PD Attitude Control

Abhishek Avadhanam

Table of Contents

Table of Contents 2
List of Figures 4
1 Root System 5
1.1 Introduction 5
2 Variables used 6
3 Assumptions 7
4 Subsystems 8
4.1 Init 8
4.1.1 Init Argument Summary 8
4.1.2 Init Function Script 8
4.2 Propagation 10
4.2.1 Propagation Argument Summary 10
4.2.2 Propagation Function Script 10
4.3 Onboard_temp 13
4.3.1 Hierarchy 13
4.3.2 Onboard_temp Input Summary 13
4.3.3 Onboard_temp Output Summary 14
4.3.4 Input (to Controller) 15
4.3.4.1 "Input" Input Summary 15
4.3.4.2 "Input" Output Summary 15
4.3.4.3 MATLAB Function Argument Summary 16
4.3.4.4 MATLAB Script 16
4.3.5 Onboard_actual (controller) 17
4.3.5.1 Onboard_actual Input Summary 17
4.3.5.2 Onboard_actual Output Summary 17
4.3.5.3 quat to (rot_vec)*theta 18
4.3.5.3.1 Introduction: 18
4.3.5.3.2 quat to (rot_vec)*theta Input Summary 18
4.3.5.3.3 quat to (rot_vec)*theta Output Summary 18
4.3.5.4 PD 19
4.3.5.4.1 PD Input Summary 19
4.3.5.4.2 PD Output Summary 19
4.3.5.1 Torque_lim 20
4.3.5.1.1 RW torque lim Argument Summary 20
4.3.5.1.2 RW torque lim Function Script 20
4.4 Graphs 21
4.4.1 Graphs Input Summary 21
5 PD gain tuning logic. 22
6 Results 23
6.1 Results Expected by Pixxel Aerospace 23
6.1.1 Attitude vs Time 23
6.1.1.1 Euler Angles vs Time 23

6.1.1.2 Quaternion vs Time 23
6.1.2 Angular velocity of satellite vs Time 24
6.1.3 Angular Momentum of satellite vs Time 24
6.1.4 Control Torque vs Time 25
6.2 For Desired Angular vel is zero 25
6.2.1 Quaternion vs Time 25
6.3 For Desired Angular vel is zero + Disturbance due to Angular momentum is zero 25
6.3.1 Quaternion vs Time 26
7 Inference 27
8 Future Scope 27
References 29

List of Figures

Figure 1-1 Root System 5
Figure 2-1 Beta-Angle, Fig 2 from Ref 2........ 7
Figure 3-1 Onboard_temp........ 13
Figure 3-2 Input to Controller........ 15
Figure 3-3 Onboard_actual (controller) 17
Figure 3-4 quat to (rot_vec)*theta 18
Figure 3-5 PD controller........ 19
Figure 3-6 Graphs 21
Figure 4-1 PD gains tuning logic (Fig 4 from Ref 1)........ 22

1 Root System

1.1 Introduction

1) *Init:* Outputs the satellite's initial and desired final states and constants used in the simulation.
2) *Onboard_temp:* Receives initial, current and desired final state of the satellite from *Init* and *Propagation* and outputs the control torque applied and the current state of the satellite to *Propagation* and *Graphs*.
   a) *Input:* Checks if the input received from the *Propagation* is valid and sets the simulation back to the initial state defined in *Init* if not.
   b) *Onboard_actual:* The onboard logic calculates the control torque and Reaction Wheels' angular-velocity change. This includes calculating the Attitude and angular velocity errors using the current state received from *Input* and the desired final state from *Init.*
3) *Propagation:* Receives relevant constants from *Init;* the control torques applied and the current state of the satellite from *Onboard_temp*; and outputs state of the satellite after one time-step to *Onboard_temp* and *Graphs*. This includes rk4 propagation of angular velocity and the Attitude of the satellite (Eq 1 and Eq 2 of reference 2).
4) *Graphs:* Receives the current state of the satellite and the torques applied on it from *Propagation* and *Onboard_temp*, then graphs relevant variables as a function of time.

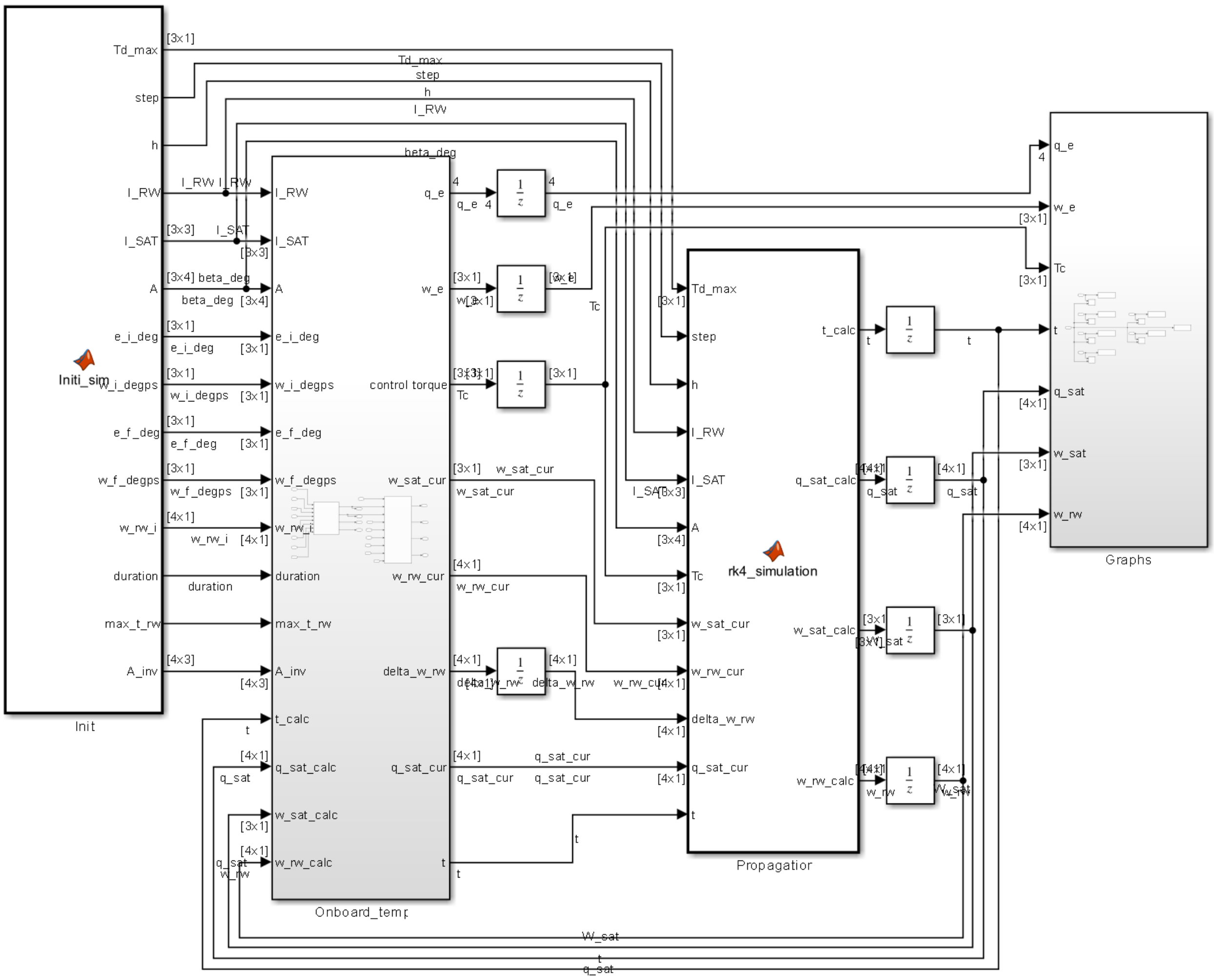


Figure 1-1 Root System

2 Variables used

Variable	Stands for
A	Mapping between Body axis and Reaction Wheels (Eq. 4, Ref 2).
A_inv	Pseudo inverse of A.
I_RW	Moment of inertia of Reaction Wheel (kg m^2).
I_SAT	Moment of Inertia of Satellite (kg m^2)
Td_max	The maximum Disturbance torque on the satellite (Nm).
duration	The duration of the simulation (sec).
e_f_deg	Desired Final Attitude (deg).
e_i_deg	Initial Attitude in (deg).
h	Step for rk4 propagation of Attitude and angular velocity. (sec).
max_t_rw	Maximum Torque per reaction wheel (Nm).
step	Step size for the controller (1 sec).
w_f_degps	Desired Final angular velocity (deg/s).
w_i_degps	Initial Angular velocity (deg/s).
w_rw_i	Initial Angular velocities of each reaction wheel (rad/s).
q_sat_calc	Calculated Attitude after rk4 propagation.
t_calc	Current time after propagation.
w_rw_calc	The calculated reaction wheel angular velocity after rk4 propagation (rad/sec).
w_sat_calc	The calculated satellite angular velocity after rk4 propagation (rad/sec).
q_e	Quaternion Error for the current time step.
w_e	Angular velocity error for current time step (rad/s).
control torque	Control torque to be applied on satellite.
w_sat_cur	Current satellite angular velocity (rad/sec).
w_rw_cur	Current reaction wheel angular velocities (rad/sec).
delta_w_rw	Change in angular velocity in the current time step (rad/sec).
q_sat_cur	The current orientation of the satellite.
t	Current time (sec).

3 Assumptions

1) Beta-Angle (*beta_ang_rad* in *Init*)= sin^(-1)(0.25)
   a) In case of a single reaction wheel failure, the failure axis (either x or y axes) and the z-axis have 0.01125 Nm available as max Torque, and the non-failure axis's max torque=0.0225 Nm.
   b) This has not been optimised yet but can be worked on further.

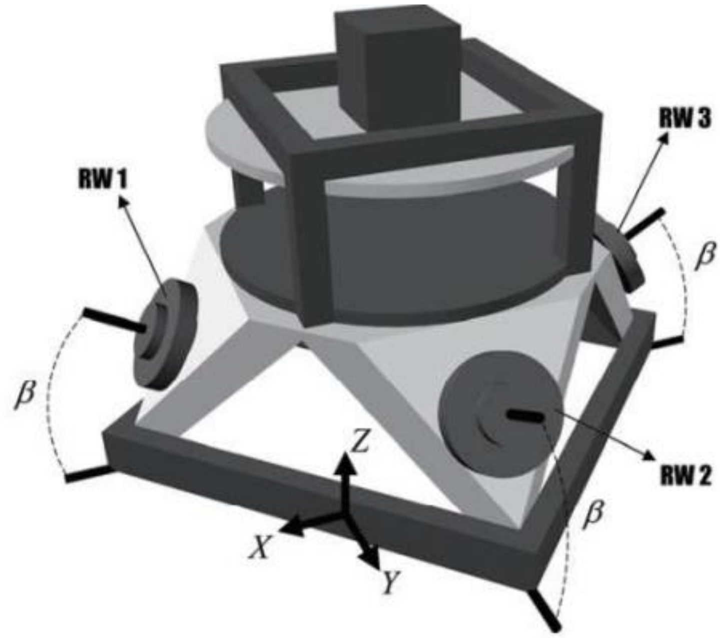


Figure 3-1 Beta-Angle, Fig 2 from Ref 2.

2) Initial angular momentum per reaction wheel (*H_rw_i* in *Init)* = 0.035 Nms (body frame).
   a) This determines the initial angular velocity of the reaction wheels.
3) Discrete PD controller's sample time (*step* in *Init*) = 1 s
4) Attitude and angular velocity propagator's step size (*h* in *Init*)= 0.00125 sec.
5) The initial and the desired final orientations (*e_i_deg* and *e_f_deg* respectively in *Init*) are given in Euler Angles (X, Y, Z).

4 Subsystems

4.1 Init

4.1.1 Init Argument Summary

Name	Scope	Port	Data Type	Size
A	Output	6	double	[3, 4]
A_inv	Output	14	double	[4, 3]
I_RW	Output	4	double	1
I_SAT	Output	5	double	[3, 3]
Td_max	Output	1	double	[3, 1]
duration	Output	12	double	1
e_f_deg	Output	9	double	[3, 1]
e_i_deg	Output	7	double	[3, 1]
h	Output	3	double	1
max_t_rw	Output	13	double	1
step	Output	2	double	1
w_f_degps	Output	10	double	[3, 1]
w_i_degps	Output	8	double	[3, 1]
w_rw_i	Output	11	double	[4, 1]

4.1.2 Init Function Script

```
function [Td_max,step,h,I_RW,
I_SAT,A,e_i_deg,w_i_degps,e_f_deg,w_f_degps,w_rw_i,duration,max_t_rw,A_inv]=
Initi_sim

  %beta_angle_deg=30; % Beta-Angle of reaction wheels in Degrees
  beta_angle_rad=asin(0.25); % Beta-Angle of reaction wheels in Radians
  cB=cos(beta_angle_rad); %cos(beta)
  sB=sin(beta_angle_rad);%sin(beta)
  A=[cB, 0, -cB, 0; 0, cB, 0,-cB; sB, sB, sB, sB]; % Tc=A*Ti, where Tc is control torque (pid),
Ti is Torque by individual reaction wheels.
  A_inv= pinv(A); %pseudo inverse of A

  e_f_deg=[30.00;50.00;70.00]; %Euler angles in degrees of final Attitude (XYZ)
  e_i_deg=[0.00;0.00;0.00]; %Euler angles in degrees of initial Attitude (XYZ)
  w_f_degps=[0.20;0.20;0.20];% Final angular vel of satellite (deg/s)
  w_i_degps=[15.00;-15.00;15.00];% Initial Angular vel of satellite (deg/s)
```

```
    I_SAT=[2.10 0.00 0.01;0.00 2.30 -0.03;0.01 -0.03 1.72]; %Moment of Inertia of satellite
    I_RW=0.00042; %Moment of inertia of individual reaction wheels
    Td_max=[10.00^(-6);10.00^(-6);10.00^(-6)]; %max disturbance torque in Nm
    step=1; %step size of the control loop in seconds
    H_rw_i=0.035;%Angular momentum of individual reaction wheel (Nms)
    w_rw_i=H_rw_i/I_RW*[1.00;1.00;1.00;1.00]; %initial angular vel of reaction wheels (rad/s)
    h=0.00125; %step size of propagation (angular vel and quaternions)
    duration=86400.0000; %duration of simulation in seconds
    max_t_rw=0.015; %max torque per reaction wheel Nm
end
```

4.2 Propagation

4.2.1 Propagation Argument Summary

Name	Scope	Port	Data Type	Size
A	Input	6	double	[3, 4]
I_RW	Input	4	double	1
I_SAT	Input	5	double	[3, 3]
Tc	Input	7	double	[3, 1]
Td_max	Input	1	double	[3, 1]
delta_w_rw	Input	10	double	[4, 1]
h	Input	3	double	1
q_sat_cur	Input	11	double	[4, 1]
step	Input	2	double	1
t	Input	12	double	1
w_rw_cur	Input	9	double	[4, 1]
w_sat_cur	Input	8	double	[3, 1]
q_sat_calc	Output	2	double	[4, 1]
t_calc	Output	1	double	1
w_rw_calc	Output	4	double	[4, 1]
w_sat_calc	Output	3	double	[3, 1]

4.2.2 Propagation Function Script

```
function [t_calc,q_sat_calc, w_sat_calc, w_rw_calc] = rk4_simulation(Td_max,step,
h,I_RW,I_SAT,A,Tc, w_sat_cur, w_rw_cur, delta_w_rw,q_sat_cur,t)
    Iinv=pinv(I_SAT); %inverse of Moment of inertia of satellite
    flag=0; % toggle between initialised and calculated values in functions, had some
compilation error.

    t_calc=-5.00; %initialzed time output to negative, just a sanity check to make sure the output
of this function only goes into pid if initialisations are done right

    %output initialised, compilation error
    q_sat_calc=[1;0;0;0]; %attitude of satellite
    w_sat_calc=[0;0;0]; %angular vel of satellite (rad/s)
    w_rw_calc=[0;0;0;0]; %angular vel of reaction wheel (rad/s)
    h_rw_cur=[0.00;0.00;0.00]; %current angular momentum of reaction wheel (Nms)
```

```
    delta_h_rw=[0.00;0.00;0.00]; % total change in angular momentum of reaction wheel (Nms)

    if t>=0.00
        flag=1;
        h_rw_cur=A*(I_RW*w_rw_cur);
        delta_h_rw=A*(I_RW*delta_w_rw);

        Td_temp=[0;Td_max*sin(t)]; %Disturbance torque in inertial frame changing with time
        %frame conversion from inertial frame to body frame
        Td_temp=quatmultiply(quatmultiply(q_sat_cur,Td_temp,flag),[q_sat_cur(1,1);-
1*q_sat_cur(2,1);-1*q_sat_cur(3,1);-1*q_sat_cur(4,1)],flag);
        Td=[Td_temp(2,1);Td_temp(3,1);Td_temp(4,1)]; %distuarbance torque in body frame

        for k = 0:h:step
            % Compute h_rw
            h_rw = h_rw_cur + k * delta_h_rw;

            % RK4 method for angular velocity
            a = funcEval_dwdt(I_SAT, Iinv, Tc, Td, w_sat_cur, h_rw, h,flag);
            temp = w_sat_cur + a / 2;

            b = funcEval_dwdt(I_SAT, Iinv, Tc, Td, temp, h_rw, h,flag);
            temp = w_sat_cur + b / 2;

            c = funcEval_dwdt(I_SAT, Iinv, Tc, Td, temp, h_rw, h,flag);
            temp = w_sat_cur + c;

            d = funcEval_dwdt(I_SAT, Iinv, Tc, Td, temp, h_rw, h,flag);

            w_sat_cur = w_sat_cur + (a / 6 + b / 3 + c / 3 + d / 6);

            % RK4 method for quaternion
            w_sat_cur_quat = [0;w_sat_cur];

            k1 = qb2i(q_sat_cur, w_sat_cur_quat, h,flag);
            temp2 = q_sat_cur + k1 / 2;

            k2 = qb2i(temp2, w_sat_cur_quat, h,flag);
            temp2 = q_sat_cur + k2 / 2;

            k3 = qb2i(temp2, w_sat_cur_quat, h,flag);
            temp2 = q_sat_cur + k3;

            k4 = qb2i(temp2, w_sat_cur_quat, h,flag);

            q_sat_cur = q_sat_cur + (k1/6 + k2/3 + k3/3 + k4/6);

q_sat_cur=q_sat_cur/((q_sat_cur(1,1)^2+q_sat_cur(2,1)^2+q_sat_cur(3,1)^2+q_sat_cur(4,1)^2)
^0.5);
        end

        t_calc=t+step;
        q_sat_calc=q_sat_cur;
        w_sat_calc=w_sat_cur;
        w_rw_calc=w_rw_cur+delta_w_rw;
```

```
    end

end


function w_sat_next = funcEval_dwdt(I_SAT, Iinv, Tc, Td, w_sat_cur, h_rw_cur,h,flag)
    w_sat_next=[0;0;0]; %initialized, compilation error

    if flag==1
        H = (I_SAT * w_sat_cur) + (h_rw_cur); %total angular momentum of satellite + reaction
wheel
        d = cross(w_sat_cur, H);
        T = Tc + Td - d;
        w_sat_next = Iinv * T * h;
    end
end

function q = qb2i(y, w, h,flag)
    q=[1;0;0;0];%initialized, compilation error

    if flag==1
        % Quaternion update based on angular velocity
        q = quatmultiply(w, y,flag) * (h/2);
    end
end

function q = quatmultiply(q1, q2,flag)
    q=[1;0;0;0];%initialized, compilation error

    if flag==1
        % Multiply two quaternions
        q(1,1) = q1(1)*q2(1) - q1(2)*q2(2) - q1(3)*q2(3) - q1(4)*q2(4);
        q(2,1) = q1(1)*q2(2) + q1(2)*q2(1) - q1(3)*q2(4) + q1(4)*q2(3);
        q(3,1) = q1(1)*q2(3) + q1(2)*q2(4) + q1(3)*q2(1) - q1(4)*q2(2);
        q(4,1) = q1(1)*q2(4) - q1(2)*q2(3) + q1(3)*q2(2) + q1(4)*q2(1);
    end
end
```

4.3 Onboard_temp

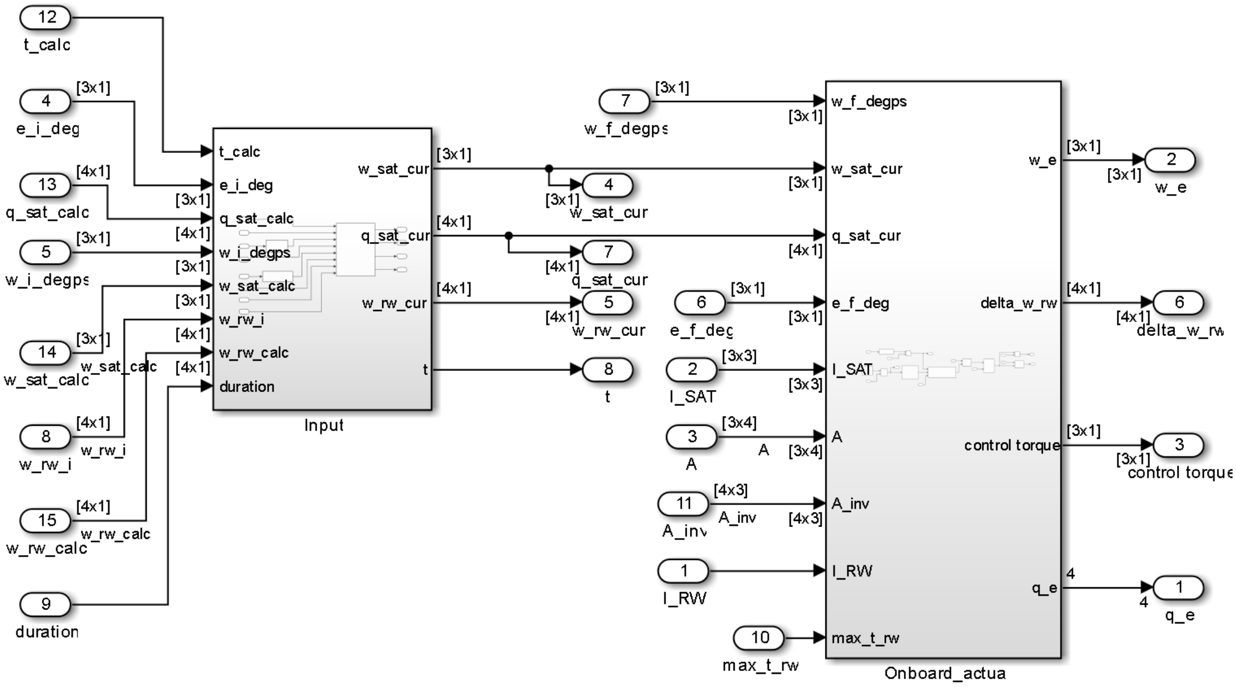


Figure 4-1 Onboard_temp

4.3.1 Hierarchy

- Onboard_temp
  - Input
    - Euler to quat
  - Onboard_actual
    - PID
    - euler to quat
    - quat error calc
    - quat to (rot_vec)*theta

4.3.2 Onboard_temp Input Summary

Port	Import Block	Source	Name	DataType
1	I_RW	ADCS/Init/ SFunction (Port 5)	I_RW	double
2	I_SAT	ADCS/Init/ SFunction (Port 6)	I_SAT	double
3	A	ADCS/Init/ SFunction (Port 7)	beta_deg	double
4	e_i_deg	ADCS/Init/ SFunction (Port 8)	e_i_deg	double
5	w_i_degps	ADCS/Init/ SFunction (Port 9)	w_i_degps	double
6	e_f_deg	ADCS/Init/ SFunction (Port 10)	e_f_deg	double

Port	Import Block	Source	Name	DataType
7	w_f_degps	ADCS/Init/ SFunction (Port 11)	w_f_degps	double
8	w_rw_i	ADCS/Init/ SFunction (Port 12)	w_rw_i	double
9	duration	ADCS/Init/ SFunction (Port 13)	duration	double
10	max_t_rw	ADCS/Init/ SFunction (Port 14)		double
11	A_inv	ADCS/Init/ SFunction (Port 15)		double
12	t_calc	ADCS/Unit Delay1	t	double
13	q_sat_calc	ADCS/Unit Delay2	q_sat	double
14	w_sat_calc	ADCS/Unit Delay11	W_sat	double
15	w_rw_calc	ADCS/Unit Delay12	w_rw	double

4.3.3 Onboard_temp Output Summary

Port	Outport Block	Destination	Name	DataType
1	q_e	ADCS/Unit Delay10	q_e	double
2	w_e	ADCS/Unit Delay9	w_e	double
3	control torque	ADCS/Unit Delay7	Tc	double
4	w_sat_cur	ADCS/Propagation/ SFunction (Port 8)	w_sat_cur	double
5	w_rw_cur	ADCS/Propagation/ SFunction (Port 9)	w_rw_cur	double
6	delta_w_rw	ADCS/Unit Delay5	delta_w_rw	double
7	q_sat_cur	ADCS/Propagation/ SFunction (Port 11)	q_sat_cur	double
8	t	ADCS/Propagation/ SFunction (Port 12)	t	double

4.3.4 *Input (to Controller)*

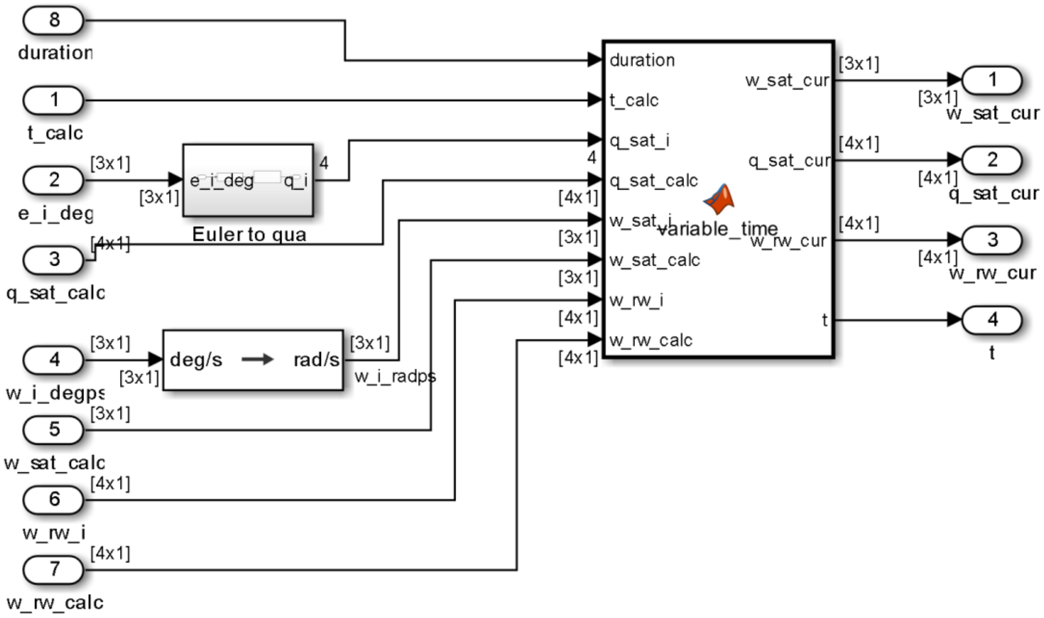


Figure 4-2 Input to Controller

4.3.4.1 *"Input" Input Summary*

Port	Import Block	Source	Name	DataType
1	t_calc	ADCS/Unit Delay1		double
2	e_i_deg	ADCS/Init/ SFunction (Port 8)		double
3	q_sat_calc	ADCS/Unit Delay2		double
4	w_i_degps	ADCS/Init/ SFunction (Port 9)		double
5	w_sat_calc	ADCS/Unit Delay11	w_sat_calc	double
6	w_rw_i	ADCS/Init/ SFunction (Port 12)	w_rw_i	double
7	w_rw_calc	ADCS/Unit Delay12	w_rw_calc	double
8	duration	ADCS/Init/ SFunction (Port 13)		double

4.3.4.2 *"Input" Output Summary*

Port	Outport Block	Destination	DataType
1	w_sat_cur	• ADCS/Onboard_temp/Onboard_actual/angular vel error calc (Port 2) • ADCS/Propagation/ SFunction (Port 8)	double
2	q_sat_cur	• ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Inverse/Quaternion Norm/Demux	double

Port	Outport Block	Destination	DataType
		• ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Inverse/Quaternion Conjugate/Demux • ADCS/Propagation/ SFunction (Port 11)	
3	w_rw_cur	ADCS/Propagation/ SFunction (Port 9)	double

4.3.4.3 MATLAB Function Argument Summary

Name	Scope	Port	Data Type	Size
duration	Input	1	double	1
q_sat_calc	Input	4	double	[4, 1]
q_sat_i	Input	3	double	4
t_calc	Input	2	double	1
w_rw_calc	Input	8	double	[4, 1]
w_rw_i	Input	7	double	[4, 1]
w_sat_calc	Input	6	double	[3, 1]
w_sat_i	Input	5	double	[3, 1]
q_sat_cur	Output	2	double	[4, 1]
t	Output	4	double	1
w_rw_cur	Output	3	double	[4, 1]
w_sat_cur	Output	1	double	[3, 1]

4.3.4.4 MATLAB Script

```
function [w_sat_cur, q_sat_cur,w_rw_cur, t] =
variable_time(duration,t_calc,q_sat_i,q_sat_calc, w_sat_i, w_sat_calc,w_rw_i, w_rw_calc)
  %Output initialised
  t=0.00;
  w_sat_cur=w_sat_i; %inital angular vel of satellite rad/s
  q_sat_cur=q_sat_i; % initial attitude of the satellite
  w_rw_cur=w_rw_i; %inital angular vel of reaction wheels rad/s
  if t_calc>0.00 && t_calc<=duration
    t=t_calc; %current time, output from rk4 progation function
    w_sat_cur=w_sat_calc;%current angular vel of satellite (rad/s), output from rk4
progation function
    q_sat_cur=q_sat_calc;%current attitude of satellite, output from rk4 propagation
function
    w_rw_cur=w_rw_calc;%current angular vel of reaction wheels (rad/s), output from rk4
progation function
  end
end
```

4.3.5 *Onboard_actual (controller)*

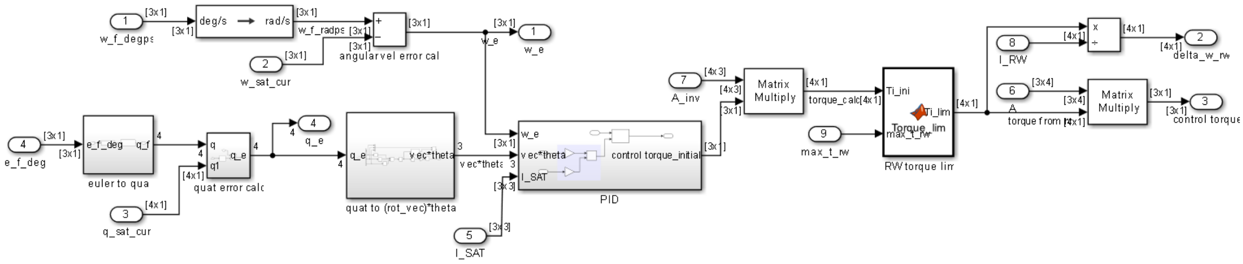


Figure 4-3 Onboard_actual (controller)

4.3.5.1 *Onboard_actual Input Summary*

Port	Import Block	Source	Name	DataType
1	w_f_degps	ADCS/Init/ SFunction (Port 11)		double
2	w_sat_cur	ADCS/Onboard_temp/Input/MATLAB Function2/ SFunction (Port 2)		double
3	q_sat_cur	ADCS/Onboard_temp/Input/MATLAB Function2/ SFunction (Port 3)		double
4	e_f_deg	ADCS/Init/ SFunction (Port 10)		double
5	I_SAT	ADCS/Init/ SFunction (Port 6)		double
6	A	ADCS/Init/ SFunction (Port 7)	A	double
7	A_inv	ADCS/Init/ SFunction (Port 15)	A_inv	double
8	I_RW	ADCS/Init/ SFunction (Port 5)		double
9	max_t_rw	ADCS/Init/ SFunction (Port 14)		double

4.3.5.2 *Onboard_actual Output Summary*

Port	Outport Block	Destination	DataType
1	w_e	ADCS/Unit Delay9	double
2	delta_w_rw	ADCS/Unit Delay5	double
3	control torque	ADCS/Unit Delay7	double
4	q_e	ADCS/Unit Delay10	double

4.3.5.3 quat to (rot_vec)*theta

4.3.5.3.1 Introduction:

Receives *q_e* (attitude-error quaternion) and outputs a vector: *angle_of_rotation *rotation vector*.

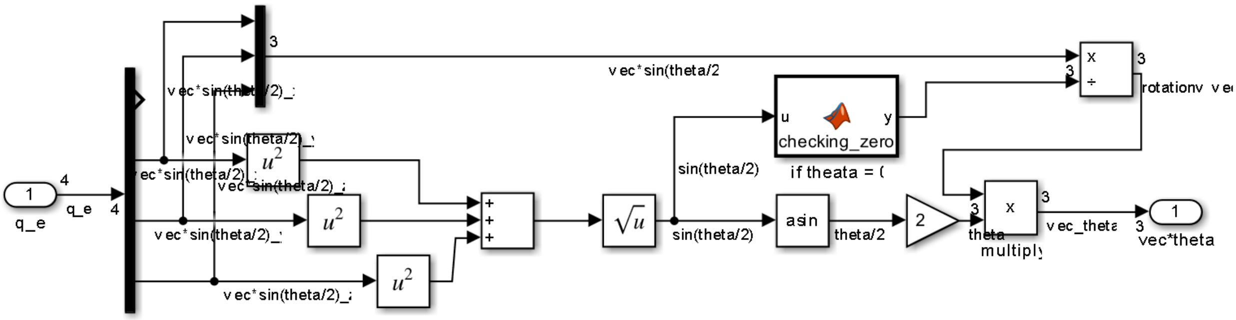


*Figure 4-4 quat to (rot_vec)*theta*

4.3.5.3.2 quat to (rot_vec)*theta Input Summary

Port	Import Block	Source	DataType
1	q_e	• ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Multiplication/q0/Sum • ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Multiplication/q1/Sum • ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Multiplication/q2/Sum • ADCS/Onboard_temp/Onboard_actual/quat error calc/Quaternion Multiplication/q3/Sum	double

4.3.5.3.3 quat to (rot_vec)*theta Output Summary

Port	Outport Block	Destination	Name	DataType
1	vec*theta	ADCS/Onboard_temp/Onboard_actual/PID/Kp	vec*theta	double

4.3.5.4 PD

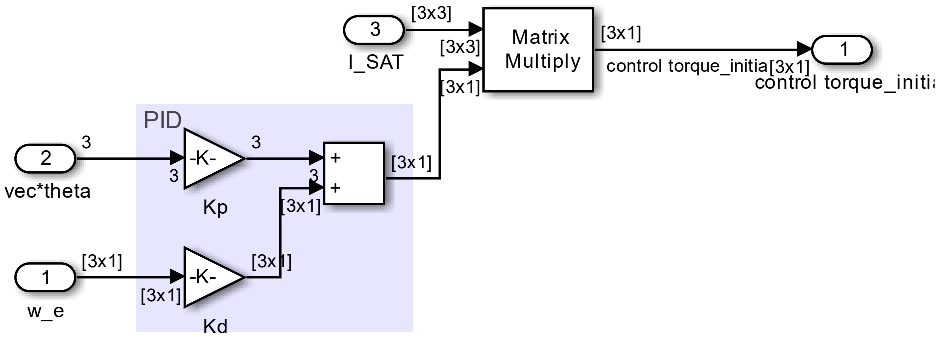


Figure 4-5 PD controller

4.3.5.4.1 PD Input Summary

Port	Import Block	Source	Name	DataType
1	w_e	ADCS/Onboard_temp/Onboard_actual/angular vel error calc	w_e	double
2	vec*theta	ADCS/Onboard_temp/Onboard_actual/quat to (rot_vec)*theta /multiply	vec*theta	double
3	I_SAT	ADCS/Init/ SFunction (Port 6)		double

4.3.5.4.2 PD Output Summary

Port	Outport Block	Destination	DataType
1	control torque_initial	ADCS/Onboard_temp/Onboard_actual/Matrix Multiply (Port 2)	double

4.3.5.1 *Torque_lim*

4.3.5.1.1 RW torque lim Argument Summary

Name	Scope	Port	Data Type	Size
Ti_ini	Input	1	double	[4, 1]
max_t_rw	Input	2	double	1
Ti_lim	Output	1	double	[4, 1]

4.3.5.1.2 RW torque lim Function Script

```
function Ti_lim = Torque_lim(Ti_ini,max_t_rw)
  Ti_lim=Ti_ini;
  for i=1:1:4
    if Ti_ini(i,1)>max_t_rw
      Ti_lim(i,1)=max_t_rw;

    elseif Ti_ini(i,1)<(-max_t_rw)
      Ti_lim(i,1)=-max_t_rw;

    end
  end

end
```

4.4 Graphs

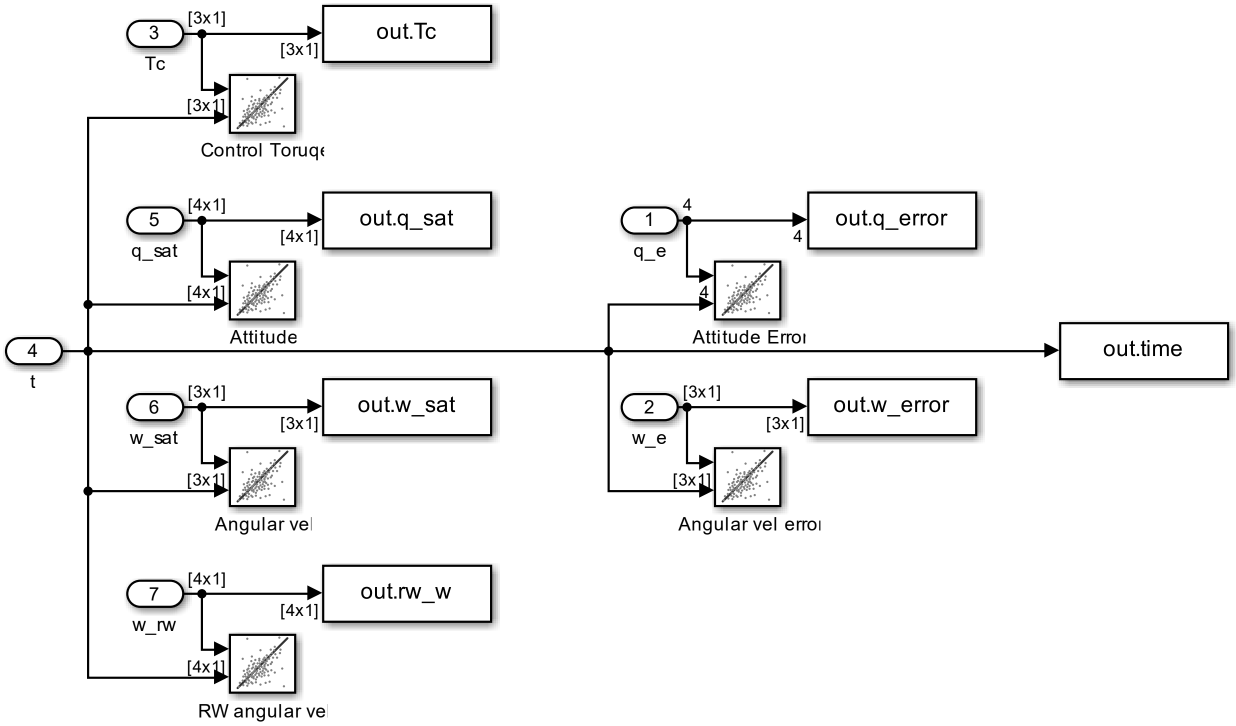


Figure 4-6 Graphs

4.4.1 Graphs Input Summary

Port	Import Block	Source	Name	DataType
1	q_e	ADCS/Unit Delay10	q_e	double
2	w_e	ADCS/Unit Delay9	w_e	double
3	Tc	ADCS/Unit Delay7	Tc	double
4	t	ADCS/Unit Delay1	t	double
5	q_sat	ADCS/Unit Delay2	q_sat	double
6	w_sat	ADCS/Unit Delay11	W_sat	double
7	w_rw	ADCS/Unit Delay12	w_rw	Double

5 PD gain tuning logic.

1) PD tuning logic: Assum the transfer function is 1/(s^2), sample time=1 sec, and max alpha (rad/s^2) = 0.0088 rad/sec^2. Follow the procedure per Section 5 of Reference 1.
2) The following logic calculates max Alpha's (angular acceleration's) value:
   a) The Max torque per reaction wheel is 0.015 Nm; thus, the max Torque is 0.0225 Nm, 0.0225 Nm and 0.015 Nm, respectively, for the x,y and z axes.
   b) Thus, the maximum angular acceleration is 0.0107 rad/s^2, 0.0099 rad/s^2 and 0.0088 rad/s^2, respectively, for the x, y and z axes.
   c) The minimum of the three is used to tune the PD gain values.

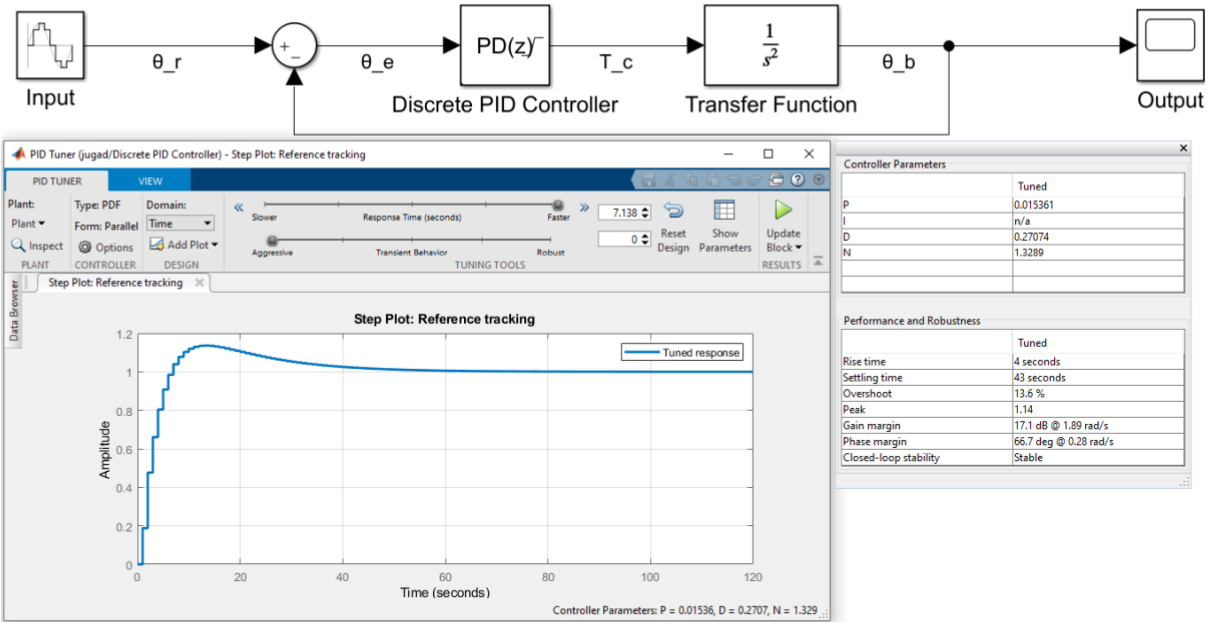


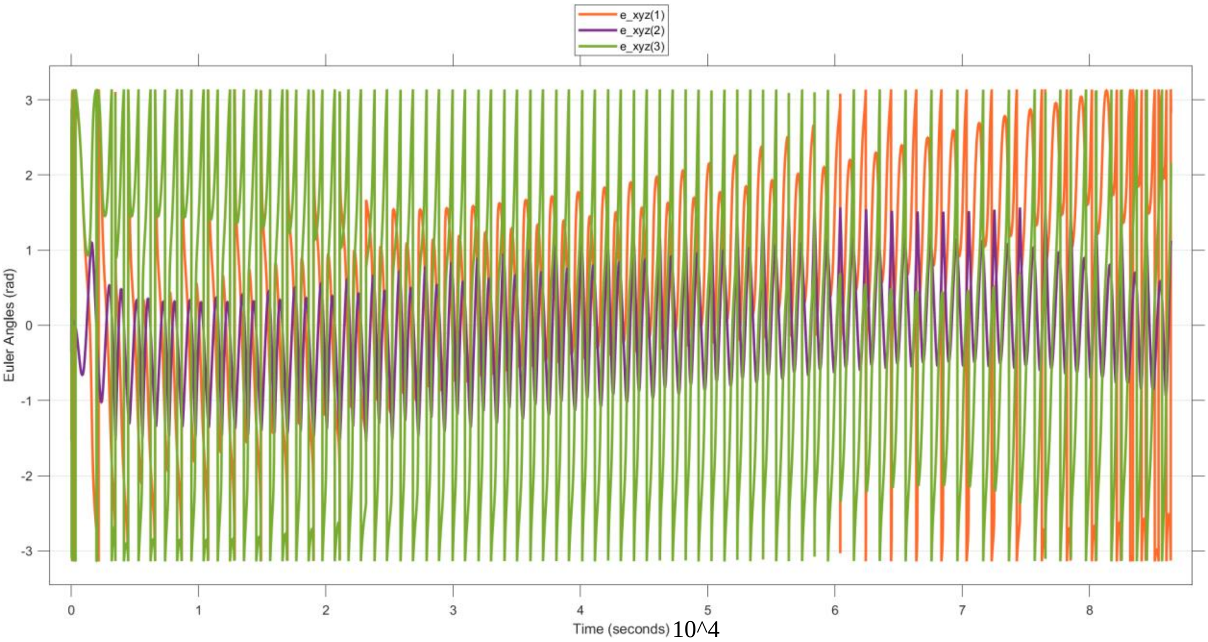
Figure 5-1 PD gains tuning logic (Fig 4 from Ref 1)

6 Results

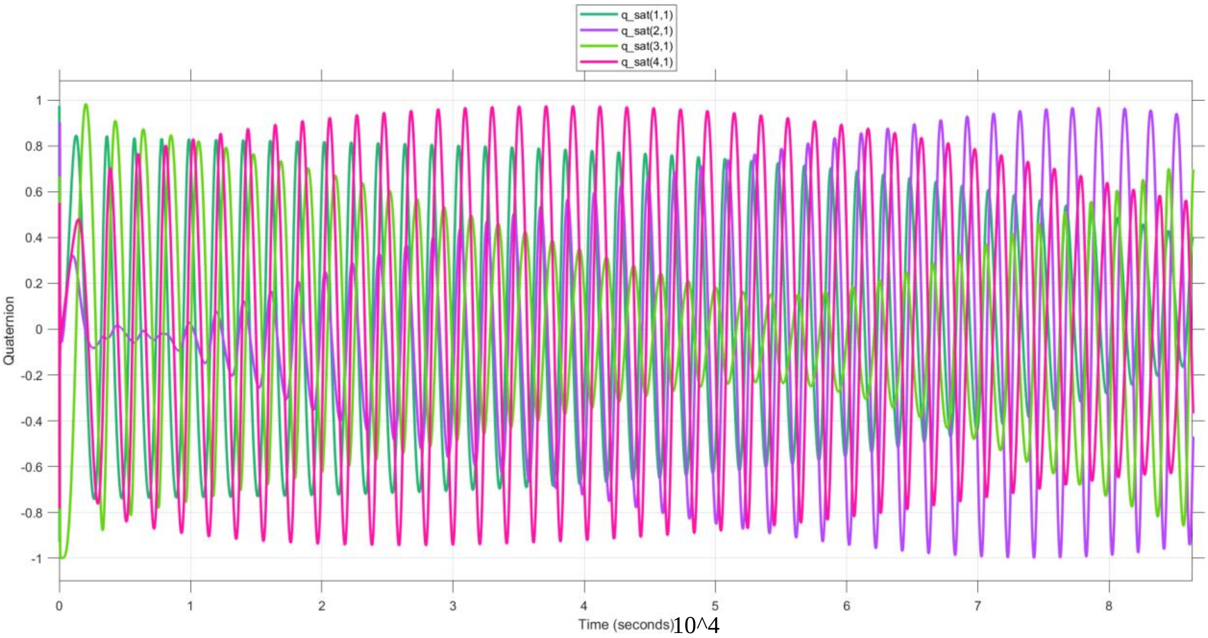
6.1 Results Expected by Pixxel Aerospace

6.1.1 Attitude vs Time

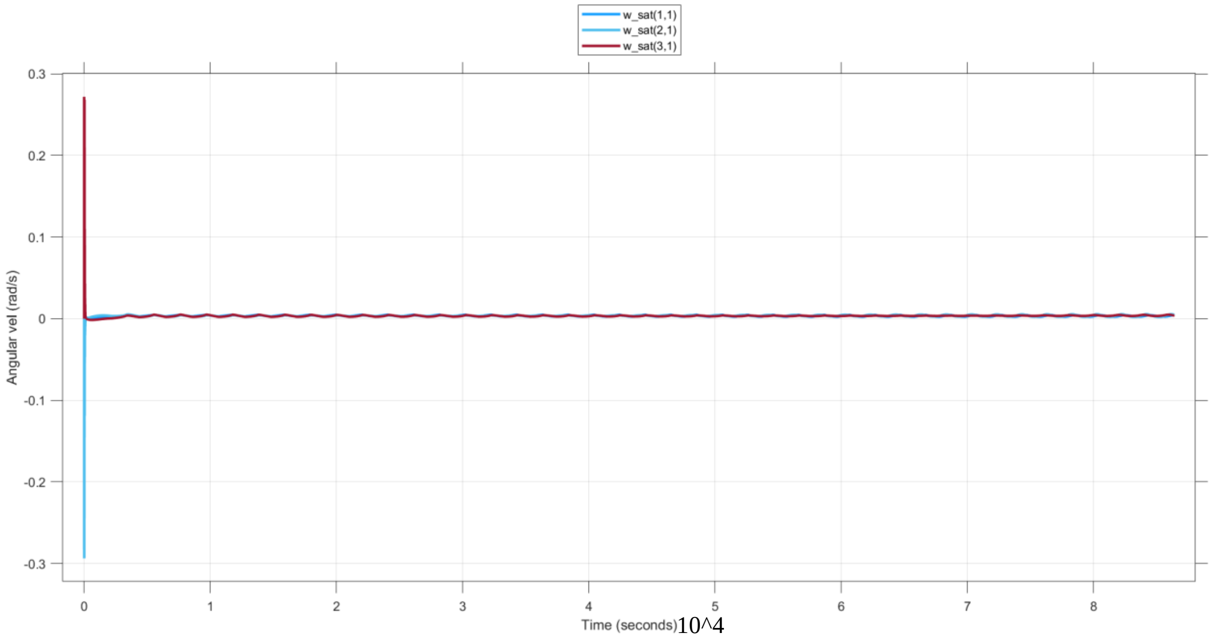
6.1.1.1 Euler Angles vs Time



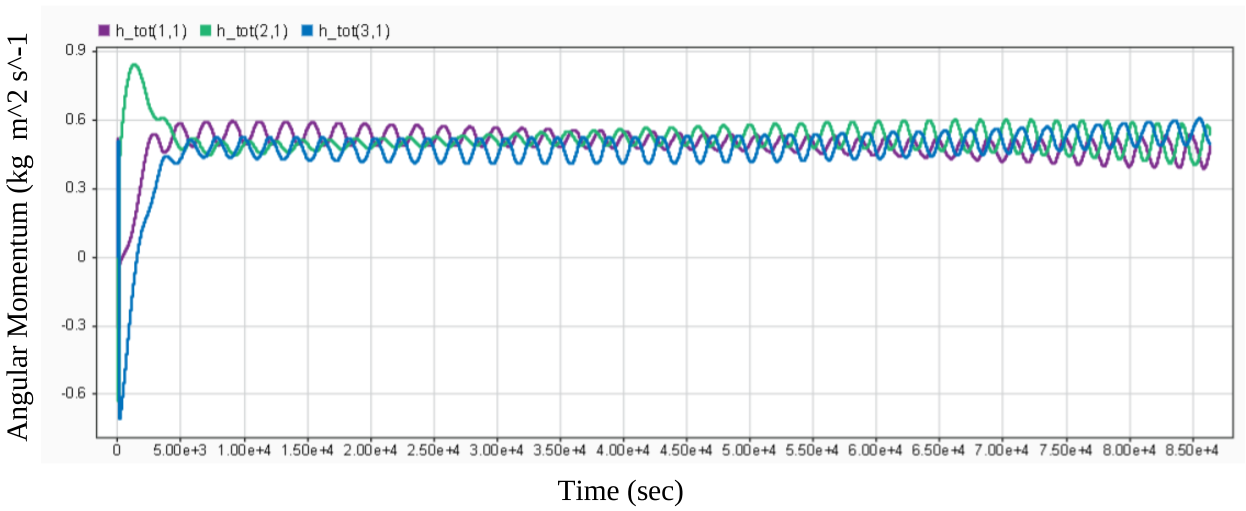
6.1.1.2 Quaternion vs Time



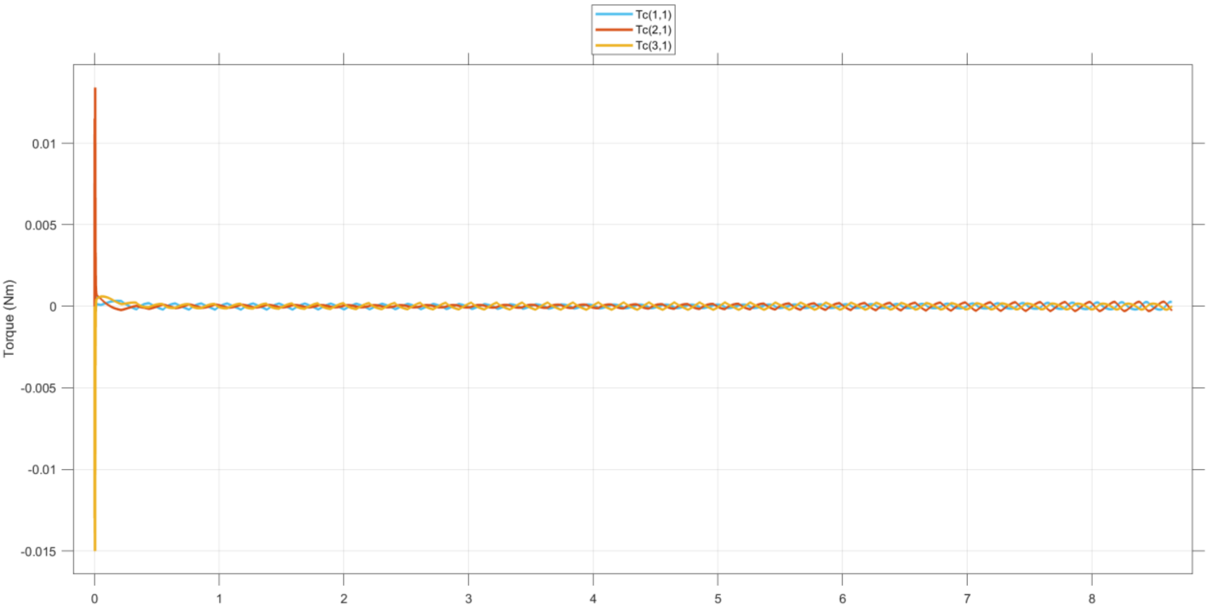
6.1.2 Angular velocity of satellite vs Time



6.1.3 Angular Momentum of satellite vs Time



6.1.4 Control Torque vs Time

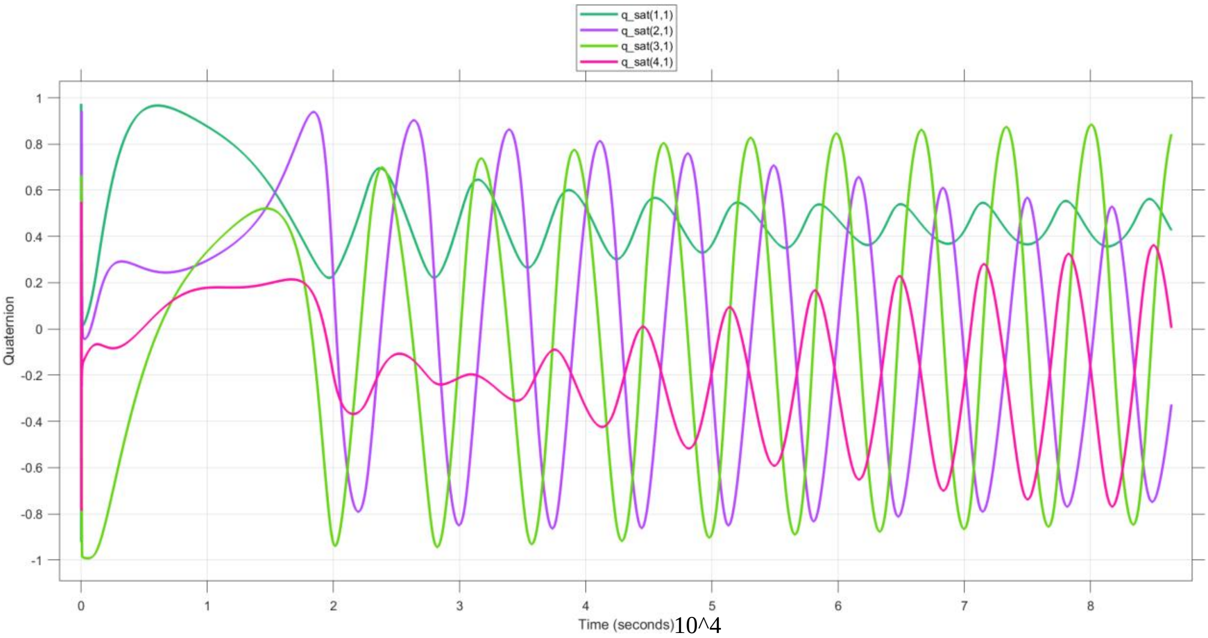


10^4

6.2 For Desired Angular vel is zero

w_f_degps=[0.00;0.00;0.00] in *Init* (refer Section 4.1.2)

6.2.1 Quaternion vs Time



10^4

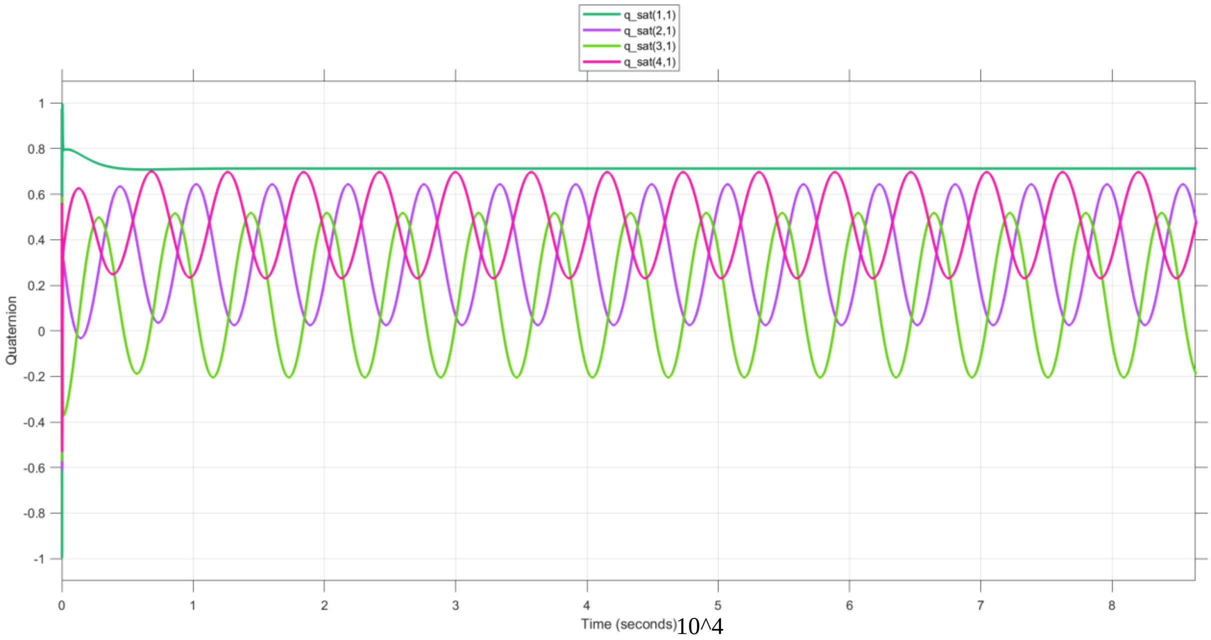
6.3 For Desired Angular vel is zero + Disturbance due to Angular momentum is zero

w_f_degps=[0.00;0.00;0.00] in *Init* (refer Section 4.1.2)

And,

d = cross(w_sat_cur, H) is changed to d=[0;0;0] in *funcEval_dwdt* (refer Section 4.2.2)

6.3.1 Quaternion vs Time



7 Inference

1) Reviewing 6.1.1.2 and 6.2.1 we can conclude that the desired final state for 6.1 is unstable, this is because the angular velocity is non-zero for a constant reference attitude.
2) Reviewing 6.2.1 and 6.3.1 we can conclude that the majority of instability seen in 6.1 and 6.2 is due to the angular momentum of the satellite and reaction wheels imparting a “disturbance torque”.
    a) This “Disturbance Torque” is equal to:

       -1 angular velocity of satellite* x *(total angular momentum of satellite and reaction wheel)*

8 Future Scope

1) Beta-Angle can be optimised. (Reference 2).
2) The PD controller described in Section 4.3.5.4 can be modified into a PID controller by including an integral for *vec*theta*.
    a) The procedure in Section 5 of Reference 1 can be used to find the Proportional, Integral and Derivative gains.
3) The PD controller described in Section 4.3.5.4 can be modified to offset the “disturbance torque” due to the angular momentum of the satellite.
    a) This “Disturbance Torque” is equal to:
       -1 angular velocity of satellite* x *(total angular momentum of satellite and reaction wheel)*

    b) ***NOTE:*** This was attempted
        i) The simulation ran based on Section 6.2, with the controller described in 4.3.5.2 modified to the following:

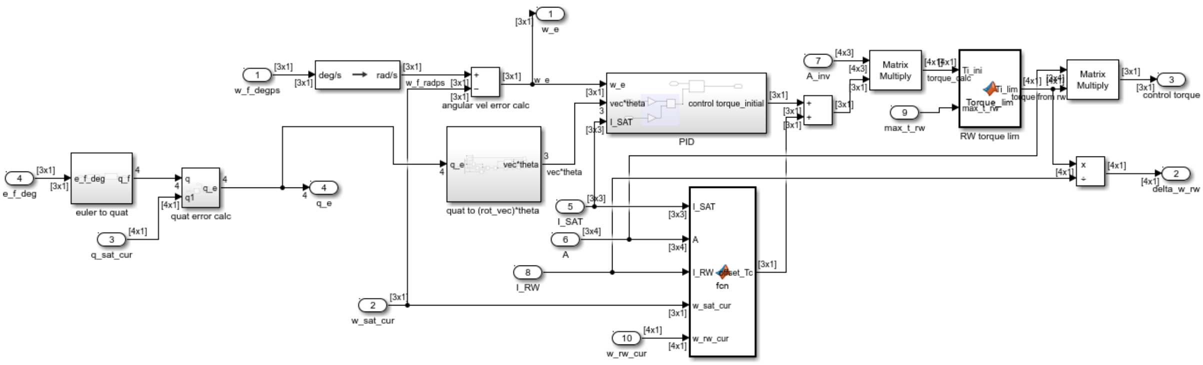
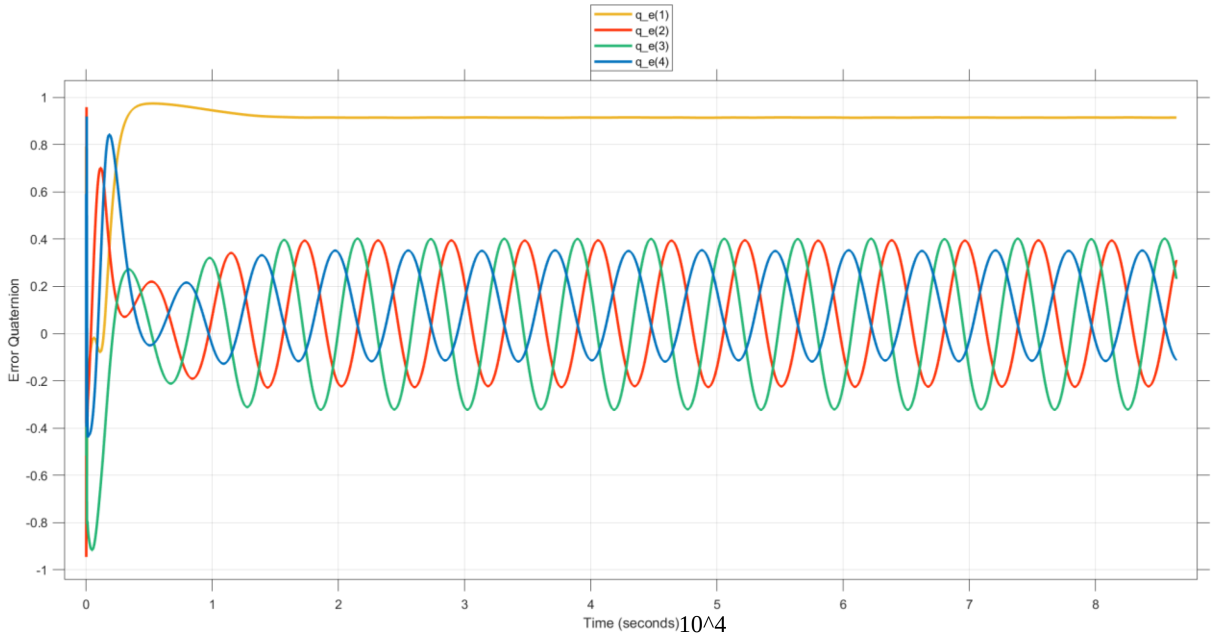


Figure 8-1 Offset Method Controller

The only addition was the following offset being added to the control torque:

Offset=* *angular velocity of satellite* x *(total angular momentum of satellite and reaction wheel)*

ii) Result: Quaternion vs Time



4) The PD controller described in Section 4.3.5.4 can be replaced with non-linear controllers more suited to this application.
5) The output of *Propagation*, along with an orbit-propagator, can be used to simulate sensor readings using various sensor models available in MATLAB.
   a) These simulated sensor readings can be used to test and optimise attitude acquisition algorithms (e.g. QuEst).
   b) Error/Noise can be added to the simulated sensor readings based on earlier hardware testing, which would be used to further optimise the various onboard algorithms.
6) The output of *Propagation*, along with an orbit-propagator, can be used to simulate Disturbance Torques more accurately using various physics models available in MATLAB.

References

1) D. Gundecha et al., "Complete Failure Analysis of Attitude Determination and Control System," 2021 IEEE Aerospace Conference (50100), Big Sky, MT, USA, 2021, pp. 1-16, doi: 10.1109/AERO50100.2021.9438456.
2) A. Kasiri, F. F. Saberi and M. Kashkul, "Optimisation of Pyramidal Reaction Wheel Configuration for Minimizing Angular Momentum," 2021 7th International Conference on Control, Instrumentation and Automation (ICCIA), Tabriz, Iran, 2021, pp. 1-6, doi: 10.1109/ICCIA52082.2021.9403596