**Quadcopter sumulation**

* **Data**

1. X=[3xN double] **The initial frame axes**
2. omega= [3x1 double]; **Initial Angular velocity vector**
3. T= [3x1 double]; **Thrust**
4. Input= [4xN double]
5. t = [1xN double]; **Simulation times, in seconds**
6. Dt= 0.0050
7. x = zeros(3, N);
8. xdot= zeros(3, N);**Initial Linear Velocity**
9. theta= zeros(3, N); **initial Body axes.**
10. thetadot = zeros(3, N); **Derivatives of roll, pitch, yaw.**
11. inputs=3; **Given Current Inputs: Angular velocity of the 𝑖𝑡ℎ rotor.**
12. **R**=[3x1 double];**Rotation transformation matrix**
13. **Fd**=[3x1 double];**Global drag forces**
14. a =[3x1 double];**Acceleration**
15. omegadot =[3x1 double];**Angular acceleration**
16. tau =[3x1 double];**Torques**
17. 𝜙= Roll angle. 𝜃=Pitch angle. 𝜓= Yaw angle.

* **Physical constants**.

1. g = 9.81; **Acceleration due to gravity**
2. m = 0.5; **Mass of quadcopter**
3. L = 0.25; **Length between the quadcopter center and any propellers**
4. k = 3e-6; **Thrust Coefficient**
5. b = 1e-7; **Drag coefficient**
6. I = [5e-3, 5e-3, 10e-3]; **Inertia matrix,** **Inertial tensor of the quadcopter**
7. kd = 0.25; **Global drag coefficient**
8. deviation = 100; **Deviation in the angular velocity. Deviation is in degrees/sec**.

* **Functions**

1. **omega = thetadot2omega(thetadot, theta);**

***(Compute angular velocity. Convert derivatives of roll, pitch, yaw to omega.)***

* phi = theta(1);
* theta\_ = theta(2);
* psi = theta(3);

1. **T = thrust(inputs, k)**

***(Compute thrust.)***

1. **a = acceleration(inputs, theta, xdot, m, g, k, kd)**

***(Compute acceleration.)***

* phi = theta(3);
* theta = theta (2);
* psi = theta (1);

1. **tau = torques(inputs, L, b, k)**

***(Compute torques.)***

1. **omegadot = angular acceleration** **(inputs, omega, I, L, b, k)**

***(Compute angular acceleration.)***

1. **thetadot = omega2thetadot(omega, theta)**

***(Compute roll, pitch, yaw derivatives.*** ***Convert omega to roll, pitch, yaw derivatives)***

* phi = theta(1);
* theta\_ = theta(2);
* psi = theta(3);

**MY C++ program**

#include "stdafx.h"

#include "stdio.h"

#include <stdlib.h>

#include <conio.h>

#include <iostream>

#include <vector>

#include <time.h>

#include <math.h>

using namespace std;

vector<vector<double>> acceleration(double inputs,vector<double>angle,vector<double>xdot\_,double m,double g,double k,double kd);

vector<vector<double>> angular\_acceleration(int j,vector<vector<double>>omega,vector<vector<double>>I,double L,double b,double k);

//vector<vector<double>> torques(vector<double>timevector,int j,double L,double b,double k);

void createvector(vector<double>&myvec);

vector<vector<double>> thetadot2omega(vector<double>thetadot,vector<double>angle);

vector<vector<double>>omega2thetadot(vector<vector<double>>omega,vector<double>angle);

int main()

{

srand( time(0));

double start\_time=0.0,end\_time=10.0, dt=0.005;

vector<double>timevector;

createvector(timevector);

int N=0;

for(int i=0;i<timevector.size();i++)

{

++N;

}

vector<double>x(2,0);

x.push\_back(10);

vector<double>xdot(3,0);

vector<double>theta(3,0);

double randnumber;

int deviation =100,temp;

double pi=3.142;

vector<double>thetadot;

for(int i=0;i<3;++i)

{

randnumber=(double) rand()/ RAND\_MAX;

temp=(2\*deviation\*randnumber-deviation);

thetadot.push\_back((temp\*pi)/180);

//cout<<thetadot[i];

};

double i,g=9.81,m=0.5,L=0.25,k=3e-6,b=1e-7,kd=0.25;

vector<vector<double>>I(3,vector<double>(3));

I[0][0]=5e-3;

I[0][1]=0;

I[0][2]=0;

I[1][0]=0;

I[1][1]=5e-3;

I[1][2]=0;

I[2][0]=0;

I[2][1]=0;

I[2][2]=10e-3;

int j;

for ( j=0;j<timevector.size();j++)

{

i=timevector[j];

vector<vector<double>>omega=thetadot2omega(thetadot,theta);

vector<vector<double>>a=acceleration(i,theta,xdot,m,g,k,kd);

vector<vector<double>>omegadot=angular\_acceleration(j,omega,I,L,b,k);

omega[0][0]=omega[0][0]+dt\*omegadot[0][0];

omega[1][0]=omega[1][0]+dt\*omegadot[1][0];

omega[2][0]=omega[2][0]+dt\*omegadot[2][0];

vector<vector<double>>thetadot=omega2thetadot(omega,theta);

theta[0]=theta[0]+dt\*thetadot[0][0];

theta[1]=theta[1]+dt\*thetadot[1][0];

theta[2]=theta[2]+dt\*thetadot[2][0];

/\*for (int h=0;h<3;h++)

cout<<j<<" a "<<a[h][0]<<"\n";\*/

xdot[0]=xdot[0]+dt\*a[0][0];

xdot[1]=xdot[1]+dt\*a[1][0];

xdot[2]=xdot[2]+dt\*a[2][0];

/\*for (int h=0;h<3;h++)

cout<<j<<" x "<<x[h]<<" xdot "<<xdot[h]<<"\n\n";\*/

x[0]=x[0]+dt\*xdot[0];

x[1]=x[1]+dt\*xdot[1];

x[2]=x[2]+dt\*xdot[2];

/\*for (int n=0;n<3;n++)

cout<<j<<" theta "<<theta[n]<<"\n";\*/

for (int h=0;h<3;h++)

cout<<j<<" x "<<x[h]<<"\n";

cout<<"\n";

}

cout<<endl;

/\*for (int i=0;i<3;i++)

for (int j=0;j<1;j++)

cout<<a[i][j]<<" ";\*/

cout<<endl;

\_getch();

return 0;

}

void createvector(vector<double>& myvec)

{

double start\_time=0.0,end\_time=10.0, dt=0.005;

myvec.push\_back(start\_time);

while(start\_time<=end\_time)

{

start\_time+=dt;

myvec.push\_back(start\_time);

}

}

vector<vector<double>>thetadot2omega(vector<double>thetadot,vector<double>angle)

{

double phi = angle[0],theta\_ = angle[1], psi = angle[2];

vector<vector<double>>w(3,vector<double>(3));

vector<vector<double>>omega(3,vector<double>(1));

w[0][0]=1;

w[0][1]=0;

w[0][2]=-sin(theta\_);

w[1][0]=0;

w[1][1]=cos(phi);

w[1][2]=cos(theta\_)\*sin(phi);

w[2][0]=0;

w[2][1]=-sin(phi);

w[2][2]=cos(theta\_)\*cos(phi);

for (int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

omega[i][j]=0;

for(int k=0;k<3;k++)

{

omega[i][j]=omega[i][j]+w[i][k]\*thetadot[k];

}

//cout << "omega "<<omega[i][j]<<"\n ";

}

}

return omega;

}

vector<vector<double>> acceleration(double inputs,vector<double>angle,vector<double>xdot\_,double m,double g,double k,double kd)

{

double phi = angle[2],theta\_ = angle[1], psi = angle[0];

vector<vector<double>>R(3,vector<double>(3));

R[0][0]=cos(phi)\*cos(theta\_);

R[1][0]=cos(theta\_)\*sin(phi);

R[2][0]=-sin(theta\_);

R[0][1]=cos(phi) \* sin(theta\_) \* sin(psi) - cos(psi) \* sin(phi);

R[1][1]=cos(phi) \* cos(psi) + sin(phi) \* sin(theta\_) \* sin(psi);

R[2][1]=cos(theta\_)\*sin(phi);

R[0][2]=sin(phi) \* sin(psi) + cos(phi) \* cos(psi) \* sin(theta\_);

R[1][2]=cos(psi) \* sin(phi) \* sin(theta\_) - cos(phi) \* sin(psi);

R[2][2]=cos(theta\_)\*cos(phi);

vector<double>timevector;

createvector(timevector);

double sumtime=0;

for(int i=0;i<2001;i++)

{

sumtime+=timevector[i];

}

vector<vector<double>>T(3,vector<double>(1));

T[0][0]=0;

T[1][0]=0;

T[2][0]=k\*12e+6;

vector<vector<double>>gravity(3,vector<double>(1));

gravity[0][0]=0;

gravity[1][0]=0;

gravity[2][0]=-g;

vector<vector<double>>T\_(3,vector<double>(1));

for (int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

T\_[i][j]=0;

for(int k=0;k<3;k++)

{

T\_[i][j]=T\_[i][j]+R[i][k]\*T[k][j];

}

}

}

vector<vector<double>>FD(3,vector<double>(1));

vector<vector<double>>T2(3,vector<double>(1));

vector<vector<double>>a(3,vector<double>(1));

for(int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

FD[i][j]=-kd\*xdot\_[i];

T2[i][j]=(1/m)\*T\_[i][j];

a[i][j]=gravity[i][j]+T2[i][j]+FD[i][j];

}

};

return a;

}

/\*vector<vector<double>> torques(vector<double>timevector,int j,double L,double b,double k)

{

vector<vector<double>>tau(3,vector<double>(1));

tau[0][0]=L\*k\*(timevector[j]-timevector[j+2]);

tau[1][0]=L\*k\*(timevector[j+1]-timevector[j+3]);

tau[2][0]=b\*(timevector[j]-timevector[j+1]+timevector[j+2]-timevector[j+3]);

return tau;

}\*/

vector<vector<double>> angular\_acceleration(int j,vector<vector<double>>omega,vector<vector<double>>I,double L,double b,double k)

{

double start\_time=0.0,end\_time=10.0, dt=0.005;

int j\_=0;

vector<double>timevector;

timevector.push\_back(start\_time);

while(start\_time<=end\_time)

{

start\_time+=dt;

timevector.push\_back(start\_time);

}

j\_=j;

vector<vector<double>>tau(3,vector<double>(1));

tau[0][0]=L\*k\*((3e+6)-(3e+6));

tau[1][0]=L\*k\*((3e+6)-(3e+6));

tau[2][0]=b\*((3e+6)-(3e+6)+(3e+6)-(3e+6));

//vector<vector<double>>tau=torques(timevector,j,L,b,k);

vector<vector<double>>omega\_(3,vector<double>(1));

vector<vector<double>>omega2(3,vector<double>(1));

for (int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

omega\_[i][j]=0;

for(int k=0;k<3;k++)

{

omega\_[i][j]=omega\_[i][j]+I[i][k]\*omega[k][j];

omega2[i][j]=omega2[i][j]+omega\_[k][j]\*omega[k][j];

}

}

}

vector<vector<double>>invI(3,vector<double>(3));

float det=0;

for(int i=0;i<3;i++)

{

det=det+(I[0][i]\*(I[1][(i+1)%3]\*I[2][(i+2)%3]-I[1][(i+2)%3]\*I[2][(i+1)%3]));

}

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

invI[j][i]=((I[(i+1)%3][(j+1)%3]\*I[(i+2)%3][(j+2)%3])-(I[(i+1)%3][(j+2)%3]\*I[(i+2)%3][(j+1)%3]))/det;

}

}

vector<vector<double>>omegadot2(3,vector<double>(1));

omegadot2[0][0]=tau[0][0]-omega2[0][0];

omegadot2[1][0]=tau[1][0]-omega2[1][0];

omegadot2[2][0]=tau[2][0]-omega2[2][0];

vector<vector<double>>omegadot(3,vector<double>(1));

for (int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

omegadot[i][j]=0;

for(int k=0;k<3;k++)

{

omegadot[i][j]=omegadot[i][j]+invI[i][k]\*omegadot2[k][j];

}

}

}

return omegadot;

}

vector<vector<double>>omega2thetadot(vector<vector<double>>omega,vector<double>angle)

{

double phi = angle[0],theta\_ = angle[1], psi = angle[2];

vector<vector<double>>w(3,vector<double>(3));

w[0][0]=1;

w[0][1]=0;

w[0][2]=-sin(theta\_);

w[1][0]=0;

w[1][1]=cos(phi);

w[1][2]=cos(theta\_)\*sin(phi);

w[2][0]=0;

w[2][1]=-sin(phi);

w[2][2]=cos(theta\_)\*cos(phi);

vector<vector<double>>invw(3,vector<double>(3));

float det=0;

for(int i=0;i<3;i++)

{

det=det+(w[0][i]\*(w[1][(i+1)%3]\*w[2][(i+2)%3]-w[1][(i+2)%3]\*w[2][(i+1)%3]));

}

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

invw[j][i]=((w[(i+1)%3][(j+1)%3]\*w[(i+2)%3][(j+2)%3])-(w[(i+1)%3][(j+2)%3]\*w[(i+2)%3][(j+1)%3]))/det;

}

}

vector<vector<double>>thetadot(3,vector<double>(1));

for (int i=0;i<3;i++)

{

for(int j=0;j<1;j++)

{

thetadot[i][j]=0;

for(int k=0;k<3;k++)

{

thetadot[i][j]=thetadot[i][j]+invw[i][k]\*omega[k][j];

}

//cout << "omega "<<omega[i][j]<<"\n ";

}

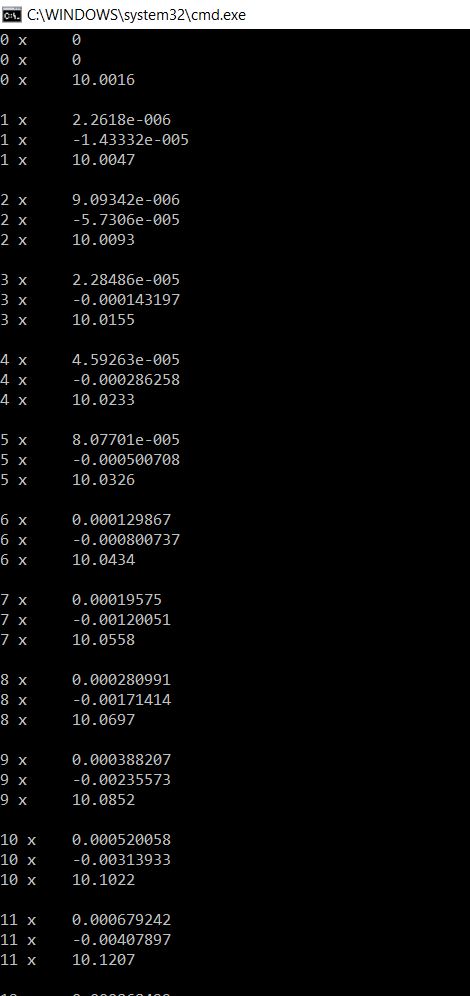
}

return thetadot;

}

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Description** | **Value** | **Units** |
| **Inputs(1)** | **Current inputs:**  **Angular velocity of the 𝑖𝑡ℎ rotor** | **3e+6** | **Rad/s** |
| **Inputs(2)** | **Current inputs:**  **Angular velocity of the 𝑖𝑡ℎ rotor** | **3e+6** | **Rad/s** |
| **Inputs(3)** | **Current inputs:**  **Angular velocity of the 𝑖𝑡ℎ rotor** | **3e+6** | **Rad/s** |
| **Inputs(4)** | **Current inputs:**  **Angular velocity of the 𝑖𝑡ℎ rotor** | **3e+6** | **Rad/s** |

**1. TABLE I: Quadcopter parameters**

****