Differential Equation Solver

By Thomas DeWitt

What is it?

The code `diffeq.py` numerically solves an arbitrary system of coupled differential equations

$$\begin{cases} \frac{d\phi_1}{dt} = f_1(\phi_1, \phi_2, ...\phi_n) \\ \frac{d\phi_2}{dt} = f_2(\phi_1, \phi_2, ...\phi_n) \\ ... \\ \frac{d\phi_N}{dt} = f_N(\phi_1, \phi_2, ...\phi_n) \end{cases} \tag{7}$$

by discretizing them into time steps Δt such that

$$\phi_i^{t+\Delta t} = \phi_i^{\Delta t} + f_i(\phi_1, \phi_2, ...\phi_n)\Delta t. \tag{8}$$

Usage guide

The function `diffeq.solve()` takes 3 required parameters

- `ic`: Array-like list of initial conditions $\phi_1^{t=0}, \phi_2^{t=0}, ..., \phi_n^{t=0}$.
- `func`: Python function that takes a single argument $[\phi_1, \phi_2, ..., \phi_n)]$
  and returns $[f_1(\phi_1, \phi_2, ..., \phi_n), f_2(\phi_1, \phi_2, ...\phi_n), ..., f_N(\phi_1, \phi_2, ..., \phi_n)]$. Output must be a `np.ndarray` with `dtype=dtype` (see below).
- `dt`: Δt
- `end_time`: How long to simulate for.

and 3 optional parameters

- `save_data`: Whether to save variables $\phi_1^{t_i}, \phi_2^{t_i}, ...\phi_n^{t_i}$ for $t_i \in \{\Delta t, 2\Delta t, ..., (n-1)\Delta t\}$. If `True`, return a 2-D `np.ndarray` where column 0 is t_i, columns [1:] are the variables $\phi_1^{t_i}, \phi_2^{t_i}, ..., \phi_n^{t_i}$, and each row corresponds to a time step t_i. If False, return only $[\phi_1^{t_{end}}, \phi_2^{t_{end}}, ..., \phi_n^{t_{end}}]$. Default `False`.
- `jit`: Whether `func` is jitted using `numba.njit()`. `jit=True` speeds computation for many time steps. Default `False`.
- `dtype`: `dtype` of all arrays. Default `np.float32`.

Example

Consider the system

$$\frac{d^2x}{dt^2} = -10x \tag{9}$$

which can be turned into the system

$$\frac{dx}{dt} = y \tag{10}$$

$$\frac{dy}{dt} = -10x. \tag{11}$$

This can be implemented as a Python function like

```
def f(vars):
    x,y = vars
    return np.array([y,- 10 * x], dtype=np.float32)
```

and

```
solve([1,0], f, .0001, 100, save_data=True)
```

produces the expected sine wave.