CitAST for Python

Author: Killian Dally, MSc. Student
Original Code: Dave Kroezen, MSc. and Clark Borst, Assistant Professor.

Control & Simulation Division
Faculty of Aerospace Engineering

DELFT UNIVERSITY OF TECHNOLOGY

November 16, 2020

Contents

1 Compiling to C Code

To be able to use the `CitAST` environment in Python, you first have to compile the Simulink model to C code. It is suggested that you familiarize yourself with the `CitAST` environment by reading its documentation. For this tutorial, we will be working in the `CITAST_PYTHON` directory.

First, make sure you have MATLAB's Embedded Coder and compiler add-on packages installed. Two MATLAB executable `.MEX` files are already present in the directory but may not be compatible with your configuration. You can recompile them from the `ac_atmos.c` and `ac_axes.c` files from the MATLAB command line using:

```
mex <filename>.c
```

Run the `initcit.m` file and open the `citation_to_python.slx` Simulink model. In its present form, the input array has 10 dimensions, 8 aerodynamic control inputs and 2 thrust control inputs. The output has also been simplified to the 12-dimensional aircraft state. A yaw damper and an auto-throttle are already implemented.
You may add external disturbance and turbulence inputs or put back the original 32-dimensional output. When you have adjusted the Simulink model, make sure it runs without errors. Note that by changing the Simulink model input and outputs, you will have to modify the `swig.i` file discussed in the next section.

In 'Model Configuration Parameters', the following settings should be adjusted:

- Fixed time step (e.g. 0.01s) with ode5 solver.

- Select the appropriate device type as hardware implementation for your configuration.
- Select embedded coder `ert.tlc` as system target file, 'faster runs' as build configuration, and execution and RAM efficicency as objectives.
- In the interface tab, select floating-point numbers, continuous time, and non-finite numbers support.
- Choose to have a code generation report and have it open automatically.
- In the code placement tab, select 'modular' as file packaging format.

Then you can use the Embedded Coder to generate the code. A code generation report will be displayed, giving you an overview of the generated C files in the `CITAST_PYTHON\CITATION_TO_PYTHON_ERT_RTW` directory.

2 Create the Python Executable

We will use SWIG to create a Python executable from these C files, but other methods exist.[1] Download SWIG from `www.swig.org`, copy the SWIG folder to your home root directory and follow the installation instructions from the `INSTALL` document.

Copy the `setup.py`, `swig.i` and `numpy.i` files from `CITAST_PYTHON\PYTHON_UTILS` to the `CITATION_TO_PYTHON_ERT_RTW` directory. In `setup.py`, adapt the matlab_root variable for your configuration. Also, make sure that calling `python` from the terminal calls Python version 3.0 or higher. Start a terminal window from the current directory and run the following commands:

```
swig -python swig.i
python setup.py build_ext --inplace
```

You should now have a `citation.py` file from which you can run the `.PYD` (on Windows) or `.SO` (on Mac OS and Linux) executable. The executable and Python file can be moved to a directory of your choice.

If you are encountering issues with SWIG, we suggest to try running it on example files on the tool's website.[2]

[1] https://tristan-ka.github.io/IBOAT_RL/_downloads/SIMULINK_TO_C__PYTHON.pdf
[2] http://www.swig.org/Doc1.3/Python.html