

Control

CONTROL OF AIRCRAFT

PYTHON AND CONTROL

December 9, 2023

CONTROL OF AIRCRAFT

- 1 CONTROL SOFTWARE, MATLAB AND PYTHON
- 2 ANACONDA
- 3 GOOGLE COLAB
- 4 PYTHON AND CONTROL
- 5 SISOTOOL

We have seen commands using Matlab. Most of them, except the block diagram approach with Simulink, are available under Octave, a free software.

An alternative approach is to use Scilab, which has control tools and an almost Simulink equivalent, Xcos, which handles block diagram simulation. But the syntax is different from Matlab.

Another alternative is to use Python, which also has a control package and packages to handle graphics and matrix calculations. In the next slides, we will see Python equivalent to Matlab commands. Your configuration of Python should include the following packages:

- control
- slycot
- scipy
- numpy
- pylab
- matplotlib
- math (should be installed)
- ipython

We will see how to install them on Windows, Linux and Mac OS X.

INTRODUCTION TO PYTHON

In the mini project, the Python programming language will be used. You should download and install this free software Python (3.xx version) with the following packages:

- scipy: a scientific library (offers integration, optimization ...)
- numpy: a package to deal with arrays and matrices
- slycot: a mathematical library
- pylab: a part of matplotlib which gives a Matlab like environment
- matplotlib: a package to create plots
- control: a package for control
- ipython: an enhanced interactive console

Python can be used with a command line shell, or alternatively with an IDE. An IDE such as Spyder provides an interface similar to the Matlab interface. It must be installed separately.

In Spyder, to act interactively with the plots of `sisopy31`, in Preferences/Python console/Graphics, change Backend to another option than `Inline`. Then close and reopen Spyder. You can also, once `matplotlib` has been loaded, type `%matplotlib qt5` in the command line (not in a script) to obtain the same behavior.

There is now an equivalent to Matlab `sisotool` in python control package (with limited capabilities), but there is a python script available on Moodle which provides an equivalent to `sisotool` with more information provided to the user: `sisopy31.py`

CONTROL OF AIRCRAFT

- 1 CONTROL SOFTWARE, MATLAB AND PYTHON
- 2 **ANACONDA**
- 3 GOOGLE COLAB
- 4 PYTHON AND CONTROL
- 5 SISOTOOL

PYTHON INSTALLATION ON WINDOWS, MACOS AND LINUX

On Windows, Mac OS X and Linux, the Python Anaconda distribution can be used because slycot and control are available for these 3 OS. You can download the distribution at

<https://www.anaconda.com/download>

You must choose a **64 bit** version.

- This has been tested on Mac OS X and Linux Ubuntu in 2018 and python **3.6**.
- It has also been tested with Python **3.9** on Windows 11.
- It has also been tested with Python **3.11** on Linux Ubuntu in october 2023 (and this is the recommended version).

Then type the following command in a terminal under macOS or Linux and an Anaconda prompt under Windows, to install the required packages (this command does not work inside Python)

```
conda install -c conda-forge slycot control
```

The last versions in October 2023 are available for Python 3.11:

- v0.5.4.0 for the slycot package, see <https://anaconda.org/conda-forge/slycot>
- v0.9.4 for the control package (Windows 64, Linux 64 and OSX 64), see <https://anaconda.org/conda-forge/control>

ANACONDA ENVIRONMENT

You can create an environment in Anaconda. This allows you to have several concurrent versions of Python installed on your computer and may prove useful if you have to deal with incompatible libraries needed in different courses.

To create your environment named myenv, with a specific version of Python (e.g. 3.11):

```
conda create --name myenv python=3.11
```

To activate the environment you have created

```
conda activate myenv
```

And to deactivate the environment

```
conda deactivate myenv
```

CONTROL OF AIRCRAFT

- 1 CONTROL SOFTWARE, MATLAB AND PYTHON
- 2 ANACONDA
- 3 GOOGLE COLAB**
- 4 PYTHON AND CONTROL
- 5 SISOTOOL

GOOGLE COLAB

There is an alternative of using a local Python installation: this is to use a remote computer on collab.

Go to <https://colab.google/>

You have to connect to collab with a google gmail address.

Create a new notebook.

Create a new section of code (+code) and type

```
%env SKBUILD_CONFIGURE_OPTIONS="--DBLA_VENDOR=Generic"  
!pip install slycot control  
# in order to have interactive matplotlib windows  
!pip install ipympl  
import numpy as np  
import slycot  
import control
```

GOOGLE COLAB

```
# Mount your google drive in google colab  
from google.colab import drive  
drive.mount( '/content/drive' )
```

Copy sisopy31.py in your google drive in the "content" directory, or alternatively create a new file in the content directory named sisopy31.py and copy and paste the content of the file from your disk to the newly created file on Google drive.

Note also that if you don't succeed in copying the file in content, you can copy it in content/drive/MyDrive and copy the file to current directory using:

```
!cp drive/MyDrive/sisopy31.py .
```

```
from google.colab import output  
output.enable_custom_widget_manager()
```

```
# system example  
sys=control.ss(np.array([[1,2],[3,4]]),\  
np.array([[1],[2]]),[0,1],0)
```

```
import ipyml  
import sisopy31  
sisopy31.sisotool(sys)
```

This is now possible to move the red stars on the root locus, which represent the location of closed loop poles in the complex plane. This has been tested in November 2023. Packages versions used are

- Python 3.10.12
- control 0.9.4
- slycot 0.5.4

CONTROL OF AIRCRAFT

- 1 CONTROL SOFTWARE, MATLAB AND PYTHON
- 2 ANACONDA
- 3 GOOGLE COLAB
- 4 PYTHON AND CONTROL**
- 5 SISOTOOL

How to use python:

- Launch ipython or spyder (preferred to python because ipython is more interactive)
- in the prompt, you can type commands such as

```
>>> a=0.127  
>>>a  
0.127  
>>> print( "a=%.3f "%a)  
a=0.127
```

- you can run a script (a text file you have written, call myscript.py) by typing:

```
>>> run myscript.py
```


A few things to know about python:

- indentation counts and is used to delimit blocks. For example:

```
i=True
if i==True :
    print("i _ true")
else :
    print("i _ false")
```

Note also the use of :

- in numpy, the indice of matrices begin at 0
- the indice of the end is not included.

```
>>> import numpy as np
>>> a=np.matrix ([[1],[2],[3],[4]])
>>> # row 2 to 3
>>> a[1:3]
matrix ([[2],
         [3]])
```

At the beginning of your script, you should put the following commands, in order to make all the packages available for the mini project (and you must copy sisopy31.py in your working directory):

PYTHON HEADINGS

```
#!/usr/bin/python  
# coding: utf-8
```

```
from __future__ import unicode_literals  
from matplotlib.pyplot import *  
import control  
from control.matlab import *  
from math import *  
from scipy.interpolate import interp1d  
from pylab import *  
from matplotlib.widgets import Slider  
import numpy as np  
import scipy.interpolate
```

The first line is used to make the script executable on a UNIX system.
The second line defines the coding convention of the text (utf-8).
The line with unicode allows the use of unicode in matplotlib plots.
To import a package, you have several options:

- import *packagename*: all commands of the package are available via *packagename.command*
- import *packagename* as *pk*: creates an alias and the commands are available via *pk.command*
- from *packagename* import *: all commands are directly available via *command*
- from *packagename* import *command*: import only the command *command* which is directly available via *command*

PYTHON CODE EXAMPLES

Define a matrix in Python

```
A=np.matrix([[ -4.232e-02,  -2.418e-02,  -4.764e-02,
               0.0,      0.0,      0.0],
 [  4.727e-02,   0.0,      1.628,
   0.0,      0.0,      0.0],
 [ -4.727e-02,   0.0,  -1.628e+00,
   1.0,      0.0,      0.0],
 [  0.0,      0.0,  -6.420e+01,
  -9.048e-01,   0.0,      0.0],
 [  0.0,      0.0,      0.0,
   1.0,      0.0,      0.0],
 [  0.0,      4.056e+02,   0.0,
   0.0,      0.0,      0.0]])
```

```
B=np.matrix([[ 0.          , 0.          ],  
             [ 0.4424, 0.          ],  
             [ -0.4424, 0.          ],  
             [-76.3469, 0.          ],  
             [ 0.          , 0.          ],  
             [ 0.          , 0.          ]])
```

```
# submatrix (matrix slicing)  
# note that indices begin at 0 and  
# the indice of the end is not included  
Ar=A[2:4 ,2:4]  
Br=B[2:4 ,0:1]  
Cr=np.matrix([0.0 ,1.0])  
Dr=0.0  
  
# define a state space system  
sys_q=ss(Ar,Br,Cr,Dr)  
# define a transfer function  
sys_q_tf=tf(sys_q)  
# damping ratio  
control.matlab.damp(sys_q)  
# dcgain  
dcgain(sys_q)
```

```
# step response
figure(1)
Yq,Tq=control.step(sys_q)
plot(Tq,Yq,'b',lw=2)
grid(True)
title('Réponse indicielle q/Dm')
show()

# feedback
Kq=-0.115
Tqbo=feedback(Kq*sys_q,1.0)
```

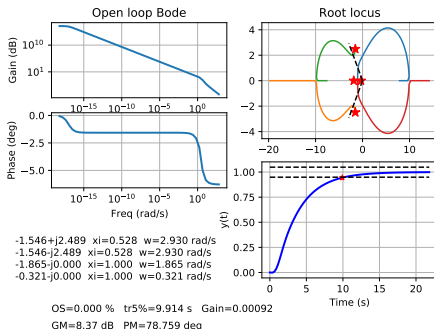

CONTROL OF AIRCRAFT

- 1 CONTROL SOFTWARE, MATLAB AND PYTHON
- 2 ANACONDA
- 3 GOOGLE COLAB
- 4 PYTHON AND CONTROL
- 5 **SISOTOOL**

The sisotool in python is provided in Python file sisopy31.py. Its usage is:

```
from sisopy31 import *  
sisotool(sys) # or sisotool(sys, gainmin, gainmax)
```

The gain is tunable with by clicking on root locus plot which moves the closed loop poles.



There is a plot giving closed loop poles, with their corresponding damping ratio.

On the main figure:

- OS is the overshoot of the closed loop step response.
- $t_{r5\%}$ is the settling time to within 5%.
- GM is the gain margin.
- PM is the phase margin.