Melon - a Task Scheduling Package for Personal Todo Lists

using Markov Chain Monte-Carlo Methods

An MMSC Special Topic on Python in Scientific Computing

Candidate Number: 1072462

Abstract

In this project report we will review the central concepts utilised in the group work conducted to make progress in the Partial Differential Equation (PDE) problem associated with the electrochemical model of a battery cell and present numerical results.

Our Goal: Numerically obtain the solution $\{a(x,T), b(x,T)\}$.

The Finite Difference schemes are implemented in Julia and Python, whereas the Spectral Method is implemented in C++.

Figure 1: The Graphical User Interface (GUI) of the Spectral Solver.

1 Problem Introduction

UIDs are useful because they make collisions very unlikely, which is not to say that these should not be checked, but if two clients are connected that each generated a set of UIDs it is very unlikely to have to do conflict resolving.

We recommend usage with `xandikos`, a version-controlled DAV server, capable of syncing calendars (events, todos and journals) and contacts.

Published on PyPi.

1.1 Usage

Use `invoke -l` to list all available tasks.

Is platform-independent, for example due to the usage of `pathlib.Path`.

2 Code Quality

2.1 Formatting

2.2 Docstrings

2.3 Documentation

2.4 Tests

2.5 Coverage

2.6 Type Checking

Using `pyright` instead of `mypy` as it is much faster.

2.7 Using Appropriate Language Features

Using `autoflake` and `pyupgrade`. Used `logging`.

2.8 Maintaining Code Quality

Using `pre-commit` and GitHub Actions CI/CD. Uses `invoke` to manage common development tasks.

```
import numpy
x = 5
print(x ** 2)
```

3 Runtime Performance

```
In [1]: %timeit str(t.icalendar_component["uid"])
  122 µs ± 1.06 µs per loop (7 runs, 10,000 loops each)
In [2]: %timeit t.vtodo.contents["uid"][0].value
  355 ns ± 7.14 ns per loop (7 runs, 1,000,000 loops each)
In [3]: %timeit
 ↪  t.vobject_instance.contents["vtodo"][0].contents["uid"][0].value
  296 ns ± 7.06 ns per loop (7 runs, 1,000,000 loops each)
In [4]: %timeit
 ↪  t._vobject_instance.contents["vtodo"][0].contents["uid"][0].value
  208 ns ± 23.7 ns per loop (7 runs, 10,000,000 loops each)
```

Table 1: Profile obtained by running `./main.py --profile | grep todo.py`.

16958	0.008	0.000	0.939	0.000	todo.py:36	vtodo
32475	0.047	0.000	0.705	0.000	todo.py:96	uid
856	0.003	0.000	0.579	0.001	todo.py:26	upgrade
117	0.000	0.000	0.489	0.004	todo.py:111	priority
417	0.001	0.000	0.461	0.001	todo.py:121	isIncomplete
5512	0.003	0.000	0.278	0.000	todo.py:45	summary
856	0.002	0.000	0.112	0.000	todo.py:21	___init___
1363	0.006	0.000	0.024	0.000	todo.py:164	___lt___
7844	0.004	0.000	0.009	0.000	todo.py:61	dueDate
2605	0.001	0.000	0.003	0.000	todo.py:85	dueTime

Acronyms

GUI	Graphical User Interface	1
PDE	Partial Differential Equation	1