FDTD3Dfun (Calls: 5999, Time: 97.085 s)

Generated 06-Oct-2017 10:19:27 using performance time.
function in file C:\Gits\IndiEngiSchola\Matlab\FDTD\FDTD3Dfun.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
FDTD3Dtesting	script	5999

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
53	`ux(:, 2:end-1, :) = ux(:, 2:en...`	5999	21.329 s	22.0%	
54	`uy(2:end-1, :, :) = uy(2:end-1...`	5999	20.567 s	21.2%	
55	`uz(:, :, 2:end-1) = uz(:, :, 2...`	5999	19.576 s	20.2%	
89	`- pCy*(uy(2:end, :, :) - uy(1:...`	5999	11.942 s	12.3%	
90	`- pCz*(uz(:, :, 2:end) - uz(:,...`	5999	11.090 s	11.4%	
All other lines			12.581 s	13.0%	
Totals			97.085 s	100%	

y, pCz, ux, uy, uz, uCx,...
zN, ZzP)
ethod for acoustic simulation.

ıce calculations on
y. This function assumes
 solved, and so assumes that
re no cross-terms. This
dary conditions, using the
 normalised aproximation of

ı in x direction
ı in y direction
ı in z direction
ı in x direction
ı in y direction
ı in z direction

Children (called functions)
No children

Code Analyzer results
No Code Analyzer messages.

Coverage results
Show coverage for parent directory

Total lines in function	79
Non-code lines (comments, blank lines)	60
Code lines (lines that can run)	19
Code lines that did run	19
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
                  13  function [p, ux, uy, uz] = FDTD3Dfun(p, pCx, pCy
                  14      uCy, uCz, Rx, Ry, Rz, ZxN, ZxP, ZyN, ZyP, Zz
                  15  % Function that performs one timestep of FDTD me
                  16  %
                  17  % This function performs central finite differer
                  18  % matricies that represent pressure and velocity
                  19  % that a linear acoustic wave equation is being
                  20  % the velocity terms are orthoganal and there ar
                  21  % function solves empirical semi-absorbing bounc
                  22  % acoustic impedance of the boundary based on a
                  23  % absorption coefficient.
                  24  %
                  25  % Takes the following arguments:
                  26  % p = N:N:N matrix of pressure values
                  27  % ux = N:N+1:N matrix of velocity values
                  28  % uy = N+1:N:N matrix of velocity values
                  29  % uz = N:N:N+1 matrix of velocity values
                  30  % pCx = constant related to pressure calculatior
                  31  % pCy = constant related to pressure calculatior
                  32  % pCz = constant related to pressure calculatior
                  33  % uCx = constant related to velocity calculatior
                  34  % uCy = constant related to velocity calculatior
                  35  % uCz = constant related to velocity calculatior
```

```
ield constants
ield constants
ield constants
 -x direction
 +x direction
 -y direction
 +y direction
 -z direction
 +z direction

ity field matricies


 to velocity field
ep excluding the boundarys
pressure
 direction
(p(:, 2:end,:) - p(:, 1:end-1, :));
*(p(2:end, :, :) - p(1:end-1, :, :));
*(p(:, :, 2:end) - p(:, :, 1:end-1));

ndary
d z = time and space step
on * current velocity values
al pressure value
 1, :)...


ndary
:, end, :) ...


ndary
 :, :)...


ndary
end, :, :) ...


ndary
```

```
                36  % Rx = (rho0*dx)/(0.5*dt) Constant related to f
                37  % Ry = (rho0*dy)/(0.5*dt) Constant related to f
                38  % Rz = (rho0*dz)/(0.5*dt) Constant related to f
                39  % ZxN = acoutsitc impedance term at boundary in
                40  % ZxP = acoutsitc impedance term at boundary in
                41  % ZyN = acoutsitc impedance term at boundary in
                42  % ZyP = acoutsitc impedance term at boundary in
                43  % ZzN = acoutsitc impedance term at boundary in
                44  % ZzP = acoutsitc impedance term at boundary in
                45  %
                46  % This functions returns the pressure and veloc
                47  %
                48
                49      % Calculate central difference aproximation
                50      % Velocity in a direction at current timest
                51      % = velocity 1 time step ago - constants * p
                52      % differential half a time step ago in that
21.33     5999  53      ux(:, 2:end-1, :) = ux(:, 2:end-1,:) - uCx*
20.57     5999  54      uy(2:end-1, :, :) = uy(2:end-1, :, :) - uCy
19.58     5999  55      uz(:, :, 2:end-1) = uz(:, :, 2:end-1) - uCz
                56
                57      % update the velocity at the negative x bou
                58      % Velocity at this boundary for all of y an
                59      % normalised by the lovel impedance conditi
                60      % - 2 / time and space discretization * loca
 0.33     5999  61      ux(:, 1, :) = ((Rx - ZxN)/(Rx + ZxN))*ux(:,
          5999  62          - (2/(Rx + ZxN))*p(:, 1, :);
                63
                64      % update the velocity at the positive x bou
 0.18     5999  65      ux(:, end, :) = ((Rx - ZxP)/(Rx + ZxP))*ux(
          5999  66          + (2/(Rx + ZxP))*p(:, end, :);
                67
                68      % update the velocity at the negative y bou
 0.63     5999  69      uy(1, :, :) = ((Ry - ZyN)/(Ry + ZyN))*uy(1,
          5999  70          - (2/(Ry + ZyN))*p(1, :, :);
                71
                72      % update the velocity at the positive y bou
 0.26     5999  73      uy(end, :, :) = ((Ry - ZyP)/(Ry + ZyP))*uy(e
          5999  74          + (2/(Ry + ZyP))*p(end, :, :);
                75
                76      % update the velocity at the negative z bou
```

```
 :, 1)...


ıdary
:, :, end)...



ɔss domain 1 time step ago -
ıtral difference of
ıree dimensions
l, :))...
 :))...
-1));
```

```
 0.32    5999    77      uz(:, :, 1) = ((Rz - ZzN)/(Rz + ZzN))*uz(:,
         5999    78          -(2/(Rz + ZzN))*p(:, :, 1);
                 79
                 80      % update the velocity at the positive z boun
 0.26    5999    81      uz(:, :, end) = ((Rz - ZzP)/(Rz + ZzP))*uz(
         5999    82          +(2/(Rz + ZzP))*p(:, :, end);
                 83
                 84      % update the pressure at all nodes
                 85      % new pressure across domain = pressure acro
                 86      % (space,time and wave speed constant) * cen
                 87      % velocities half a time step  ago in all th
33.51    5999    88      p = p - pCx*(ux(:, 2:end, :) - ux(:, 1:end-1
         5999    89          - pCy*(uy(2:end, :, :) - uy(1:end-1, :,
         5999    90          - pCz*(uz(:, :, 2:end) - uz(:, :, 1:end-
 0.05    5999    91  end
```