Structure of User Defined Control Functions in Python Power Electronics

Shivkumar .V. Iyer

1 Introduction

This short paper will describe the facilities available to the user for designing controllers in Python Power Electronics. As a circuit simulator for power electronics, it is critical that the simulator provide users the ability to design control in a way such that they would mimic how they would behave when implemented in hardware. For this purpose, a number of special variables are available within a control function. This short paper will describe the need and effectiveness of each of these special variables.

Table 1: Control files in circuit_inputs.csv

Name of control files	control1.py	control2.py

2 Control hierarchy

In Python Power Electronics, users can write control functions as Python (.py) files. A user can include any number of control files in a simulation but these files need to specified in the simulation parameters spreadsheet circuit_inputs.csv. The relevant row from the spreadsheet circuit_inputs.csv is in Table 1. In this example, it is assumed that the simulation has two control files control1.py and control2.py. These two control files are specified in separate cells on the

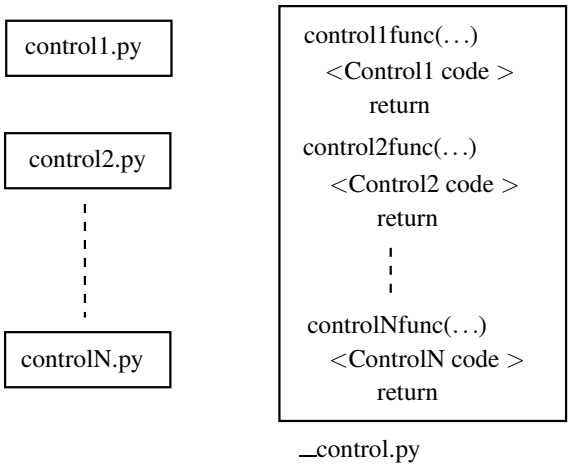

Figure 1: Inclusion of control functions in the simulator

same row and any additional control files will need to be added in individual cells in the same row. The circuit simulator will include all the control files specified in circuit_inputs.csv and found in the local directory into a special file called __control.py. It is essential that the user not define a control file by this name as the simulator creates a special file with this name. The process of inclusion of control files can be depicted by Fig. 1. The simulator creates a function for each control file defined by the user by appending a "func" to the control file name. For example control1.py becomes control1func. The code included by the user in this file control1.py will be copied into the function control1func as shown in Fig. 1. With all the user control functions now copied into __control.py, the simulator can execute all these control functions when needed.

In order to understand the need for the special variables to be described in the next chapter, let us list out the requirements that most of us would expect when designing a controller.

1. Inputs - As would be fairly obvious, any controller would need to accept in the form of inputs measured quantities from the circuit. Any meter in the circuit would be a potential input to a controller. Currently the simulator has the Ammeter and the Voltmeter as meters to measure current and voltage respectively.

2. Outputs - Also, fairly obvious is that in most cases controllers would need to change a circuit in some form as a result of an algorithm. Such a change in the circuit could be a change in the output voltage of a controlled voltage source, a change in the conduction

state of an ideal switch or a change in the resistor value of a variable resistor. The simulator supports a number of components that can change their state upon receiving commands from control functions.

3. Stored Variables - As stated before, the simulator embeds the code in each control file into a function by a similar name. In Python, there is no need to define a variable as there is in in a programming language like C. A variable comes into existence when it is used for the first time. However, if a user creates a control function that has a number of variables that are used as simple Python variables, these will be created in the control functions and will exist only within that control function. In many cases, control functions need to store the value of certain variables. Examples of such cases include integrators, filters and differentiators. For this reason, there was the need to create special variables whose values are stored by the simulator and are available in the next iteration of the control function. These variables are called StaticVariables and they are not volatile and destroyed when the control function is executed. However, these variables are local to the control function and can only be used within the control function where it has been defined.

4. Execution instants - Controllers designed using these control functions are typically discrete time controllers. Therefore, the frequency of their execution is extremely important. Additionally, in some cases, the next time instant when a controller needs to be executed is only known as a result of a calculation in the present instant. To ensure execution of control functions at their desired time, the simulator offers a special variable called TimeEvent. When a Time Event variable has been defined and updated in the control function, the simulator guarantees that the control function will be executed at the time instant corresponding to the TimeEvent. Therefore, this TimeEvent variable offers a dual advantage to the user - it allows the user to set the time intervals at which the control function is executed and also it is the interface to the timing scheduler of the circuit simulator informing it when it must execute the control function next.

5. Global variables - The first two items in this list dealt with inputs and outputs to the control

functions. However, inputs were measured quantities from the circuit while outputs were to controlled elements in the circuit. Essentially, inputs and outputs were from and to physical quantities in the circuit. However, an input can also be from another control function. Similarly, an output can be to another control function. A last option is that a control variable may be for the sole purpose of being written to the output data file so that it can be plotted with respect to time. To make all these possible, the VariableStorage type has been provided for control functions. A VariableStorage type is global to all control functions. Therefore, once defined it can be accessed and altered in any control file and its contents are stored by the circuit simulator for future availability. This is in contrast to the StaticVariables that are defined for a control function and are local to that control function only. By using these variables, control signals can be exchanged between control functions and therefore, one control function can accept as inputs control variables updated by another control function. Additionally, these VariableStorage types can be written to the output data file.

3 Control files and descriptors

The previous section described the different variables available to the user for designing controllers in Python Power Electronics. This section will describe how the above variables are defined and used. As stated in the previous section, Python does not need variables to be declared. Therefore, if a variable does not have to perform any of the functions listed in the previous section, they do not have to be defined separately. However, these variables will be volatile and will exist only while the function is being executed. In order to define the special variables, a descriptor is used. A descriptor for a control function is a spreadsheet with a "_desc.csv" appended to the name of the control function. Therefore, control1.py will have a descriptor by the name "control1_desc.csv". Let us consider a descriptor for a sample control file as shown in Table 2.

Table 2 is just an example containing all the types of special variables available in control function. Let us describe each row as follows:

Table 2: Sample descriptor

Input	Element name in circuit spreadsheet = Ammeter_A1	Desired variable name in control code = curr_input		
Output	Element name in circuit spreadsheet = ControlledVoltageSource_Vin	Control tag defined in parameters spreadsheet = Vsource	Desired variable name in control code = Vsource	Initial output value = 0
Static Variable	Desired variable name in control code = curr_integ	Initial value of variable = 0.0		
Variable Storage	Desired variable name in control code = plot_variable1	Initial value of variable = 0.0	Plot variable in output file = yes	
Time Event	Desired variable name in control code = t1	First time event = 0.0		

1. The first row describes an input variable. This construct connects a measured quantity in a circuit (for now either current or voltage) and copies it to a variable in the control function. The second column asks the user for the name of the element in the circuit (note that the entire name has to be provided as it is in the circuit). The third column asks the user for the desired variable name in the control code as this will be variable the measured quantity of the circuit element will be copied to. In this specific case, the current measured by Ammeter_A1 will be copied to the variable curr_input. When the user writes code

referencing curr_input, indirectly the current measured by Ammeter_A1 is used.

2. The second row describes an output variable. As with the case of inputs, it connects a circuit component to a variable in the control code. In this case, the element name in the circuit spreadsheet is ControlledVoltageSource_Vin. This element is controllable and therefore has a control tag. The control tag is the name of the control input to this element and should be unique to the element. The third column asks the user to specify the control tag that the control function is modifying. The fourth column is the name of the variable by which this control action should be performed. In this particular case, any change made to the variable Vsource in the control function will directly be reflected in the control tag with the name Vsource in the element ControlledVoltageSource_Vin. This variable has the default initial value of 0.0 but can be changed to anything by the user.

3. The third row describes a StaticVariable type. It merely defines the variable whose value when altered by the control function should be stored by the simulator. This variable has an initial value which by default is 0.0 but can be changed by the user.

4. The fourth row describes a VariableStorage type. Similar to the static variable, it defines a variable which can be accessed by any control file in the simulation. This variable has an initial value. Besides, there is the last field that asks the user if the value of this variable should be written to the output file. If the answer is yes, the variable will be stored and can be plotted which is useful in debugging. When the answer is yes or no, the variable is made available in all control files.

5. The last row describes a TimeEvent type. This tells the simulator that a variable with that name decides the next time instant when the control function must be evaluated.

To support the above explanation, consider the block of code below. This code is merely an example, it will be never as it is below perform any meaningful control action.

```
curr_integ += curr_input*dt
Vsource = 100*curr_integ
plot_variable1 = curr_integ
```

In the above code, the current measured by the Ammeter_A1 and contained in curr_input is integrated by adding the product of the current with the time step to the existing value of the static variable curr_integ. This integrated value of current multiplied by 100 is made the value of ControlledVoltageSource_Vin which is contained in the variable Vsource. To fine tune the controller, the integrated value of the current is written to the output data file. In this case, there is only one control function and therefore the VariableStorage type plot_variable1 does not perform the role of a global variable. However, if there existed another control function, plot_variable1 could have been accessed (read and modified) in that file thus passing the integrated value of current to another control function.

The last variable type will now be described. The time event t1 is to decide the time instant when this block of code will be executed. Consider the code:

```
if t_clock>=t1:
        <Code above>
        t1 += 100.0e-6
```

The variable t_clock is a built-in variable that contains the current simulation time instant. This is provided and updated by the simulator and can be used for any purpose in a control function. However, this variable is read-only. It is only for a control function to know the present time instant of the simulation. By comparing t_clock to the TimeEvent t1 and executing the above block of code when t_clock exceeds t1, it is ensured that the control code will be executed only at time instant t1. Moreover, when the code block is executed, the time event t1 is updated. In the above case, it is incremented by 100 μs which implies that the control code will be executed again after 100 μs.

4 Conclusions

This paper described briefly how a user can write control functions in Python Power Electronics. Special variables provided in the control functions allow users to design fairly complex code and ensure that they are executed at the desired time instant. Using a fairly simple example, the

method of using the special variables has been demonstrated. This combines the flexibility of discrete control and the accuracy of analog control and therefore, can mimic a hardware implementation of a power converter circuit. This short paper is an extract from Chapter 4 of the book on circuit simulation to be released next year. For the table of contents, please visit the website and check out the documents download link.