CSALT Test System

September, 2017

Table of Contents

Introduction

The Collocation Standalone Library and Toolkit (CSALT) provides core capabilities for collocation based optimization. The system is built into a shared library named libCSALT_<ReleaseID> by default. The library is tested using a stand-alone application, TestCSALT_<ReleaseID>. The tag <ReleaseID> here is a release label, set to R2018a at this writing.

The test program drives a collection of collocation test problems selected to exercise core CSALT functionality, generating two output files per test problem. The first output file is the output generated by the Optimizer running at the core of CSALT, SNOPT. The SNOPT output has the suffix “Data.txt” appended to the name of the test problem run. The second output file is the CSALT generated optimization control history file, with the suffix “.och”.

Testing CSALT is a three step process:

1. Build the library and tester.
2. Generate test data.
3. Run the system tester.

The build process is driven by a CMake configuration file found in the top level CSALT directory. The default settings in this file generate the test application in a folder outside of the CSALT file folders named LowThrustBinary. Test data is generated in the test/run folder inside of that folder. The resulting data files are compared to data generated from a previous runs, stored in the test/truth folder of the LowThrustBinary folder.

Building CSALT

Build instructions for CSALT are provided separately, in the document titled “Building CSALT,” located in the Docs folder of the CSALT repository.

Generating Test Data

Now that the CSALT test application has been built, the test data needs to be generated. This step can be performed either by running the test program interactively or from a command line launch. The test data files will be generated in the folder from which the program is run. The following capture of commands (boldfaced) from a terminal run demonstrate the process. Terminal responses are italicized in this text.

$ **cd LowThrustBinary**
$ **ls**

bin lib test

$ **cd test**
$ **mkdir run**
$ **cd run**
$ **../../bin/TestCSALT_R2018a -run All -exit**

Running 'All'
Tue Sep 5 09:17:08 2017
**** Running the Brachistochrone CSALT problem ****

…

Time for solution output 0.00 seconds
Time for constraint functions 0.00 seconds
Time for objective function 0.00 seconds
**** END Hull95 TEST ****
Total run time: 23.331 sec

$

Testing CSALT

The CSALT system tester is a Python script named, TestCSALT.py. That script is used to compare CSALT test output data to accepted[1] values. The current implementation compares SNOPT and OCH output data.

TestCSALT uses a comparison file located in the same directory as the script and named Comparison.txt. This file is a two-column, tab- or space-delimited file. The first column contains test

1 CSALT accepted values are output data files that have been reviewed and deemed correct.

names, and the second contains target Objective values. This file is used for all SNOPT file comparisons to determine the accepted objective function value for the test. The file is used in OCH comparisons when the test system run is set to compare “All” test cases, telling the system the names of each test case to be run.

TestCSALT has several command line arguments. If run without command line arguments, or with arguments that are not parsed correctly, it will display usage text. To run TestCSALT, it must have arguments identifying the type of test to run and the test case. Invocation in test mode is made using the syntax

python3 TestCSALT.py ***[Type] [Test]***

Several values used in the test run can be changed by adding additional command line arguments to the end of the command invocation, as shown in the following table.

Command Line Argument	Description	Default Value
-compFile [Path]	The full path of the comparison file.	./Comparison.txt
-runpath [Path]	The path to the directory containing CSALT test data output (Test.och for OCH files, TestData.txt for SNOPT files)	./run/
-truepath [Path]	The path to the directory containing OCH .truth files (useful for OCH only)	./truth/
-tolerance [value]	Sets the comparison tolerance	0.0005

Example: Running All Test Cases

Open the directory containing TestCSALT and the truth and run folders in a terminal. Run the following to compare OCH data for all of the test cases in the comparison file:

python3 TestCSALT.py OCH All

or, to compare objective function values,

python3 TestCSALT.py SNOPT All

Both sets are run using the command

python3 TestCSALT.py All All

Example: Running A Specific Test Case

Open the directory containing TestCSALT and the truth and run folders in a terminal. Run the following to compare OCH data for the Brachistochrone of the test cases in the comparison file:

python3 TestCSALT.py OCH Brachistochrone

or, to compare objective function values,

```
python3 TestCSALT.py SNOPT Brachistochrone
```

or, for both sets,

```
python3 TestCSALT.py All Brachistochrone
```

Appendix: Creating a Comparison File

The CSALT test code includes a Python script that can be used to generate the Comparison.txt file from the data in the test/truth folder, and a second Python script that can be used to populate the truth folder from a run and generate the corresponding Comparison file. Those utilities are described here.

Generating a Comparison File from a Truth Folder

The script SetupCompare.py builds a comparison file named NewCompare.txt from the files in a truth folder. It is run as follows:

```
python3 SetupCompare.py
```

This script is run form the folder containing a folder named truth, which contains the results of a test system run. If the truth folder is missing or empty, the resulting NewCompare.txt file will be empty.

Generating Truth Files from a Run

The script SetupTruth.py reads the data files in a run folder, uses those files to populate a truth folder, and then generates a comparison file (“NewCompare.txt”) from the contents of the truth folder. This utility is useful for adding new test case run data to the truth folder; it replaces identically named files in the truth folder with the new test case files, preserves other existing test case run data, and produces a comparison file for all of the files in the truth folder. It is run as follows:

```
python3 SetupTruth.py
```

The script copies the run data files to the truth folder, renames files as needed, and then generates the NewCompare.txt file using the same mechanism used in SetupCompare.py.