nonlinearQuadrotorDynamics (Calls: 936156, Time: 41.116 s)

Generated 09-May-2019 04:13:21 using performance time.
function in file /Users/tonirv/Google Drive (arosinol@mit.edu)/PhD MIT/Courses/16.32 Principles of Optimal Control and Estimation/Project/MIT16.32_Optimal_Control_And_Estimation_project/src/nonlinearQuadrotorDynamics.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
continuous	function	936156

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
3	`global Quad;`	936156	4.684 s	11.4%	
39	`Quad.Dynamics.r_dot = (State.p...`	936156	4.515 s	11.0%	
21	`Quad.Dynamics.X_ddot = ((C_phi...`	936156	3.550 s	8.6%	
23	`Quad.Dynamics.Z_ddot = ((C_phi...`	936156	3.175 s	7.7%	
16	`Quad.Dynamics.X_dot = State.X_...`	936156	3.167 s	7.7%	
All other lines			22.025 s	53.6%	
Totals			41.116 s	100%	

Children (called functions)
No children

Code Analyzer results
No Code Analyzer messages.

Coverage results
Show coverage for parent directory

Total lines in function	39
Non-code lines (comments, blank lines)	17
Code lines (lines that can run)	22
Code lines that did run	22
Code lines that did not run	0

```
_psi) .* Control.U1 - Quad.Kdx .* State.X_dot) ./ Quad.m;
_phi) .* Control.U1 - Quad.Kdy .* State.Y_dot) ./ Quad.m;
      .* Control.U1 - Quad.Kdz .* State.Z_dot) ./ Quad.m - Quad.g;


a) + State.q .* (S_phi .* T_theta);

./ C_theta) .* State.r;


system for

) ... %- Quad.Jp*State.q*Quad.Obar

) ... %+ Quad.Jp*State.p*Quad.Obar

) + Control.U4) ./ Quad.Jz;
```

Coverage (did run/can run)	100.00 %

Function listing

Color highlight code according to time

time	Calls	line	
		1	function nonlinearQuadrotorDynamics(State, Control)
		2	
4.684	936156	3	global Quad;
		4	
		5	%% Cache calculations.
0.680	936156	6	C_phi = cos(State.phi);
0.488	936156	7	S_phi = sin(State.phi);
0.334	936156	8	C_psi = cos(State.psi);
0.331	936156	9	S_psi = sin(State.psi);
0.346	936156	10	C_theta = cos(State.theta);
0.345	936156	11	S_theta = sin(State.theta);
0.348	936156	12	T_theta = tan(State.theta);
		13	
		14	%% Linear Velocities
		15	% Encoded in the quad state:
3.167	936156	16	Quad.Dynamics.X_dot = State.X_dot;
1.577	936156	17	Quad.Dynamics.Y_dot = State.Y_dot;
1.816	936156	18	Quad.Dynamics.Z_dot = State.Z_dot;
		19	
		20	%% Linear Accelerations
3.550	936156	21	Quad.Dynamics.X_ddot = ((C_phi .* S_theta .* C_psi + S_phi .* S_
2.990	936156	22	Quad.Dynamics.Y_ddot = ((C_phi .* S_psi .* S_theta - C_psi .* S_
3.175	936156	23	Quad.Dynamics.Z_ddot = ((C_phi .* C_theta)
		24	
		25	%% Angular Velocities.
		26	% World frame.
2.136	936156	27	Quad.Dynamics.phi_dot = State.p + State.r .* (C_phi .* T_theta
1.986	936156	28	Quad.Dynamics.theta_dot = C_phi .* State.q - S_phi .* State.r;
2.080	936156	29	Quad.Dynamics.psi_dot = (S_phi./C_theta) .* State.q + (C_phi .
		30	
		31	%% Angular Accelerations
		32	% Body frame.
		33	% Ignoring for now the Obar because I have to solve the linear s
		34	% the control inputs... This is as if Jp was 0.
2.974	936156	35	Quad.Dynamics.p_dot = (State.q .* State.r .* (Quad.Jy - Quad.Jz)
	936156	36	+ Control.U2) ./ Quad.Jx;
2.685	936156	37	Quad.Dynamics.q_dot = (State.p .* State.r .* (Quad.Jz - Quad.Jx)
	936156	38	+ Control.U3)/Quad.Jy;
4.515	936156	39	Quad.Dynamics.r_dot = (State.p .* State.q .* (Quad.Jx - Quad.Jy)

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