.

Student Satellite Project
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076, INDIA

Website: www.aero.iitb.ac.in/satlab

README - LQR Controller

Guidance, Navigation and Controls Subsystem

linear_dynamics ()

Code author: Ronit
Created on: 15/07/2022
Last modified: 15/07/2022
Reviewed by: NA
Description:
Computes the derivative of state using linearized attitude dynamics. It has not been used in the code so far and only included for completeness.
Formula & References:
$x = \begin{bmatrix} q_1 & q_2 & q_3 & \omega_1 & \omega_2 & \omega_2 \end{bmatrix}^T$ Here q_1, q_2, q_3 represent the vector components of a quaternion. $\omega_1, \omega_2, \omega_3$ represent components of angular velocity in body frame.
u denotes control torque

$$\dot{x} = Ax + Bu$$

Here $A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \end{bmatrix}$, B is $\begin{bmatrix} \mathbf{I}^{-1} \\ \mathbf{O}_3 \end{bmatrix}$

$\mathbf{I}$ is moment of inertia matrix of satellite in body frame and $\mathbf{O_3}$ is the 3×3 zero matrix.
Input parameters:

1. **time** : (float) - Time at which derivative is computed. *seconds*

2. **state** : (numpy array) - State at which derivative is computed. *SI units for all*

Output:
It returns the $\dot{x}$ value mentioned earlier as a numpy array.

nonlinear_dynamics ()

Code author: Ronit
Created on: 15/07/2022
Last modified: 15/07/2022

Reviwed by: NA
Description:
Computes the derivative of state using nonlinear attitude dynamics. It has been used for conducting the simulations.
Formula & References:

$$\dot{q} = -\frac{1}{2}\omega \times q + \frac{1}{2}q_0\omega$$
$$\dot{\omega} = \mathbf{I}^{-1}u$$

Here q represents vector part of quaternion, q_0 represents scalar part of quaternion, ω represents angular velocity in body frame, $\mathbf{I}$ represents inertia matrix of satellite in body frame.
Input parameters:

1. **time** : (float) - Time at which derivative is computed. *seconds*
2. **state** : (numpy array) - State at which derivative is computed. *SI units for all*

Output:
It returns the $\dot{x}$ value mentioned earlier as a numpy array.

initialize_gain ()

Code author: Ronit
Created on: 15/07/2022
Last modified: 15/07/2022
Reviwed by: NA
Description:
Computes the gain matrix for finding control.
Formula & References: The cost function that will be minimized by the controller is given by

$$\frac{1}{2}\int_0^{\infty}[x^TQx + u^TRu]dt$$

Gain matrix is given by

$$K = -R^{-1}B^TF$$

here B is the defined in the linear_dynamics description, $Q = \begin{bmatrix} Q_1 & \mathbf{O_3} \\ \mathbf{O_3} & Q_2 \end{bmatrix}$, $F = \begin{bmatrix} F_{11} & F_{12} \\ F_{12}^T & F_{22} \end{bmatrix}$

$$F_{11} = \mathbf{I}R^{1/2}\left(Q_1 + \frac{1}{2}(\mathbf{I}R^{1/2}Q_2^{1/2} + Q_2^{1/2}R^{1/2}\mathbf{I})\right)^{1/2}$$

$$F_{12}^T = \mathbf{I}R^{1/2}Q_2^{1/2}$$
$$F_{22}^T = 2Q_2^{1/2}(Q_1 + \mathbf{I}R^{1/2}Q_2^{1/2})^{1/2}$$

For further details here is the link to the paper being referred.
Input parameters:
This function has no input parameters. However it uses certain constants relevant to controller that have been defined in the constants file.
Output:
It returns the 3×6 gain matrix as a numpy array.

control_law ()

Code author: Ronit
Created on: 15/07/2022
Last modified: 15/07/2022
Reviewed by: NA
Description:
Computes the control torque that needs to be applied at each state.
Formula & References:

$$u = Kx$$

here K is the gain matrix described in the previous function.
Input parameters:

1. **state** : (numpy array) - State at which control is computed. *SI units for all*

Output:
It returns the control torque that needs to be applied as a numpy array.