kelm = zeros(7,7,nelm);
  RHS = zeros(ntot,1);

%*** passing from linear pressure to constant pressure per element:
  P = eye(9);
  P(:,[3 6]) = [];
  P([3 6 9],7) = 1;

  t0 = clock;
  for el = 1:nelm
    EXY = XY(Elm(el,:),:);
%*** dofs: KE: [ui vi , uj vj , uk vk , p]:
    [EK,ERHS]=elmelasticity2Dp1p0(EXY,DATA,EForce(el,:)',CQ(ECell(el)));
%*** store elementary matrix:
    Kelm(:,:,el) = EK;
%*** assemble RHS: first displacements, then pressure:
    for i=1:2
      RHS(2*(Elm(el,:)-1)+i,:) = RHS(2*(Elm(el,:)-1)+i,:) + ERHS(i:2:6,:);
    end;
    RHS(2*nnod+ECell(el),:) =  RHS(2*nnod+ECell(el),:) + ERHS(7,:);
  end;
```



```
%*** assemble global contributions to global stiffness matrix: displacement:
  K = sparse([],[],[],ntot,ntot);
  for ei=1:3
    ni = Elm(:,ei);
%--- displacement-displacement:
    for ej=1:3
      nj = Elm(:,ej);
      for i=1:2
        for j=1:2
          ki = 2*(ni-1)+i;
          kj = 2*(nj-1)+j;
          kv = squeeze(Kelm(2*(ei-1)+i,2*(ej-1)+j,:));
          K = K + sparse(ki,kj,kv,ntot,ntot);
        end;
      end;
    end;
%--- displacement-pressure and pressure-displacement:
    for i=1:2
      ki = 2*(ni-1)+i;
      kj = 2*nnod + ECell;
      kv = squeeze(Kelm(2*(ei-1)+i,7,:));
      K = K + sparse(ki,kj,kv,ntot,ntot);
      kv = squeeze(Kelm(7,2*(ei-1)+i,:));
      K = K + sparse(kj,ki,kv,ntot,ntot);
    end;
  end;
%--- pressure-pressure block:
  kv = squeeze(Kelm(7,7,:));
  K = K + sparse(kj,kj,kv,ntot,ntot);
return;

%-------------------------------------------------------------------------------
% subfunction elmelasticity2Dp1p0:
%-------------------------------------------------------------------------------

function [EK,ERHS]=elmelasticity2Dp1p0(XY,DATA,Fvol,Qvol)
  Eyoung = DATA.Eyoung;
  Epoisson = DATA.Epoisson;
  Emu = 0.5*Eyoung/(1+Epoisson);
  if (abs(Epoisson-0.5)<1e-10)
    Elambda = Inf;
  else
    Elambda = 2*Emu*Epoisson/(1-2*Epoisson);
  end;
%--- basis functions:
  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);
  Area2x = det(A);
%*** dof: [ ui vi  ,  uj   vj  ,  uk   vk  ,  p], operators:
  Eps = [ bi   0   ,  bj    0  ,  bk    0  ,  0;        % <-- eps_11
           0  ci   ,   0   cj  ,   0   ck  ,  0;        % <-- eps_22
         ci/2 bi/2 , cj/2 bj/2 , ck/2 bk/2 ,  0];       % <-- eps_12
  DivU = [bi  ci   ,  bj   cj  ,  bk   ck  ,  0];
  Pave = [ 0   0   ,   0    0  ,   0    0  ,  1];
  Uave =  [1 0 , 1 0 , 1 0 , 0;
           0 1 , 0 1 , 0 1 , 0]/3;
  Pmass = zeros(7,7); Pmass(7,7) = 1;
  I = diag([1 1 2]',0);

%--- assemble stiffness matrix and right-hand side:
```

```
  Vol = Area2x/2;
  EK = Vol * 2*Emu * (Eps'*I*Eps - 1/3*DivU'*DivU) + ...
      -Vol * (DivU'*Pave + Pave'*DivU) + ...
      -Vol * 3/(3*Elambda+2*Emu)*Pmass;
  ERHS = Vol * Uave'*Fvol + ...
       -Vol * Pave'*Qvol;
%--- the stiffness matrix should be symmetric (symmetrize it perfectly):
  EK = (EK+EK')/2;

return;
```



```
% mixed formulation of linear plane-strain problem
% (GLS-stabilized) P1-P1 finite element formulation,
% ie. continuous piecewise linear displacement
% and continuous piecewise linear pressure
% works also for compressible materials, cf. DATA.Epoisson below
%-------------------------------------------------------------------------
% mail: ales.janka@unifr.ch                    Universite de Fribourg, 2011
% http://perso.unifr.ch/ales.janka
%-------------------------------------------------------------------------

%*** mechanical data:
  DATA.alpha = 0.0;             % <-- GLS stabilization param, should be O(1)
  DATA.density  = 1e-4;
  DATA.Eyoung   = 1.5e+0;
  DATA.Epoisson = 0.3;
  DATA.gravity = 10;

%*** get the mesh:
  xaspect = 3;
  [XY,Elm] = getdrapeau3(xaspect*15+1,15+1);
  XY(:,1) = XY(:,1)*xaspect;


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

%*** mesh node to node incidence, border nodes, border edges:
  [G,Border,bfaces] = incidence2(Elm);

%*** force density:
  EForce = zeros(nelm,2);
  EForce(:,2) = -DATA.gravity * DATA.density;

%*** relative change in volume:
  EQ = zeros(nelm,1);

%*** assemble linear elasticity in mixed GLS stabilized form:
  [K,RHS] = mkelasticity2Dgls(XY,Elm,DATA,EForce,EQ);

%*** Dirichlet boundary conditions:
  nodBC = find(XY(:,1)<=(min(XY(:,1))+1e-6));
  pBC = [3*nodBC-2;
         3*nodBC-1];
  dI = ones(3*nnod,1);
  dI(pBC) = 0;
  dI = spdiags(dI,0,3*nnod,3*nnod);

  RHS = dI'*RHS;
  K = dI'*K*dI + (speye(3*nnod)-dI);

  X = K\RHS;
  X = reshape(X,3,nnod)';

%*** separate solution X to displacements and pressure:
  U = X(:,1:2);
  P = X(:,3);

%*** visu: displacements:
  figure(1); clf
  gplot(G,XY);
  set(gca,'FontSize',16);
```

```
  hold on
  gplot(G,XY+U,'r');
  axis equal
  title('displacement');
  grid on
  drawnow;

%*** visu: hydrostatic pressure:
  figure(2); clf
  ts2 = trisurf(Elm,XY(:,1),XY(:,2),0.*XY(:,1),P);
  set(gca,'FontSize',16);
  axis equal tight;
  view(2);
  shading interp
  cb=colorbar;
  set(cb,'FontSize',16);
  title('pressure');
  drawnow;
```



```
function [K,RHS] = mkelasticity2Dgls(XY,Elm,DATA,EF,EQ);
% function [K,RHS] = mkelasticity2Dgls(XY,Elm,DATA,EF,EQ);
%-------------------------------------------------------------------------
% linear elasticity in 2D, plane strain, small deformations.
% Mixed formulation (through deviators and hydrostatic pressure)
%*** parameters:
% XY(nnod,2)            -in-    nodal coordinates
% Elm(nelm,3)           -in-    element connectivity (triangles)
% DATA                  -in-    mechanical data (structure)
% DATA.Eyoung           -in-    Young modulus
% DATA.Epoisson         -in-    Poisson ratio in [-1,0.5] (0.5==incompressible)
% DATA.alpha            -in-    O(1) stabilisation parameter
% EF(nelm,2)or(2*nelm,1)-in-    force density on each element
% EQ(nelm,1)            -in-    relative volume change per element
% K(3*nnod,3*nnod)      -out-   sparse stiffness matrix
% RHS(3*nnod,1)         -out-   right-hand side (ux, uy, p)
%-------------------------------------------------------------------------

  ndof = 3;
  nnod = size(XY,1);
  nelm = size(Elm,1);

%*** check dimensions of force and allocate arrays:
  EF = reshape(EF',2,nelm)';
  Kelm = zeros(ndof*3,ndof*3,nelm);
  RHS = zeros(ndof*nnod,1);

  for el = 1:nelm
    EXY = XY(Elm(el,:),:);
%*** dofs: KE: [ui vi pi , uj vj pj , uk vk pk]:
    [EK,ERHS]=elmelasticity2Dgls(EXY,DATA,EF(el,:)',EQ(el,1));
    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 global contributions to global stiffness matrix:
  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 elmelasticity2Dgls:
%-------------------------------------------------------------------------
```

```
function [EK,ERHS]=elmelasticity2Dgls(XY,DATA,Fvol,Qvol,EpsU0)
  Eyoung = DATA.Eyoung;
  Epoisson = DATA.Epoisson;
  Emu = 0.5*Eyoung/(1+Epoisson);
  if (abs(Epoisson-0.5)<1e-10)
    Elambda = Inf;
  else
    Elambda = 2*Emu*Epoisson/(1-2*Epoisson);
  end;
  alpha = DATA.alpha;
%--- basis functions:
  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);
  Area2x = det(A);
%*** dof: [ ui vi pi , uj vj pj, uk vk pk], operators:
  Eps = [ bi   0    0, bj   0    0, bk   0    0;        % <-- eps_11
           0  ci    0,  0  cj    0,  0  ck    0;        % <-- eps_22
         ci/2  bi/2 0, cj/2 bj/2 0, ck/2 bk/2 0];       % <-- eps_12
  DivU = [bi ci 0, bj cj  0, bk ck  0];
  GradP = [0 0 bi,  0  0 bj,  0  0 bk;
           0 0 ci,  0  0 cj,  0  0 ck];
  Pave  = [0 0 1, 0 0 1, 0 0 1]/3;
  Uave =  [1 0 0, 1 0 0, 1 0 0;
           0 1 0, 0 1 0, 0 1 0]/3;
  Pmass1 = (ones(3,3) + eye(3))/12;
  Pmass = spalloc(9,9,9);
  Pmass(3:3:end,3:3:end) = Pmass1;
  I = diag([1 1 2]',0);

%--- characteristic size of the element:
  H = [norm(XY(1,:)-XY(2,:));
       norm(XY(1,:)-XY(3,:));
       norm(XY(2,:)-XY(3,:))];
  Hsq = min(H)^2;

%--- assemble stiffness matrix and right-hand side:
  Vol = Area2x/2;
  EK = Vol * 2*Emu * (Eps'*I*Eps - 1/3*DivU'*DivU) + ...
      -Vol * (DivU'*Pave + Pave'*DivU) + ...
      -Vol * 3/(3*Elambda+2*Emu)*Pmass + ...
      -Vol * (alpha*Hsq/(2*Emu))* (GradP'*GradP);
  ERHS = Vol * Uave'*Fvol + ...
        -Vol * Pave'*Qvol + ...
        -Vol * (alpha*Hsq/(2*Emu))*(GradP'*Fvol);
%--- the stiffness matrix should be symmetric (symmetrize it perfectly):
  EK = (EK+EK')/2;

return;
```