FDTD3Dfun (Calls: 1200, Time: 104.840 s)

Generated 16-May-2017 09:36:17 using performance time.
function in file C:\Gits\IndiEngiSchola\Matlab\FDTD\FDTD3Dfun.m
Copy to new window for comparing multiple runs

Refresh

- [x] Show parent functions
- [x] Show busy lines
- [x] Show child functions
- [x] Show Code Analyzer results
- [x] Show file coverage
- [x] Show function listing

Parents (calling functions)

Function Name	Function Type	Calls
FDTD3Dtesting	script	1200

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
42	`uy(2:end-1, :, :) = uy(2:end-1...`	1200	23.719 s	22.6%	
41	`ux(:, 2:end-1, :) = ux(:, 2:en...`	1200	22.182 s	21.2%	
43	`uz(:, :, 2:end-1) = uz(:, :, 2...`	1200	20.263 s	19.3%	
78	`- pCz*(uz(:, :, 2:end) - uz(:,...`	1200	13.211 s	12.6%	
77	`- pCy*(uy(2:end, :, :) - uy(1:...`	1200	13.122 s	12.5%	
All other lines			12.344 s	11.8%	
Totals			104.840 s	100%	

Children (called functions)
No children

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

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











n in x direction
n in y direction
n in z direction
n in x direction
n in y direction
n in z direction
ield constants
ield constants
```

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

```
function [p, ux, uy, uz] = FDTD3Dfun(p, pCx, pC
    uCy, uCz, Rx, Ry, Rz, ZxN, ZxP, ZyN, ZyP, Z
% Function that performs one timestep of FDTD me
%
% This function performs central finite differe
% matricies that represent pressure and velocit
% that a linear acoustic wave equation is being
% the velocity terms are orthoganal and there a
% function solves empirical semi-absorbing boun
% acoustic impedance of the boundary based on a
% absorption coefficient.
%
% Takes the following arguments:
% p = N:N:N matrix of pressure values
% ux = N:N+1:N matrix of velocity values
% uy = N+1:N:N matrix of velocity values
% uz = N:N:N+1 matrix of velocity values
% pCx = constant related to pressure calculatio
% pCy = constant related to pressure calculatio
% pCz = constant related to pressure calculatio
% uCx = constant related to velocity calculatio
% uCy = constant related to velocity calculatio
% uCz = constant related to velocity calculatio
% Rx = (rho0*dx)/(0.5*dt) Constant related to f
% Ry = (rho0*dy)/(0.5*dt) Constant related to f
```

```
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
 :, 1)...
```

```
% Rz = (rho0*dz)/(0.5*dt) Constant related to f:
% ZxN = acoutsitc impedance term at boundary in
% ZxP = acoutsitc impedance term at boundary in
% ZyN = acoutsitc impedance term at boundary in
% ZyP = acoutsitc impedance term at boundary in
% ZzN = acoutsitc impedance term at boundary in
% ZzP = acoutsitc impedance term at boundary in
%
% This functions returns the pressure and veloc:
%

    % Calculate central difference aproximation
    % Velocity in a direction at current timeste
    % = velocity 1 time step ago - constants * p
    % differential half a time step ago in that
22.18   1200    ux(:, 2:end-1, :) = ux(:, 2:end-1,:) - uCx*
23.72   1200    uy(2:end-1, :, :) = uy(2:end-1, :, :) - uCy*
20.26   1200    uz(:, :, 2:end-1) = uz(:, :, 2:end-1) - uCz*

    % update the velocity at the negative x bour
    % Velocity at this boundary for all of y and
    % normalised by the lovel impedance conditic
    % - 2 / time and space discretization * loca
 0.22   1200    ux(:, 1, :) = ((Rx - ZxN)/(Rx + ZxN))*ux(:,
        1200        - (2/(Rx + ZxN))*p(:, 1, :);

    % update the velocity at the positive x bour
 0.16   1200    ux(:, end, :) = ((Rx - ZxP)/(Rx + ZxP))*ux(
        1200        + (2/(Rx + ZxP))*p(:, end, :);

    % update the velocity at the negative y bour
 0.51   1200    uy(1, :, :) = ((Ry - ZyN)/(Ry + ZyN))*uy(1,
        1200        - (2/(Ry + ZyN))*p(1, :, :);

    % update the velocity at the positive y bour
 0.36   1200    uy(end, :, :) = ((Ry - ZyP)/(Ry + ZyP))*uy(e
        1200        + (2/(Ry + ZyP))*p(end, :, :);

    % update the velocity at the negative z bour
 0.20   1200    uz(:, :, 1) = ((Rz - ZzN)/(Rz + ZzN))*uz(:,
        1200        -(2/(Rz + ZzN))*p(:, :, 1);
```

```
ndary
:, :, end)...


oss domain 1 time step ago -
ntral difference of
hree dimensions
1, :))...
 :))...
-1));
```

```
                      
                    % update the velocity at the positive z bour
  0.18    1200      uz(:, :, end) = ((Rz - ZzP)/(Rz + ZzP))*uz(
          1200          +(2/(Rz + ZzP))*p(:, :, end);

                    % update the pressure at all nodes
                    % new pressure across domain = pressure acro
                    % (space,time and wave speed constant) * cer
                    % velocities half a time step  ago in all th
 37.00    1200      p = p - pCx*(ux(:, 2:end, :) - ux(:, 1:end-1
          1200          - pCy*(uy(2:end, :, :) - uy(1:end-1, :,
          1200          - pCz*(uz(:, :, 2:end) - uz(:, :, 1:end-
  0.02    1200  end
```