Python Control Systems Toolbox

Release v0.0.7

Aug 07, 2020

Contents

The *control-toolbox* is a Python Library for implementing and simulating various systems and control strategies.

Current Supported Functionality:

- System modeling with Transfer Functions and State Space Representations.
- Time Domain Response.
- Frequency Response.
- System Representation conversion: State Space model to Transfer Function and vice versa.
- Block diagram algebra: Series and Parallel.
- Stability Analysis.
- Root Locus.
- Bode Plot.
- Parameterization of System.
- Pole-Zero / Eigenvalue plot of systems.
- Feedback analysis.
- PID control.
- Observability and Controllability.
- Full State Feedback
- Full State Observer
- Linear Quadratic Regulator(LQR)
- Linear Quadratic Estimator(LQE) / Kalman Filter
- Linearization.
- System Identification.

Future Updates:

- Linear Quadratic Gaussian Control.
- Extended Kalman Filter.
- Unscented Kalman filter.
- Model Predictive Control.

Documentation:

CHAPTER 1

Introduction

The *control-toolbox* is a Python Library for implementing and simulating various systems and control strategies. All information regarding this library can be found here, including the documentation and usage of the modules.

1.1 Installation

The *control-toolbox* library can be installed using pip.

To install it use:

```
pip install control-toolbox
```

1.2 Development

To get the latest unreleased version

git clone https://github.com/rushad7/control-toolbox.git

1.3 Getting Started

To start using the package, we simply import it as:

```
>>> import control
```

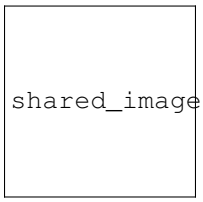
CHAPTER 2

System Representation

Systems can be defined by Transfer Functions and State Space models. Both system representations provide near identical utility.

2.1 Transfer Functions

A Transfer Function is the Laplace Transform of the ratio of output to input, and are mathematically described as:



To define a system in terms of a Transfer Function, use the *TransferFunction* class.

```
>>> import control
>>> s = control.TransferFunction(num, den)
```

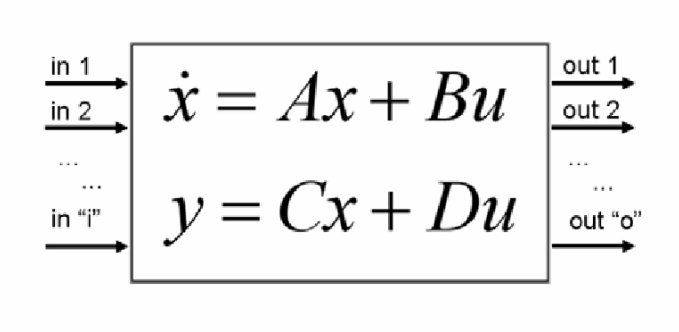
Here *num* and *den* can be lists or numpy arrays

2.2 State Space Models

Space State models are mathamatically described as:

To define a system as Space State model, use the *StateSpace* class.

```
>>> import control
>>> s = control.StateSpce(A,B,C,D)
```



Here, A,B,C,D are ndarrays.

Note: A system can be defined by any of the two representations above. If a particular method is needed but is not provided for the given system representation (which is unlikely), you can convert the system model to the desired representation using the *convert2TF()* or *convert2SS()* method.

CHAPTER 3

Function Reference

The following table lists the classes and methods that can be used to model and simulate various systems and control strategies.

3.1 System Definition

Class	Description
TransferFunction(num, den)	Creates a Transfer Function system object
StateSpace(A,B,C,D)	Creates a State Space system object

3.2 Transfer Function Methods

Method	Description
display()	Displays the Transfer Function system
parameters(settling_time_tolerance=0.02)	Returns the parameters of the system
response(input_type, time_period=10, sample_time=0.05, ret=False, show=True)	Returns the time domain response on the system
pzplot(ret=True)	Plots the Pole-Zero plot of the system and return poles and zeros
stability()	Returns the stability of the system
rootlocus(tf, gain_range=10.0)	Returns Root Locus of the system

3.3 State Space Methods

Method	Description
display()	Displays the State Space system
convert2TF()	Converts State Space model to Transfer Function
contr()	Controllability of the system
obs()	Observability of the system
stability()	Stability of the system
eigplot(ret=True)	Plots eigenvalues/poles of system
StateResponse(t, initial_cond, u, ret=False, show=True)	Returns and plots state response
OutputResponse(t, initial_cond, u, ret=False, show=True)	Returns and plots output response

3.4 System Connections

Method	Description
feedback(G, H=1.0, feedback_type="negative")	Creates a Feedback Object
*reduce.series(tf1, *tfn)*	Return the series connection (tfn * ...
*reduce.parallel(tf1, *tfn)*	Return the parallel connection (tfn + ...

3.5 Frequency Domain Analysis

Method	Description
bode.freqresp(tf)	Returns the Frequency Response of the system
bode.bode(tf)	Returns the Bode Plot of the system

3.6 PID Controller Design

Method	Description
PID(Kp, Ki, Kd, tf)	Creates a PID object
response(input_type, time_period=10, sample_time=0.05, ret=False, show=True)	Return the response of the system after PID control
tune(input_type="step", set_point=1, num_itr=70, rate=0.00000000001, lambd=0.7)	Tune the PID coefficients
display()	Display the PID block
reduced_tf	Displays the reduced Transfer Function (Controller + Plant)

3.7 State Feedback Design

Method	Description
StateFeedback(ss)	Creates a StateFeedback object
solve(roots)	Calculates the State Feedback gain matrix
model(k_ref=1)	Retruns StateSpace object after applying State Feedback

3.8 State Observer Design

Method	Description
StateObserver(ss)	Creates StateObserver object
solve(roots)	Calculates the State Observer gain matrix
model(k_ref=1)	Retruns the StateSpace object after applying State Observervation

3.9 Linear Quadratic Regulator(LQR)

Method	Description
LQR(ss, Q, R, N)	Creates an LQR object
solve()	Calculates the optimal gain matrix
model()	Returns the StateSpace object after applying State Observation

3.10 Kalman Filter / Linear Quadratic Estimator(LQE)

Method	Description
KalmanFilter(ss, Q, R, P=None, x0=None)	Creates a KalmanFilter object
predict(u = 0):	Predict next state
update(z):	Update Predictions
solve(measurements, ret_k=False, ret_x=False):	Returns the Predictions, Kalman Gain, and States of the system

3.11 Linearization

Method	Description
linearize(ss, x0, x_op, u_op, y_op)	Creates a *linearize* object
linearize()	Linearizes the system and returns the systems state space matrices

3.12 System Identification

Method	Description
SystemIdentification(path_x, path_x_dot, path_y)	Creates a *SystemIdentification* object
fit(num_epochs=500):	Models the system
model():	Returns the dictioinary of system matrices

- genindex