ParallelInternalWorkResidualStiffnessU (Calls: 60000, Time: 43.713 sec)

Generated 15-Nov-2017 15:05:05 using performance time.
function in file C:\SoftwareDevelopment\OPTIMISATION_CODE\code\assembly\ParallelInternalWo
Copy to new window for comparing multiple runs

Refresh

☑ Show parent functions ☑ Show busy lines ☑ Show child functions

☑ Show Code Analyzer results ☑ Show file coverage ☑ Show function listing

Parents (calling functions)

Function Name	Function Type	Calls
ParallelInternalWorkUAssembly	function	60000

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time
100	`kinematics.H(:,:,igauss));`	240000	14.168 s	32.4%
43	`mat_info,mat_info.material_mod...`	60000	5.724 s	13.1%
53	`mat_info.optimisation.rho(iele...`	60000	5.464 s	12.5%
23	`fem.volume.bilinear.x.DN_X);`	60000	2.565 s	5.9%
65	`mat_info.derivatives.DU.DUDJ);`	60000	1.860 s	4.3%
All other lines			13.932 s	31.9%
Totals			43.713 s	100%

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Tin Plo
VectorisedStiffnessMatricesU	function	240000	13.013 s	29.8%	■
GetDerivativesModelMechanics	function	120000	6.456 s	14.8%	■
SumDerivativesU	function	60000	4.905 s	11.2%	■

rkResidualStiffnessU.m

Time Plot
▬
▬
▬
▮
▮
▬

ne ot
▬

KinematicsFunctionVolumeCMex	MEX-file	60000	2.019 s	4.6%	▮
FirstPiolaKirchhoffStressTensorUCMex	MEX-file	60000	1.282 s	2.9%	▮
ElementResidualMatricesInitialisationU	function	60000	0.957 s	2.2%	▮
BMatrix	function	60000	0.723 s	1.7%	▮
QMatrixComputation	function	120000	0.706 s	1.6%	▮
Matrix2Vector	function	60000	0.689 s	1.6%	▮
Self time (built-ins, overhead, etc.)			12.962 s	29.7%	▬
Totals			43.713 s	100%	

Code Analyzer results

No Code Analyzer messages.

Coverage results

Show coverage for parent directory

Total lines in function	132
Non-code lines (comments, blank lines)	74
Code lines (lines that can run)	58
Code lines that did run	44
Code lines that did not run	14
Coverage (did run/can run)	75.86 %

Function listing

Color highlight code according to time

```
 time    Calls   line
                   1  function asmb          =  ParallelInternalWo
                   2
                   3
  0.21   60000     4  ngauss                 =  size(quadrature.vo
                   5  %---------------------------------------------
                   6  % Initialise assembled residuals per element
                   7  %---------------------------------------------
  1.10   60000     8  asmb                   =  ElementResidualMat
                   9  %---------------------------------------------
                  10  % Obtain gradients of kinematics and electric
                  11  %---------------------------------------------
                  12  % kinematics             =  KinematicsFunctic
                  13  %                                   solutior
                  14  %                                   fem.volu
```

```
rkResidualStiffnessU(ielem,quadrature,solution,geometry,mesh,fem,...
                        vectorisation,mat_info,mat_info_void,...
                        kinematics)
lume.bilinear.Chi,1);
-------------------------------

-------------------------------
ricesInitialisationU (geometry,mesh);
-------------------------------
cal variables
-------------------------------
onVolume(geometry.dim,...
n.x.Eulerian_x(:,mesh.volume.x.connectivity(:,ielem)),...
ume.bilinear.x.DN_X);
```

```
                  15  % [init_kinematics,dim,...
                  16  %       xelem,DNX]          =  KinematicsFunctio
                  17  %                                     solution
                  18  %                                     fem.volu
                  19  % init_kinematics         =  KinematicsFunctio
                  20  %                                     xelem,DN
  3.08    60000   21  kinematics        =  KinematicsFunctionVolum
          60000   22                                      solution.x
          60000   23                                      fem.volume
                  24  %---------------------------------------------
                  25  % Determine if linear elasticity or nonlinear
                  26  % on the current element
                  27  %---------------------------------------------
  0.17    60000   28  if mat_info.optimisation.rho(ielem)<0.9
< 0.01    60000   29     LE_flag                  =  1;
  0.40    60000   30     Identity                 =  repmat(eye(geometry
  0.84    60000   31     u                        =  solution.x.Eulerian
          60000   32                                 solution.x.Lagrangi
  0.09    60000   33     kinematics.F             =  Identity;
  0.05    60000   34     kinematics.H             =  Identity;
  0.18    60000   35     kinematics.J             =  ones(ngauss,1);
                  36  else
                  37     LE_flag                  =  0;
                  38  end
                  39  %---------------------------------------------
                  40  % First and second derivatives of the model f
                  41  %---------------------------------------------
  5.81    60000   42  mat_info                    =  GetDerivativesMode
          60000   43
                  44  %---------------------------------------------
                  45  % First and second derivatives of the model f
                  46  %---------------------------------------------
  1.41    60000   47  mat_info_void               =  GetDerivativesMode
          60000   48
                  49  %---------------------------------------------
                  50  % Sum both contributions (rho^qDWDF_solid +
                  51  %---------------------------------------------
  5.57    60000   52  mat_info.derivatives        =  SumDerivativesU(ma
          60000   53
                  54  %---------------------------------------------
                  55  % First Piola-Kirchhoff stress tensor.
                  56  %---------------------------------------------
                  57  % Piola                     =  FirstPiolaKirchho
                  58  %
                  59  %
onVolumeCInitial(geometry.dim,...
n.x.Eulerian_x(:,mesh.volume.x.connectivity(:,ielem)),...
ume.bilinear.x.DN_X);
onVolumeCMex(init_kinematics,dim,...
NX);
eCMex (kinematics,geometry.dim,...
x.Eulerian_x(:,mesh.volume.x.connectivity(:,ielem)),...
e.bilinear.x.DN_X);
-------------------------------
r elasticity shall be applied

-------------------------------



y.dim),1,1,ngauss);
n_x(:,mesh.volume.x.connectivity(:,ielem)) - ...
ian_X(:,mesh.volume.x.connectivity(:,ielem));








-------------------------------
for the solid
-------------------------------
lMechanics (ielem,geometry.dim,ngauss,kinematics.F,kinematics.H,kine
             mat_info,mat_info.material_model{mat_info.material_iden
-------------------------------
for the void
-------------------------------
lMechanics (ielem,geometry.dim,ngauss,kinematics.F,kinematics.H,kine
      mat_info_void,mat_info_void.material_model);
-------------------------------
(1 - rho^q)*DWDF_void)
-------------------------------
at_info.derivatives,mat_info_void.derivatives,...
 mat_info.optimisation.rho(ielem),mat_info.optimisation.penal,1);
-------------------------------

-------------------------------
offStressTensorU(ngauss,geometry.dim,kinematics.F,kinematics.H,...
                      mat_info.derivatives.DU.DUDF,...
                      mat_info.derivatives.DU.DUDH,...
```

```
ematics.J,...
tifier(ielem)});



ematics.J,...
```

```
                    60  %
                    61  % Piola_vectorised       =  Matrix2Vector(geo
  2.30    60000     62  mat_info.Piola          =  FirstPiolaKirchh
          60000     63
          60000     64
          60000     65
  0.91    60000     66  Piola_vectorised      =  Matrix2Vector(geom
                    67  %---------------------------------------------
                    68  % Matrix BF
                    69  %---------------------------------------------
  1.55    60000     70  BF                    =  BMatrix(ngauss,geo
          60000     71                              fem.volume.bilir
          60000     72                              vectorisation.Bx
          60000     73                              vectorisation.Bx
                    74  %---------------------------------------------
                    75  % Q matrices arising from the linearisation o
                    76  %---------------------------------------------
  0.61    60000     77  QF                    =  QMatrixComputation
  0.59    60000     78  QSigmaH               =  QMatrixComputation
  0.03    60000     79  if geometry.dim==2
  0.03    60000     80     QSigmaH            =  QSigmaH*0;
< 0.01    60000     81  end
                    82  %---------------------------------------------
                    83  % Integration weights
                    84  %---------------------------------------------
  0.47    60000     85  IntWeight             =  quadrature.volume.k
< 0.01    60000     86  if LE_flag
                    87     %------------------------------------------
                    88     %------------------------------------------
                    89     % Residuals and Stiffness matrices for lir
                    90     %------------------------------------------
                    91     %------------------------------------------
  0.01    60000     92     for igauss=1:ngauss
                    93         %--------------------------------------
                    94         % Vectorisation of stiffness matrices
                    95         %--------------------------------------
 15.99   240000     96         vect_mat      =  VectorisedStiffnes
         240000     97
         240000     98
         240000     99
         240000    100
                   101         %--------------------------------------
                   102         % Residual and stiffness matrices
                   103         %--------------------------------------
  1.04   240000    104         asmb.Tx       =  asmb.Tx    +  (vect_
                      mat_info.derivatives.DU.DUDJ);
ometry.dim^2,ngauss,Piola);
offStressTensorUCMex (mat_info.Piola,ngauss,geometry.dim,kinematics.
                   mat_info.derivatives.DU.DUDF,...
                   mat_info.derivatives.DU.DUDH,...
                   mat_info.derivatives.DU.DUDJ);
netry.dim^2,ngauss,mat_info.Piola);
-------------------------------

-------------------------------
ometry.dim,mesh.volume.x.n_node_elem,...
near.x.DN_X,...
x_matrix.LHS_indices,...
x_matrix.RHS_indices);
-------------------------------
of H: DH[].SigmaH = Q*DF[]. and
-------------------------------
 (kinematics.F,geometry.dim,ngauss);
 (mat_info.derivatives.DU.DUDH,geometry.dim,ngauss);



-------------------------------

-------------------------------
bilinear.W_v.*fem.volume.bilinear.x.DX_chi_Jacobian;

-------------------------------
-------------------------------
nearelasticity regions
-------------------------------
-------------------------------

-------------------------------

-------------------------------
sMatricesU (igauss,BF(:,:,igauss),...
   mat_info.derivatives.D2U,...
   mat_info.derivatives.DU,...
   QF(:,:,igauss),QSigmaH(:,:,igauss),...
   kinematics.H(:,:,igauss));
-------------------------------

-------------------------------
_mat.Kxx*u(:))*IntWeight(igauss);
```

```
.F,kinematics.H,...
```

```
0.48   240000   105           asmb.Kxx         =  asmb.Kxx   +  vect_r
0.60   240000   106       end
                107
                108   else
                109       %-----------------------------------------
                110       %-----------------------------------------
                111       % Residuals and Stiffness matrices for no
                112       %-----------------------------------------
                113       %-----------------------------------------
                114       for igauss=1:ngauss
                115           %-------------------------------------
                116           % Residual conservation of linear mon
                117           %-------------------------------------
                118           asmb.Tx          =  asmb.Tx   +  (BF(:,:
                119           %-------------------------------------
                120           % Vectorisation of stiffness matrices
                121           %-------------------------------------
                122           vect_mat         =  VectorisedStiffness
                123                                           mat_i
                124                                           mat_i
                125                                           QF(:,
                126                                           kiner
                127           %-------------------------------------
                128           % Stiffness matrices
                129           %-------------------------------------
                130           asmb.Kxx         =  asmb.Kxx   +  vect_r
                131       end
                132   end
```

Other subfunctions in this file are not included in this listing.

```
nat.Kxx*IntWeight(igauss);



--------------------------------
--------------------------------
onlinear elasticity regions
--------------------------------
--------------------------------

--------------------------------
nentum.
--------------------------------
:,igauss)'*Piola_vectorised(:,igauss))*IntWeight(igauss);
--------------------------------
s
--------------------------------
sMatricesU(igauss,BF(:,:,igauss),...
info.derivatives.D2U,...
info.derivatives.DU,...
,:,igauss),QSigmaH(:,:,igauss),...
natics.H(:,:,igauss));
--------------------------------

--------------------------------
nat.Kxx*IntWeight(igauss);
```