Report week 8 - INF3331

Yrjan Skrimstad (yrjansk)

October 31, 2015

1 Implementation

1.1 Files

- heat_equation.py: Contains the basic python solver and the plotting function.
- heat_equation_numpy.py: Contains the numpy solver.
- heat_equation_weave.py: Contains the weave solver written with use of inline C.
- heat_equation_ui.py: Contains the user interface for the heat equation solver.
- test_heat_equation.py: Contains the tests.
- report.tex: The report you're reading right now.

1.2 Details

heat_equation.py: Contains the basic solver **heat_equation()**. This is simply a solver that iterates through the timesteps and executes the given function for every single point in the array.

The plotting function **heat_equation_plot()** is a fairly simple plotting function that takes the equation parameters and calls the given solver (it defaults to the basic python solver) to solve the equation and then plots the data.

heat_equation_numpy.py: Fairly simple solver called **heat_equation_numpy()**. The solver uses vectorization and is therefore quite a lot faster than the python-version.

heat_equation_weave.py: Contains the fastest solver I've written, **heat_equation_weave()**. This solver is written with inline C and pointer swapping to avoid unnecessary copying of the data. There's several type conversions to make sure the inline C-code behaves as expected. The C code returns a boolean to tell us which array is currently the new array, this is necessary because of the pointer swapping.

heat_equation_ui.py: Contains a user interface for solving and plotting equations. This is written as a fairly basic python script without unnecessary functions and other fanciness. It supports quite a few options, run the script with the *-h* flag to see them.

test_heat_equation.py: Contains two tests. One test for testing if the solver is within the error specified by the assignment text. And another test to test if the error decreases as size increases. To run the tests run 'py.test' in the folder of the assignment. **Note:** This requires **py.test** to be installed on the system.

2 Runtime comparison

2.1 Test system

Debian Linux - Python 2.7.10 - Intel i5-2450M CPU @ 2.50GHz - 8GB RAM

2.2 Timings

These timings are taken from runs with the example data as given by the assignment text.

Time in seconds	Solver implementation
77.510	Python
1.082	NumPy
0.217	Weave, inline C

We here see that the pure Python-implementation is definitely the slowest, it's actually slower than the inline C-implementation by a factor of almost 400. This makes it practically unusably slow in a lot of situations.

The NumPy-implementation is quite fast, and actually usably fast. However it's still a lot slower than the inline C-implementation.

The Weave/inline C-implementation is definitely the fastest. It's a lot faster than the other implementations and would therefore be recommended for use with any larger datasets.