# Proper Orthogonal Decomposition Documentation

***Release 0.0.5***

**Johannes Scharlach**

May 14, 2014

CONTENTS


1 analysis module 3

2 debug module 5

3 example2sys module 7

4 futurescipy module 9

5 pod module 11

6 tests module 15

7 Indices and tables 17

Python Module Index 19

Index 21

Contents:

CHAPTER

# ONE

# ANALYSIS MODULE

**class** analysis.**Timer**
Bases: object

Allows some basic profiling

analysis.**plotResults**(*Y*, *T*, *label=None*, *legend_loc='upper left'*, *show_legend=False*)

analysis.**randomRuns**(*sys*, *rsys*, *T*, *sigma=10.0*, *integrator='dopri5'*)

analysis.**runAnalysis**(*n*, *k*, *N=1*, *example='butter'*, *T=20*, *sigma=1.0*, *integrator='dopri5'*)

analysis.**x0**(*n*)

CHAPTER

TWO

# DEBUG MODULE

**class** debug.**InputOfCalls**(*f*)

Bases: object

Save the different calls to functions in order to see what might go wrong somewhere

**classmethod inputs**()

return a dict of {function: [inputs],...}

**instances = {}**

CHAPTER

THREE

# EXAMPLE2SYS MODULE

example2sys.**example2sys**(*filename*)

example2sys.**generateRandomExample**(*n, m, p=None, distribution=<bound method Random.gauss of <random.Random object at 0x7fd3ec11d620>>, distributionArguments=[0.0, 1.0]*)

Generate a random example of arbitraty order

**How to use:** sys = generateRandomExample(n, m, [p, distribution])

**Inputs:** n system order m number of Inputs p number of outputs [p=m] distribution

distribution of the matrix values [distribution=gauss(0., 1.)]

**Output:** sys random StateSpaceSystem with the parameters set in the input

example2sys.**heatSystem**(*N*, *L=1.0*, *g0=0.0*, *gN=0.0*)

example2sys.**optionPricing**(*N=None*, *option='put'*, *r=0.05*, *T=1.0*, *K=100.0*, *L=None*)

example2sys.**stableRandomSystem**(**args*, ***kwargs*)

CHAPTER

FOUR

# FUTURESCIPY MODULE

Some functions that will be integrated into scipy in a future version. The use of this file should be avoided as soon as the functions are fixed in scipy.

futurescipy.**abcd_normalize**(*A=None*, *B=None*, *C=None*, *D=None*)

Check state-space matrices and ensure they are rank-2.

If enough information on the system is provided, that is, enough properly-shaped arrays are passed to the function, the missing ones are built from this information, ensuring the correct number of rows and columns. Otherwise a ValueError is raised.

**Parameters** **B, C, D** (*A,*) – State-space matrices. All of them are None (missing) by default.

**Returns** **A, B, C, D** – Properly shaped state-space matrices.

**Return type** array

**Raises** ValueError – If not enough information on the system was provided.

CHAPTER

FIVE

# POD MODULE

Model order reduction for linear state space systems can be done with Proper Orthogonal Decomposition (POD) methods. Some of them can be simply applied when creating a system.

pod.**isStable**(*A*)

**class** pod.**lss**(**create_from*, ***reduction_options*)

Bases: object

linear time independent state space system.

Default contstructor is called like lss(A,B,C,D) and a system can easily be copied by calling lss(sys) where *sys* is a lss object itself.

**Parameters**

- **A** (*array_like*) –
- **B** (*array_like*) –
- **C** (*array_like*) –
- **D** (*array_like*) – State-Space matrices. If one of the matrices is None, it is replaced by a zero matrix with appropriate dimensions.
- **reduction** (*{'truncation_square_root'}, optional*) – Choose method of reduction. If it isn't provided, matrices are used without reduction.
- ****reduction_options** (*dict, optional*) – The arguments with which the reduction method is called.

**x0 array_like, optional**

Initial state. Defaults to the zero state.

**t0 float**

Initial value for *t*

**integrator str**

Name of the integrator used by scipy.integrate.ode

**integrator_options dict**

Options for the specified integrator that can be set.

**reduction_functions dict**

The functions that can be choosen as an input parameter with the *reduction* keyword.

**__call__**(*times*, *control=None*, *force_ode_reset=False*)

Get the output at specified times with a provided control

It is possible to only request the output at one particular time or provide a list of times. If *times* is a sequence, the output will be a list of `nparrays` at these times, otherwise it's just a single `nparray`. However the control can either be specified as a function or is a constant array over all times.

**Parameters**

- **times** (*list or scalar*) – The output for these timese will be calculated
- **control** (callable `control(t, y)` or array_like, optional) – If it is specified, it will be overwritten in the attributes.
- **force_ode_reset** (*Boolean, optional*) – If it's called, the ode solver is reset and the current attributes are used.

**f** (*t*, *y*, *u*)
Rhs of the differential equation

**integrator = 'dopri5'**

**integrator_options = {}**

**reduction_functions = {'truncation_square_root': <function truncation_square_root at 0x107713cf8>}**

**setupODE** ()
Set the ode solver. All integrator, options and initial value can be set through class attributes.

**solve** (*t*)

**t**
Current time of the system

**t0 = 0.0**

**x**
State of the system at current time *t*

**x0 = None**

**y**

pod.**truncation_square_root** (*A*, *B*, *C*, *k=0*, *tol=0.0*, *balance=True*, *scale=True*, *check_stability=True*, *length_cache_array=None*)
Perform truncation of a system. Scaling and balancing are optional

This allows to reduce a linear state space system by either specifying it to a certain number of states *k* or by specifying a error tolerance *tol* relative to the input. In theory the most accurate results are achieved by using *balance* and *scale* but the size of the error strongly depends on the particular problem and scaling and balancing may in some cases cost too much.

**Parameters**

- **A** (*array_like*) –
- **B** (*array_like*) –
- **C** (*array_like*) – State-Space matrices of the system that should be reduced
- **k** (*int, optional*) – Order of the output system
- **tol** (*float, optional*) – Error of the output system, based on the Hankel Singular Values
- **balance** (*Boolean, optional*) – Balance the system before reducing it to make sure, that the error is kept small
- **scale** (*Boolean, optional*) – Scale the system

- **check_stability** (*Boolean, optional*) – Checks if all the real parts of the eigenvalues of A are in the left half of the complex plane.

**Returns**

- **Nr** (*int*) – Actual size of the system, based on the error: If the Machine error would have been bigger than the error of the reduced system, it may happen that `Nr < k` and if the error would be inconsiderably bad, It might be the case that `Nr > k`. In case *k* was never specified, this is purely based on *tol*
- **Ar, Br, Cr** (*ndarray*) – Reduced arrays
- **hsv** (*ndarray*) – Hankel singular values of the original system. The size of the error may be calculated based on this.

**Raises**

- `ValueError` – If the system that's provided is not stable (i.e. *A* has eigenvalues which have non-negative real parts)
- `ImportError` – If the slycot subroutine *ab09ad* can't be found. Occurs if the slycot package is not installed.

CHAPTER

SIX

# TESTS MODULE

Unit tests and functional tests for the pod.py package

**class** tests.**testLss**(*methodName='runTest'*)

Bases: unittest.case.TestCase

test lss functionalities

**testIdentity**()

**test_abcd_normalize**()

**test_f**()

**test_zero_control**()

CHAPTER

SEVEN

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

a
analysis, 3
d
debug, 5
e
example2sys, 7
f
futurescipy, 9
p
pod, 11
t
tests, 15

## Symbols

__call__() (pod.lss method), 11

## A

abcd_normalize() (in module futurescipy), 9
analysis (module), 3

## D

debug (module), 5

## E

example2sys (module), 7
example2sys() (in module example2sys), 7

## F

f() (pod.lss method), 12
futurescipy (module), 9

## G

generateRandomExample() (in module example2sys), 7

## H

heatSystem() (in module example2sys), 7

## I

InputOfCalls (class in debug), 5
inputs() (debug.InputOfCalls class method), 5
instances (debug.InputOfCalls attribute), 5
integrator (pod.lss attribute), 11, 12
integrator_options (pod.lss attribute), 11, 12
isStable() (in module pod), 11

## L

lss (class in pod), 11

## O

optionPricing() (in module example2sys), 7

## P

plotResults() (in module analysis), 3
pod (module), 11

## R

randomRuns() (in module analysis), 3
reduction_functions (pod.lss attribute), 11, 12
runAnalysis() (in module analysis), 3

## S

setupODE() (pod.lss method), 12
solve() (pod.lss method), 12
stableRandomSystem() (in module example2sys), 7

## T

t (pod.lss attribute), 12
t0 (pod.lss attribute), 11, 12
test_abcd_normalize() (tests.testLss method), 15
test_f() (tests.testLss method), 15
test_zero_control() (tests.testLss method), 15
testIdentity() (tests.testLss method), 15
testLss (class in tests), 15
tests (module), 15
Timer (class in analysis), 3
truncation_square_root() (in module pod), 12

## X

x (pod.lss attribute), 12
x0 (pod.lss attribute), 11, 12
x0() (in module analysis), 3

## Y

y (pod.lss attribute), 12