basicfunctions.py docs

Heather Y.

June 18, 2018

1 Overview

This is a functions-only module meant to be imported into simulations so that operations act like an analog computer's operations. Imported into module-ion.py (see relevant docs) to rewrite ion-sim-timings.py (see relevant docs).

2 Dependencies

This program uses the built-in math module, pigpio, and the reduce function from functools.

3 Structure

Most of the documentation for the functions in this program can be found in the functions themselves as well as in the overall documentation. (Note: most of this documentation does not show up using the command line's help() utility because it is hidden by the decorator and wrapper functions.)

Pigpio is used for the analog_output() function, the math module is used for the sqroot() and logalog() functions, and the reduce function from functools is used for the mult_summ() function.

The in-file documentation, however, does not cover the decorator() function, the integrate() function (for this see the overall documentation) or the derivative() functions. The decorator() function is, as the name implies, a python decorator which is applied before every "operation" (add, subtract, integrate, etc) function. It checks if chop_flag is true, and if so, applies the chop_num() function to the output to attempt to reduce floating point arithmetic errors. The chop_num() function is further explained in the error-chop.py docs.

The derivative() function is rather simple compared to the integral function. It takes in a list of (two) values, subtracts them, then divides them by the time step to find the instantaneous slope. While there are more accurate methods to numerically differentiate, they require knowledge of the function.

4 Functions and Parameters

- chop_num() - This function is used purely by a decorator function, decorator(). Neither of these function should ever be explicitly called by the user. However, chop_num() can be turned on or off. To turn on, change the chop_flag variable on line 9 of the code to "True". To turn off, change to "False". The chop_num() function will then be called (or not called) automatically using the decorator() function. For this reason, don't ever call decorator() or chop_num() from outside the module. If you add a new function to the module, right before the function definition statement, add the line of code "@decorator" if it has a numerical output.

- add() - Takes in a minimum of two values and a maximum of five values and returns their sum.

- multiply() - Takes in a minimum of two values and a maximum of five values and returns their product.

- subtract() - Takes in two values and returns their difference. The second input is subtracted from the first.

- divide() - Takes in two values and returns their quotient. The second input is the divisor and the first the dividend.

- square() - Takes in one value and returns that value raised to the second power.

- power() - Takes in two values and raises the first value to the power of the second value.

- sqroot() - Takes in one value and returns the square root of the absolute value of that number (therefore, if you wish to compute with complex values, you must import cmath and change math.sqrt() to cmath.sqrt(), though if you wish to compute with both real and complex values, it is recommended you insert an if statement to ensure nothing is a type that is not expected.

- logalog() - Takes in one value and returns the common log of that value. Name is a reference to Redwall; feel free to change to log().

- summation() - Takes in a list of values and returns their sum using the python built-in sum().

- mult_summ() - Takes in a list of values and returns their product. If you are not using this function, you do not need to import functools.

- integrate() - Takes in the current time step, the change between time steps, an array of (a minimum of) three values that represent the new inputs to the function, the past integral value, a flag called "typ", which indicates whether the trapezoidal, simpson, or simpson 3/8ths method should be

used for calculating the integral, and the initial value of the integral. See the Overall Documentation for a full description.

- derivative() - Takes in an array of (a minimum of) two values and the change between time steps. More accurate derivative functions require knowledge of the function, meaning they cannot be used.

- super_integrate() - Takes in the same values as integrate() but uses a fifth order Newton-Cotes method as described by the paper *Cumulative High-Order Newton-Cotes Algorithms* written by the same authors as of this package.

- pwm_map() - Should not be called by the user; instead, call analog_output(). Takes in the value to be scaled, the minimum and maximum possible values, and the resolution of the pwm.

- step_vary() - Does not currently work with the simpson method for integrating. Takes in all arrays of integrated data (up to four) and the "tolerance" for change of these values. If there is enough change, it increases or reduces the step size. See the Overall Documentation for a full description.

If you wish for another function to be added, make sure you add the decorator function before the definition line, using the syntax "@decorator" and make note of what the function requires and inputs and what the function returns. (You can also contact the programmers of this program for help.)