icoSolver (Calls: 1, Time: 24357.272 s)

Generated 09-Jun-2018 11:32:22 using performance time.
function in file
C:\Users\Nerv33\Desktop\project\unstructured\solver\icoSolver.m
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
init_script	script	1

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Tim Plo
222	p = Tri_p\b_p;	22514	11175.152 s	45.9%	
96	COV = sparse(o_offdiag,n_offdi...	22514	2752.305 s	11.3%	
75	Jf = ParallelCrossDiffusionTer...	22515	1031.361 s	4.2%	
114	DDT = spdiags(vol/dt,0,ncells,...	22514	1015.026 s	4.2%	
135	[U.y,~,RELRES,ITER] = bicgstab...	22514	907.466 s	3.7%	
All other lines			7475.962 s	30.7%	
Totals			24357.272 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time
bicgstab	function	45028	1740.114 s	7.1%
ParallelCrossDiffusionTerm	function	22515	1013.344 s	4.2%

ParallelSurfNormFlux	function	22514	750.760 s	3.1%
spdiags	function	22514	491.780 s	2.0%
ParallelCell2Verts	function	22515	110.576 s	0.5%
title	function	225	6.831 s	0.0%
ColorBar.doUpdate	function	225	4.511 s	0.0%
reconstructOperator	function	1	3.467 s	0.0%
diffusionOperator	function	1	1.551 s	0.0%
icoSolver>mag	subfunction	225	0.266 s	0.0%
colorbar	function	1	0.155 s	0.0%
cell2nodeinterpOperator	function	1	0.111 s	0.0%
xlabel	function	1	0.087 s	0.0%
clf	function	1	0.060 s	0.0%
num2str	function	225	0.045 s	0.0%
axis	function	1	0.045 s	0.0%
ylabel	function	1	0.014 s	0.0%
...Manager>AxesLayoutManager.doUpdate	class method	1	0.007 s	0.0%
view	function	1	0.006 s	0.0%
Self time (built-ins, overhead, etc.)			20233.543 s	83.1%
Totals			24357.272 s	100%

Code Analyzer results

Line number	Message
15	The value assigned to variable 'nbfaces' might be unused.
33	The value assigned here to 'WALL' appears to be unused. Consider replacing it by ~.

34	The value assigned here to 'MOVINGWALL' appears to be unused. Consider replacing it by ~.
148	The variable 'rAU' appears to change size on every loop iteration. Consider preallocating for speed.
185	The value assigned to variable 'L_p' might be unused.
199	This sparse indexing expression is likely to be slow.
199	This sparse indexing expression is likely to be slow.
205	This sparse indexing expression is likely to be slow.
206	This sparse indexing expression is likely to be slow.
207	This sparse indexing expression is likely to be slow.

Coverage results

Show coverage for parent directory

Total lines in function	302
Non-code lines (comments, blank lines)	144
Code lines (lines that can run)	158
Code lines that did run	132
Code lines that did not run	26
Coverage (did run/can run)	83.54 %

Function listing

Color highlight code according to code analyzer

```
   time      Calls    line
                        1  function icoSolver(IterSettings,PhysicalProp
                        2  %% Author: Yuxuan Liu
                        3  %% Read Data
                        4
                        5  [...
 < 0.01          1      6      autosave_period, ...
                 1      7      plot_period, ...
                 1      8      dt, ...
                 1      9      tf...
                 1     10      ] =   IterSettings{:};
                       11
                       12  %% Read Data
 < 0.01          1     13  ncells = mesh.ncells;
```

```
< 0.01          1   14 nfaces = mesh.nfaces;

< 0.01          1   15 nbfaces = mesh.nbfaces;

< 0.01          1   16 verts = mesh.verts;
< 0.01          1   17 vol = mesh.vol;
< 0.01          1   18 areaf = mesh.areaf;
< 0.01          1   19 link_bface_to_face = mesh.link_bface_to_face,
< 0.01          1   20 owner = mesh.owner;
< 0.01          1   21 neighbour = mesh.neighbour;
< 0.01          1   22 is_bface = mesh.is_bface;
                    23
                    24 %% Read Data
                    25
                    26 [...
< 0.01          1   27     nu,...
                1   28     U,...
                1   29     p,...
                1   30     phi,...
                1   31     INLET,...
                1   32     OUTLET,...

                1   33     WALL, ...


                1   34     MOVINGWALL,...

                1   35     setRef...
                1   36     ] = PhysicalProperties{:};
                    37
                    38 %% new feature
< 0.01          1   39 fprintf('Start computation ...\n')
< 0.01          1   40 tic
  1.55          1   41 [diffOperator,crossDiffOperator] = diffusior
  0.11          1   42 nodeInterpOperator = cell2nodeinterpOperator
  3.47          1   43 leastSquareOperator = reconstructOperator(me
< 0.01          1   44 toc
                    45
                    46 %% Iteration start point
< 0.01          1   47 loop_start_flag = 1;
```

```
                    48
< 0.01          1   49 t = dt:dt:tf;
< 0.01          1   50 stepnum = size(t,2);
                    51
                    52 %% Read saved data
                    53
< 0.01          1   54 if nargin > 3
                    55     autosave = load (SaveData);
                    56     U = autosave.('U');
                    57     p = autosave.('p');
                    58     phi = autosave.('phi');
                    59     clear savedata
                    60 end
                    61
< 0.01          1   62 ts = tic;
                    63 %% Inner Loop start point
< 0.01          1   64 for i = 1:stepnum
                    65     %% Assign value for history result
                    66
  5.98      22515   67     U.old.x = U.x;
  6.20      22515   68     U.old.y = U.y;
  6.03      22515   69     p_old = p;
                    70
                    71     %% PISO
                    72     % Compute for cross-diffusion term
                    73
 120.90     22515   74     U_vert = ParallelCell2Verts(U,nodeInterp
1031.36     22515   75     Jf = ParallelCrossDiffusionTerm(U_vert,
                    76
                    77     %% Update matrix
                    78     % Ax = b
                    79     % construct A
                    80     %CrankNicolson = 0.45; % 0 ~ 0.5
                    81     % To improve performance, consider splitt
                    82
< 0.01      22514   83     if loop_start_flag
  0.01          1   84         o_diag = [owner(~is_bface);owner(~is_
  0.01          1   85         n_diag = [neighbour(~is_bface);neighb
  0.01          1   86         o_offdiag = [owner(~is_bface) ;neighb
  0.01          1   87         n_offdiag = [neighbour(~is_bface);own
                    88
```

```
                    89          % construct visc term
  0.12          1   90          VISC_o = sparse(o_diag,o_offdiag,[di
  0.11          1   91          VISC_n = sparse(n_diag,n_offdiag,[di
  0.01          1   92          VISC = VISC_o + VISC_n; clear VISC_o
< 0.01          1   93      end
                    94
                    95      % construct convection term
2752.30     22514   96      COV = sparse(o_offdiag,n_offdiag,[phi(~i
                    97
 270.26     22514   98      Tri = COV + VISC;
                    99
                   100      % construct b
 561.78     22514  101      b_u = accumarray(o_offdiag,[-Jf.x(~is_bf
 562.08     22514  102      b_v = accumarray(o_offdiag,[-Jf.y(~is_bf
                   103
                   104      % correct boundary conditions
   0.01     22514  105      ifc = link_bface_to_face;
   0.31     22514  106      is_dirchlet = U.boundary.dirchlet;
   0.05     22514  107      is_neumann = U.boundary.neumann;
                   108
 276.96     22514  109      Tri = Tri + sparse(owner(ifc),owner(ifc),
   3.26     22514  110      b_u(owner(ifc)) = b_u(owner(ifc)) - (U.bc
   2.44     22514  111      b_v(owner(ifc)) = b_v(owner(ifc)) - (U.bc
                   112
                   113      % (Implicit)dt & dpidxi
1015.03     22514  114      DDT = spdiags(vol/dt,0,ncells,ncells) -
  64.66     22514  115      b_u = b_u + DDT*U.x;
  65.20     22514  116      b_v = b_v + DDT*U.y;
 280.62     22514  117      Tri = DDT + Tri;
                   118
                   119      % (Linear multistep)store old flux
                   120      %if loop_start_flag,f.x = b_u - Tri*U.x;
                   121      %f_0 = f;f.x = b_u - Tri*U.x;f.y = b_v -
                   122
                   123      %% Solve U equation
                   124
                   125      % direct solve
                   126      %U.x = Tri\b_u;U.y = Tri\b_v;
                   127
                   128      %Linear multistep
                   129      %U.x = U.x + dt*(1.5*f.x - 0.5*f_0.x)./vo
```

```
                    130
                    131       % pbicgstab matrix solver
567.31      22514   132       [il,iu] = ilu(Tri);
834.43      22514   133       [U.x,~,RELRES,ITER] = bicgstab(Tri,b_u,1
  3.12      22514   134       fprintf('DILUPBiCGStab:Solving Ux ,residu
907.47      22514   135       [U.y,~,RELRES,ITER] = bicgstab(Tri,b_v,1
  3.16      22514   136       fprintf('DILUPBiCGStab:Solving Uy ,residu
                    137
                    138       %% Inner loop
< 0.01      22514   139       final_inner_iter_flag = 1;
< 0.01      22514   140       for inner_iter_flag = 1:final_inner_iter_
                    141
                    142           %% Prepare to solve pressure correct
                    143
< 0.01      22514   144           if loop_start_flag
                    145
< 0.01          1   146               for icell = 1:ncells
                    147                   %rAU(icell,1) = vol(icell,1),

  0.03     249001   148                   rAU(icell,1) = dt;

  0.01     249001   149               end
                    150
< 0.01          1   151               rAUf = zeros(nfaces,1);
  0.04          1   152               rAUf(~is_bface) = 0.5*(rAU(owner
< 0.01          1   153           end
                    154
                    155           %% Update guessed surface normal flu
                    156
759.98      22514   157           phi_guess = ParallelSurfNormFlux(U,
                    158
                    159           %% Adjust phi
                    160
< 0.01      22514   161           if ~isempty(INLET)
                    162               inflow = link_bface_to_face(INLE
                    163               outflow = link_bface_to_face(OUT
                    164               if abs(sum(phi(outflow))) > 1e-0
                    165                   phi(outflow) = -phi(outflow)
                    166               end
                    167               if any(phi(outflow) < 0)
```

```
                    168                 phi(outflow) = min(0,phi(out
                    169                 fprintf('Reversed flow appea
                    170             end
                    171         end
                    172         %fprintf('Adjust outflow, mass outfl
                    173         %\n',sum(phi(outflow))); %debug
                    174
                    175         %% Build pressure correction equatio
                    176         % Ax = b
                    177         % construct A
  0.01      22514   178         if loop_start_flag
  0.12          1   179             gradp_o = sparse(o_diag,o_offdia
  0.12          1   180             gradp_n = sparse(n_diag,n_offdia
                    181
  0.01          1   182             Tri_p = gradp_o + gradp_n; clear
                    183
                    184             %alpha = max(sum(abs(Tri_p),2)./
  7.50          1   185             L_p = ichol(Tri_p, struct('type'
< 0.01          1   186         end
                    187
                    188         % construct b
733.89      22514   189         b_p = accumarray([o_offdiag;owner(~~
                    190
                    191         %% correct p boundary condition
                    192
  0.01      22514   193         if ~isempty(INLET)
                    194             if loop_start_flag
                    195                 cell_outflow = owner(link_bf
                    196             end
                    197
                    198             for icell_o = cell_outflow
                    199                 if loop_start_flag,Tri_p(ice
                    200                 b_p(icell_o,1) = 0;
                    201             end
                    202
< 0.01      22514   203         else
```

```
                    204
      0.03      22514   205              if loop_start_flag,Tri_p(setRef.
    < 0.01          1   206                  Tri_p(:,setRef.refCell) = 0;
    < 0.01          1   207                  Tri_p(setRef.refCell,setRef.
    < 0.01          1   208              end
                    209
      0.07      22514   210              b_p(setRef.refCell) = 0;
                    211
    < 0.01      22514   212          end
                    213
                    214          %% Solve pressure correction equatio
                    215
                    216          % Preconditioner DIC
                    217          %if loop_start_flag&&isempty(INLET)
                    218          %    L_p = ichol(Tri_p, struct('type
                    219          %end
                    220
                    221          % direct solve
  11175.15      22514   222          p = Tri_p\b_p;
                    223
                    224          % PCG solver
                    225          %tol = [0.01 1e-06];
                    226          %[p,~,RELRES,ITER] = pcg(Tri_p,b_p,t
                    227          %fprintf('DICPCG:Solving p ,residual
                    228
                    229          %% Correct phi
                    230
    792.60      22514   231          phi(~is_bface) = phi_guess(~is_bface
                22514   232              - rAUf(~is_bface).*(p(neighbour(
                    233
     62.40      22514   234          phi(~~is_bface) = phi_guess(~~is_bfa
                    235
                    236          %% Interpolate for pressure correcti
                    237
                    238          %dp = zeros(nfaces,1);
                    239          %dp(~is_bface) = ;%.*areaf(~is_bface
```

```
                240
636.63     22514  241          weighted_dp = mesh.lf_weighted(~is_b
114.20     22514  242          weighted_dp = repmat(weighted_dp,2,1
                243
                244
411.44     22514  245          leastSquareCorrector = [accumarray(o
           22514  246                                  accumarray(o
  0.07     22514  247          leastSquareCorrector = leastSquareCo
                248
                249
 63.64     22514  250          dpdx = leastSquareOperator.x*leastSq
 62.88     22514  251          dpdy = leastSquareOperator.y*leastSq
                252
                253          %% Update p,U
                254
 17.98     22514  255          U.x = U.x - rAU.*dpdx;
 15.65     22514  256          U.y = U.y - rAU.*dpdy;
                257          %p = p - p(setRef.refCell) + setRef.
                258
                259
< 0.01     22514  260      end
                261
                262      %% Residual
                263      %u_change = sqrt(sum((U.x-U.old.x).^2)/nc
                264      %v_change = sqrt(sum((U.y-U.old.y).^2)/nc
 15.79     22514  265      p_change = sqrt(sum((p-p_old).^2)/ncells)
                266
                267      % Residual Monitor
< 0.01     22514  268      if loop_start_flag
  0.10         1  269          clf,h = figure(1);
< 0.01         1  270          set(h,'Units','pixels','Position',[1
  1.15         1  271          TRI = delaunay(verts(:,1),verts(:,2))
  0.02         1  272          obj_1 = patch('Faces',TRI,'Vertices',
                273          %obj_1 = scatter(verts(:,1),verts(:,2
                274          %hold on, [drawx,drawy] = drawPolygon
  0.31         1  275          colorbar,xlabel('X'),ylabel('Y'),axi
< 0.01         1  276          if ~exist('image_output','dir'), mkdi
< 0.01         1  277      end
                278
  0.02     22514  279      if mod(i,plot_period) == 0||i == stepnum
  0.84       225  280          obj_1.CData = feval(@mag,U_vert.x,U_
```

6.88	225	**281**	title(['Umag, current time: ', num2st
47.27	225	**282**	drawnow
		283	%saveas(h,['.\image_output\result'
0.02	225	**284**	fprintf('Current figure has been save
< 0.01	225	**285**	end
		286	
3.24	22514	**287**	fprintf('Current time: %f, pressure relT
		288	
		289	% autosave
0.01	22514	**290**	if mod(i,autosave_period) == 0||i == step
80.72	225	**291**	save('autosave.mat','U','p','phi')
0.02	225	**292**	fprintf('Current computation has been
< 0.01	225	**293**	end
		294	
		295	%% Something else
< 0.01	22514	**296**	if loop_start_flag, loop_start_flag = 0;
< 0.01	22514	**297**	end
		298	telapsed = toc(ts);
		299	fprintf([num2str(telapsed),' sec elasped for
		300	
		301	
		302	end

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