

Contents

- [Implementation](#)
- [Adaptive controller](#)
- [Calculate dx based on the dynamics of the robot and the error](#)

```
function [ dx ] = odellinkTracking_adaptive( t,x,param )
```

Implementation

```
%Extracting the coefficients of the trajectory
a1=param(1,:);

%Extract the approximated dynamics
alpha=x(3:5,:);

%Create the actual trajectory
vec_t = [1; t; t^2; t^3];
theta_d= [a1*vec_t];

% compute the velocity and acceleration in both theta 1 and theta2.
a1_vel = [a1(2), 2*a1(3), 3*a1(4), 0];
a1_acc = [2*a1(3), 6*a1(4),0,0 ];

% compute the desired trajectory (assuming 3rd order polynomials for trajectories)
dtheta_d =[a1_vel*vec_t];
ddtheta_d =[a1_acc*vec_t];
theta= x(1);
dtheta= x(2);
```

Not enough input arguments.

Error in odellinkTracking_adaptive (line 5)
a1=param(1,:);

Adaptive controller

True dynamics of the system

```
I=7.2;
mgd=1;
fv=1;
alpha_actual=[mgd;fv;I];

% Assign the gain values
KP=250;
KD=0.05;
K=[0 1;-KP -KD];
```

```
% Calculate the alpha_cap_dot
gamma=[1 0 0;0 1 0;0 0 1];
P=[1 0;0 1];
phi=[sin(theta_d) dtheta_d ddtheta_d];
alpha_cap_dot=-inv(gamma)*transpose(phi)*[0 1]*P*[theta-theta_d;dtheta-dtheta_d];
```

Calculate dx based on the dynamics of the robot and the error

```
de=[0;ddtheta_d]-K*[theta-theta_d;dtheta-dtheta_d]+[0;1]*phi*(alpha-alpha_actual);
dx=[de;alpha_cap_dot];
```

```
end
```