.

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

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

README - RM_linear_controller.m

Guidance, Navigation and Controls Subsystem

---

rk4method()

Code author: Ronit Chitre
Created on: 30/3/2022
Last modified: 31/3/2022
Reviewed by: Not yet reviewed
Description:
This is a numerical ode solver and uses rk4 method as its solving algorithm.
Formula & References:
For an ode $\frac{dx}{dt} = f(t, x)$ define h as the length between the time values at which solution is desired. Here N is the number of points over which solution is desired. w_i is the value of $x(t)$ at the 'i'th point. $t_i = t_0 + hi$.

$$w_{i+1} = w_i + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

Here $k_1 = hf(t_i, w_i)$, $k_2 = hf(t_i + \frac{h}{2}, w_i + \frac{k_1}{2})$, $k_3 = hf(t_i + \frac{h}{2}, w_i + \frac{k_2}{2})$, $k_4 = hf(t_{i+1}, w_i + k_3)$
Here the error is of the order $O(h^4)$

Input parameters:

1. **function of ode** : (function) - This is the function $f(t, x)$ that defines the ode. *value per time*
2. **initial conditions** : (numpy array) - This array will define the initial conditions or w_0. *value*
3. **time values** : (numpy array) - This array will contain all the time values on which value of $x(t)$ is to be found. *time*

Output:
If x is an $\mathbf{R}^{n\times 1}$ vector and m time values were given it will return an (m, n) matrix where each row will contain the value of x at that time instant.

Class System

method __init__

Code author: Ronit Chitre
Created on: 30/3/2022

Last modified: 31/3/2022
Reviewed by: Not yet reviewed
Description: This is the method that defines the class
Input parameters:

1. **State transition matrix** : (numpy array) - It is the state transition matrix for the system
2. **Control matrix** : (numpy array) - It is the control matrix of the system
3. **Feedback gain** : (numpy array) - It is the feedback gain chosen for the system

Output:
Returns an object of System class

method state_equation

Code author: Ronit Chitre
Created on: 30/3/2022
Last modified: 31/3/2022
Reviewed by: Not yet reviewed
Description: This method defines the state equation for the system
Input parameters:

1. No input parameters

Output:
For a system with ode $\dot{x} = f(t, x)$ it returns the function f

method solution

Code author: Ronit Chitre
Created on: 30/3/2022
Last modified: 31/3/2022
Reviewed by: Not yet reviewed
Description: This method solves the ode for the system numerically using *rk4method()* function defined earlier
Input parameters:

1. **Initial Conditions** : (numpy array) - The state vector of the system at $t = t_0$
2. **time values** : (numpy array) - The values of time were $x(t)$ has to be calculated

Output:
If x is an $\mathbf{R}^{n\times 1}$ vector and m time values were given it will return an (m, n) matrix where each row will contain the value of x at that time instant.

ackermann()

Code author: Ronit Chitre
Created on: 4/4/2022
Last modified: 4/4/2022
Reviewed by: Not yet reviewed
Description:
This will be used to find appropriate feedback gain for single input systems.
Formula & References:
For a system of order n of the form

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

$$y = Cx + Du$$

Feedback gain K being applied as $u = -Kx$ is given by

$$K = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix} U_C^{-1} \phi(A)$$

Here U_C is the controllability matrix.

$$U_C = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$$

$\phi(A)$ is the closed loop characteristic polynomial being evaluated at A. In practice however it will be found from the desired roots of the closed loop characteristic polynomial.
Link to proof
Input parameters:

1. **state transition matrix (A)** : (numpy array) - It is the state transition matrix for the system. It will always be a square matrix *no units*
2. **control matrix (B)** : (numpy array) - It is the control matrix for the system. It will always be a vector *no units*
3. **desired poles** : (array) - This array will contain the desired roots the closed loop characteristic polynomial must have. *no units*

Output: The output will be the appropriate feedback gain that needs to chosen to get the desired roots. It's datatype will be numpy array.

controllability()

Code author: Ronit Chitre
Created on: 4/4/2022
Last modified: 4/4/2022
Reviewed by: Not yet reviewed
Description:
This will be used to find the controllability matrix for a give state space.
Formula & References:
For a system of order n of the form

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

$$y = Cx + Du$$

Controllability matrix is given by

$$U_C = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$$

Input parameters:

1. **state transition matrix (A)** : (numpy array) - It is the state transition matrix for the system. It will always be a square matrix *no units*
2. **control matrix (B)** : (numpy array) - It is the control matrix for the system. It will always be a vector *no units*

Output: The output will be the controllability matrix for the system. It's datatype will be numpy array.

closed_loop_poly()

Code author: Ronit Chitre
Created on: 4/4/2022
Last modified: 4/4/2022
Reviewed by: Not yet reviewed
Description:
This will be used to find the closed loop characteristic polynomial evaluated at A, given its roots
Formula & References:
For a give set of roots $p1, p2, \dots$ characteristic polynomial ϕ is defined as

$$\phi(s) = (s - p1)(s - p2)\dots$$

The coefficients are obtained by *polyfromroots()* function from numpy module. Link to documentation
Input parameters:

1. **desired poles** : (array) - This array will contain the desired roots the closed loop characteristic polynomial must have. *no units*
2. **state transition matrix (A)** : (numpy array) - It is the state transition matrix for the system. It will always be a square matrix *no units*

Output: The output will be the $\phi(A)$ matrix described in *ackermann()*. It will be a numpy array.