SFDTD2Dfun (Calls: 2400, Time: 30.530 s)

Generated 16-May-2017 09:57:53 using performance time.
function in file C:\Gits\IndiEngiSchola\Matlab\SFDTD\SFDTD2Dfun.m
Copy to new window for comparing multiple runs

Refresh

☑ Show parent functions ☑ Show busy lines ☑ Show child functi
☑ Show Code Analyzer results ☑ Show file coverage ☑ Show function list

Parents (calling functions)

Function Name	Function Type	Calls
SFDTD2Dtesting	script	2400

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time
8	`ux(i,i1) = ux(i,i1) - uCx*(p(i...`	17079859	2.657 s	8.7%
47	`p(i,i1) = p(i,i1) - pCx*(ux(i,...`	17277992	2.646 s	8.7%
48	`- pCy*(uy(i+1, i1) - uy(i, i1)...`	17277992	2.531 s	8.3%
16	`uy(i,i1) = uy(i,i1) - uCx*(p(i...`	17079859	2.523 s	8.3%
46	`if(idx(i, i1) > 0)`	17669044	2.371 s	7.8%
All other lines			17.803 s	58.3%
Totals			30.530 s	100%

Children (called functions)
No children

ons

ting

Time Plot





Code Analyzer results
No Code Analyzer messages.

Coverage results
Show coverage for parent directory

Total lines in function	56
Non-code lines (comments, blank lines)	17
Code lines (lines that can run)	39
Code lines that did run	39
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Function listing
Color highlight code according to time

time	Calls	line	
		1	function [p, ux, uy] = SFDTD2Dfun(p, pCx, pCy,
		2	
0.43	2400	3	if(size(idx(idx > 0),1) > 10)
		4	% mat(col, row)
< 0.01	2389	5	for i = 1 : size(ux,1)-1
0.04	205454	6	for i1 = 2 : size(ux,2)-2
2.26	17463590	7	if(idx(i, i1) > 0)
2.66	17079859	8	ux(i,i1) = ux(i,i1) - uCx*(p(i,
1.99	17079859	9	end
2.07	17463590	10	end
0.03	205454	11	end
		12	
< 0.01	2389	13	for i = 2 : size(uy,1)-2
0.03	203065	14	for i1 = 1 : size(uy,2)-1
2.19	17463590	15	if(idx(i, i1) > 0)
2.52	17079859	16	uy(i,i1) = uy(i,i1) - uCx*(p(i,
2.00	17079859	17	end
1.96	17463590	18	end
0.03	203065	19	end
< 0.01	11	20	else
		21	% update the non-boundary condition nodes :
< 0.01	11	22	ux(:, 2:end-1) = ux(:, 2:end-1) - uCx*(p(:,
0.01	11	23	uy(2:end-1, :) = uy(2:end-1, :) - uCy*(p(2
< 0.01	2400	24	end
		25	

```
ux, uy, uCx, uCy, Rx, Ry, ZL, ZR, ZT, ZB, idx)
```

```
,i1)-p(i,i1-1));
```

```
,i1)-p(i-1,i1));
```

```
for velocity
, 2:end) - p(:, 1:end-1));
:end, :) - p(1:end-1, :));
```

```
                    26  % update the velocity at the right wall
  0.02      2400    27  ux(:, end) = ((Rx - ZR)/(Rx + ZR))*ux(:, end)
            2400    28      + (2/(Rx + ZR))*p(:, end);
                    29
                    30  %update the velocity at the left wall
< 0.01      2400    31  ux(:, 1) = ((Rx - ZL)/(Rx + ZL))*ux(:, 1) - (2/
                    32
                    33  %update the velocity at the top wall
  0.01      2400    34  uy(end, :) = ((Ry - ZT)/(Ry + ZT))*uy(end, :)
            2400    35      + (2/(Ry + ZT))*p(end, :);
                    36
                    37  %update the velocity at the bottom wall
< 0.01      2400    38  uy(1, :) = ((Ry - ZB)/(Ry + ZB))*uy(1, :) - (2/
                    39
                    40  % update the pressure at all nodes
                    41  %     p(idx > 0) = p(idx > 0) - pCx*(ux(idx3 >
                    42  %         - pCy*(uy(idx2 > 0) - uy(idx > 0));
  0.47      2400    43  if(size(idx(idx > 0),1) > 10)
< 0.01      2389    44      for i = 1 : size(p,1)-1
  0.04    205454    45          for i1 = 1 : size(p,2)-1
  2.37  17669044    46              if(idx(i, i1) > 0)
  5.18  17277992    47                  p(i,i1) = p(i,i1) - pCx*(ux(i,
        17277992    48                      - pCy*(uy(i+1, i1) - uy(i,
  1.99  17277992    49              end
  2.09  17669044    50          end
  0.03    205454    51      end
< 0.01        11    52  else
< 0.01        11    53      p = p - pCx*(ux(:, 2:end) - ux(:, 1:end-1)/
              11    54          - pCy*(uy(2:end, :) - uy(1:end-1, :));
< 0.01      2400    55  end
  0.03      2400    56  end
```

```
...


/(Rx + ZL))*p(:, 1);


...


/(Ry + ZB))*p(1, :);


 0) - ux(idx > 0))...




i1 + 1) - ux(i, i1))...
i1));



)...
```