RHEOS Testing - A Brief Overview

J Louis Kaplan

Abstract

This document is a short guide into how the RHEOS testing procedure works. Many aspects of it also apply to the testing procedures of other Julia packages – the process is fairly standardised. The test creation and running process can be summarised by the following. Tests are written in the file 'RHEOS/test/runtests.jl'. They consist of code blocks that return `true` or `false` depending on whether or not the test has passed, and these code blocks are prefaced by the `@test` macro. For readability, tests longer than one line of code should be encapsulated in functions. For organisational benefit, the tests are generally split off into one test file per source file, with the test files being included from the main 'runtests.jl' script. The tests can then be run locally, and will automatically run remotely after running `git push`.

1 Local Testing

1.1 Directory Structure

Everything testing related lives in the 'RHEOS/tests' folder. Note that in the test folder there is one file per file in 'RHEOS/src' and an additional file called 'runtests.jl'. Having one test file per source file is purely for organisational purposes. The only file (and filename) of special significance is the 'runtests.jl' file. This is the Julia file that will be run when we type `test RHEOS` in the Julia REPL's `pkg` mode. It is therefore also the file that will be run remotely by Travis CI and Appveyor when they run the tests on their Linux and Windows servers respectively.

1.1.1 Writing a Test

Ideally, each function in the source code should have at least one test. For a clear organisation, any test that requires more than one simple line of code should be written as a function. The line of code / function test should return a boolean value of `true` or `false` depending on whether the test passed or not. Then the line of code / function call should be prefaced with an `@test` macro. Let's take a look at an example:

```
function _derivCD(tol)
```

```
    x = Vector(0.0:0.001:1.5)
    y = x.^2
    dy = 2*x
    dy_numeric = RHEOS.derivCD(y, x)
    isapprox(dy_numeric, dy, atol=tol)
end
@test _derivCD(tol)
```

This function tests the central difference derivative function in RHEOS. Notice that although it is an approximate test it still returns a discrete boolean true/-false value indicating whether or not the result is within a prescribed absolute tolerance. (In this case the tol used is defined as a global constant earlier in the source file.)

1.1.2 Running Tests Locally

To run tests locally you can either go into `pkg` mode and type `test RHEOS`, or you can navigate into the 'RHEOS/tests' directory, open up a Julia REPL and type `include("runtests.jl")`.

If a test fails, an error will be generated.

2 Remote Testing

The RHEOS github repository is set-up with Travis CI and Appveyor to test on their Linux and Windows servers respectively. If any of the tests fail, this will be indicated on the appropriate badge in the github repository readme page.

If all tests pass locally, but fail on either Travis CI or Appveyor you will need to investigate. To do this you should log in to either Travis CI (https://travis-ci.org/) or Appveyor (https://www.appveyor.com/) and find the build whose test has failed. This will show the terminal window, the same as would be displayed as if you had the tests locally, so you can identify which test has failed.