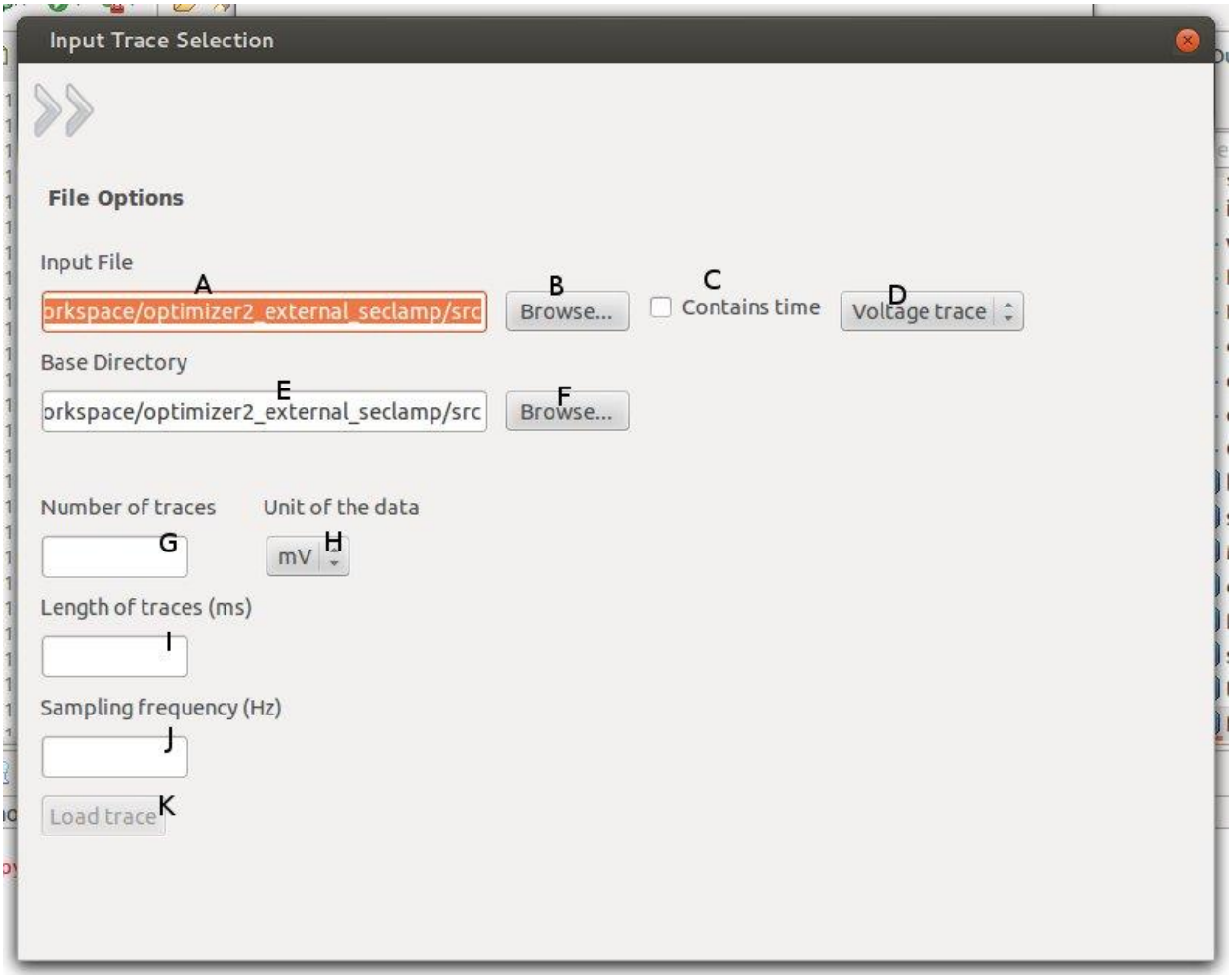
Optimizer tutorial

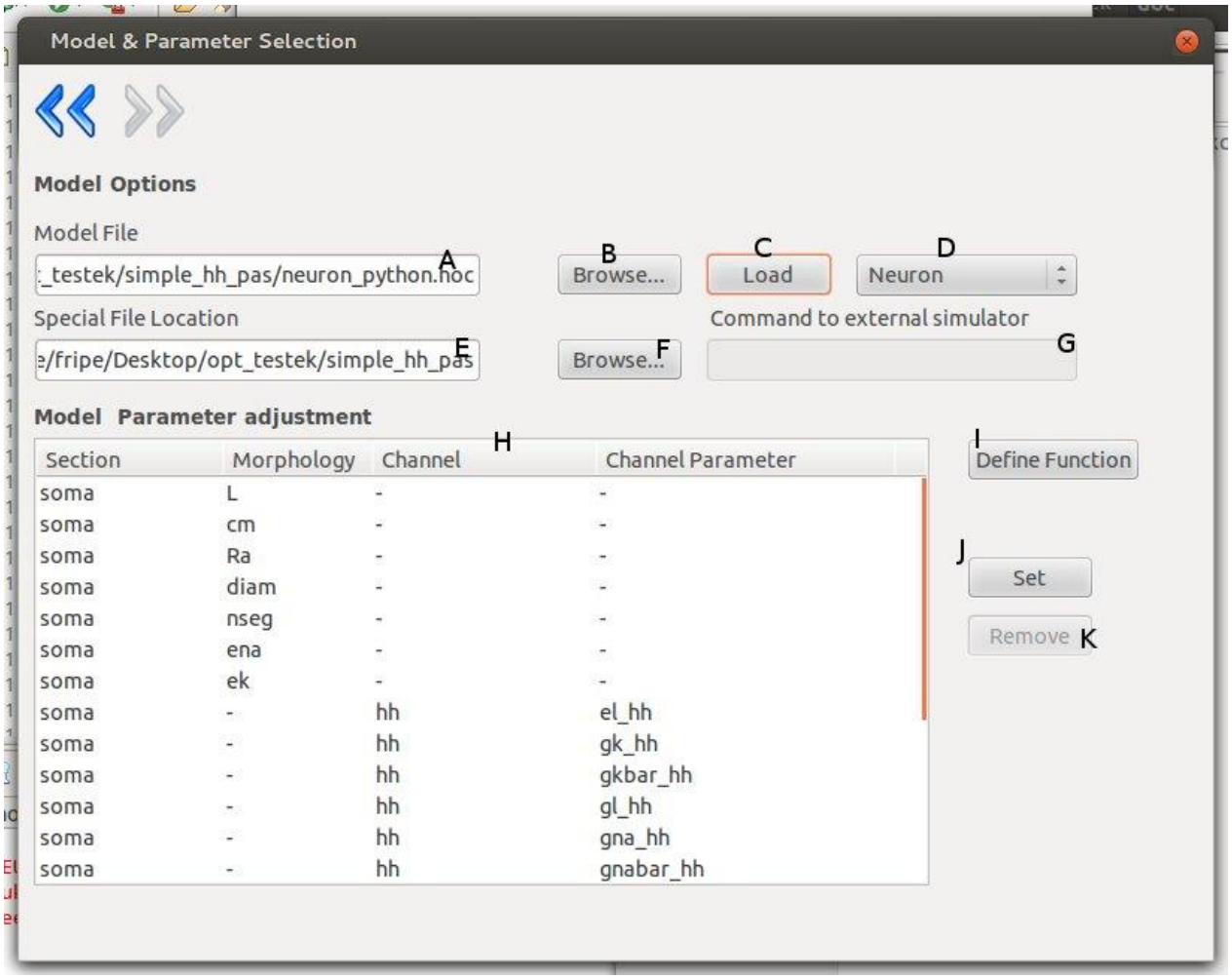
Layer 1



A	Path to the given input file
B	Through this you can select the input file
C	Indicates if the trace containes the time steps as well
D	Specify the type of the input: voltage, current spike times, other
E	Path to the base directory, this directory will contain the results
F	Through this you can select the base directory
G	The number of traces in the file. It expects the values in columns
H	The unit of the data (for conversions)
I	The length of the traces
J	Sampling frequency of the traces
K	With this you can load the contents of the given file. You can only load one file, if you want to have

	multiple input traces, you have to put them into one file. If the type is set to spike times it will load the given file in addition (won't overwrite the other)

Layer 2

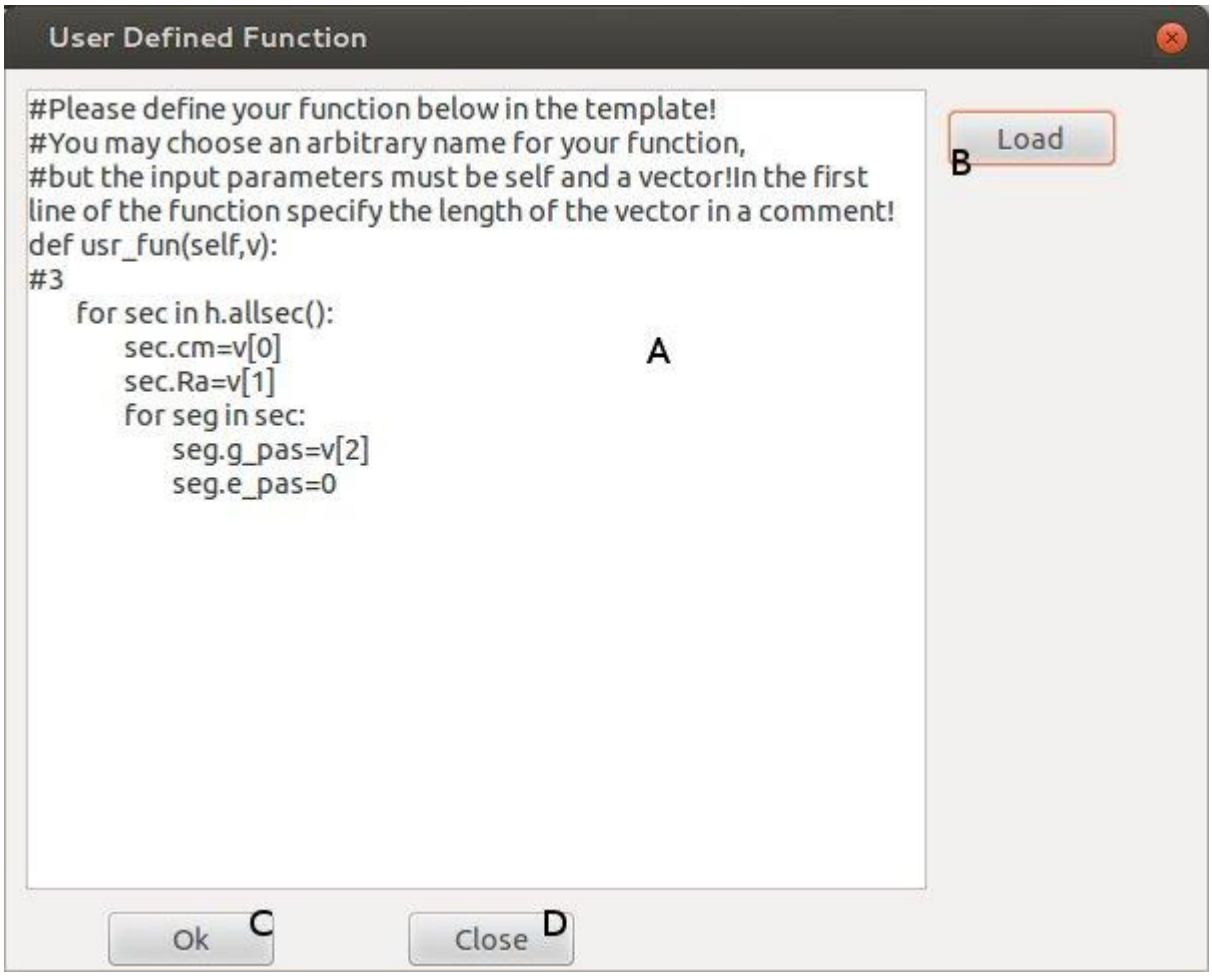


A	Path of the model file. The model file must contain only the model definition. Curently we are supporting .hoc files only. For other file types use the external option (see D)
B	Through tis you can select the model file
C	Loads the model. If the loading was not successful, or you selected the wrong model, you must restart the program.
D	Simulator selector. For a simulator other than neuron, select the external
E	Path to the additional files required by the model (.mod files)
F	Through this you can select the directory containing the special files
G	Here you can specify a command for the external simulator. The command mus have the following format: executable model_file options number_of_parameters. E.g: „nrniv test_model.hoc 2" will run the model in test_model.hoc, using neuron, optimizing 2 parameters specified in the model file. The model file in this case must interact with the software: it must read the parameters from a file „params.param" and put its input to a file named „trace.dat". The trace.dat must contain the time

	steps as well.
H	This table serves as the parameter selector. After loading the model, here you can see every mechanisms that the model contains.
I	If you want to optimize a combination of parameters or there is an additional task apart from optimizing some parameters, here you can do this by writing a function using the hoc+python syntax (see next section)
J	With this you can set the selected parameter to be optimized. The selected parameter will turn to red. You can't select a parameter mor than once.
K	With this you can remove the selected parameter from the list of parameters to be optimized.

Layer 2.1

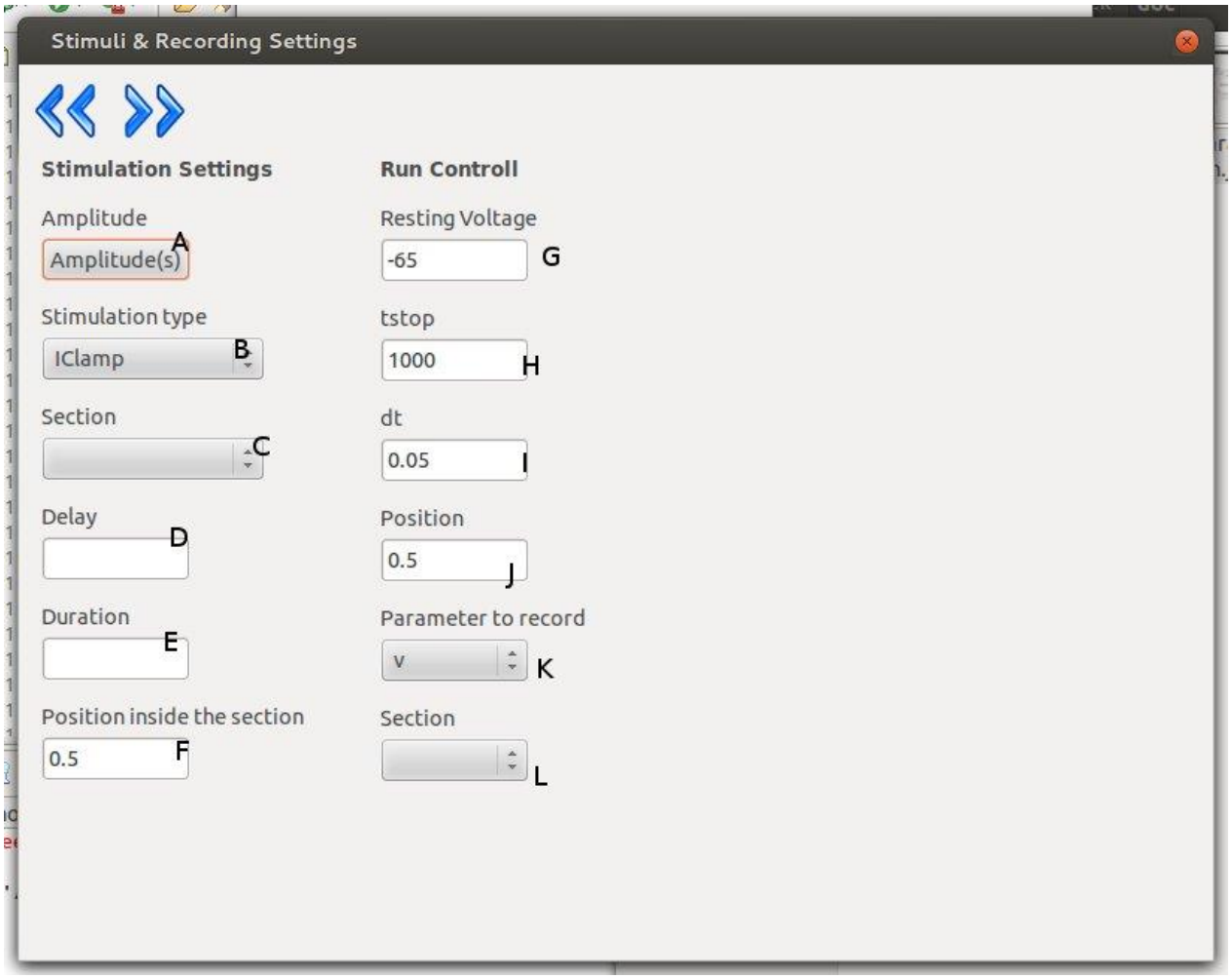
This section demonstrates the user defined function option (acces it I on layer2).



A	Here you can define your function. As you can see in the provided example you can use python here. In this example we had many sections and we wanted the specific parameters to be the same over the sections, thus we had to write this little piece of code. You can access hoc functions thorugh the „h" object, like h.allsec(). You can either use this function, or select the parameters one by one on layer2, you can not combine the two methods. If you are using this option, your parameters will be named parameter0, parameter1,…parameteri according to their order in the input vector.

B	If you use a function often, or doesn't want to write it every time, you can put it in a txt file and load it.
C	Store the function specified in the input field. After pressing this, the parameter selector table on layer2 will become empty.
D	Closes the window and returns to layer2

Layer 3

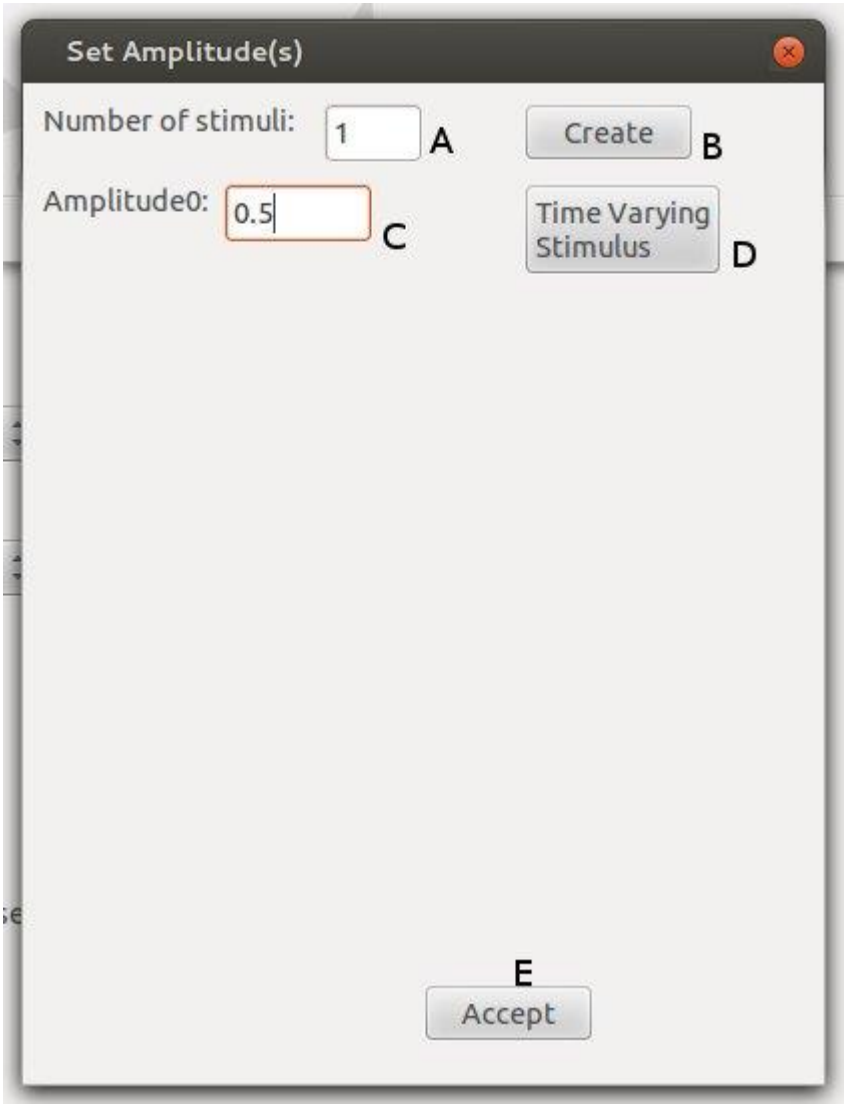


A	Here you can specify the stimuli amplitudes
B	Sitimulation type: IClamp,VClamp: Current clamp from neuron, seclamp form neuron
C	The section which receives the stimulus
D	Delay of the stimulus in msce if IClamp is used or the resistance in mOhm if VClamp is used.
E	Duration of stimulus in msec
F	Postion inside the section (0-1) where the stimulus is received
G	Resting membrane potential
H	Length of simualtion in msec
I	Integration step: must be smaller or equal to the reciprocal of thesampling frequency of the input

J	The position inside the section where the recording takes place
K	The variable to be recorded during simulation: current or voltage
L	The section where the recording takes place

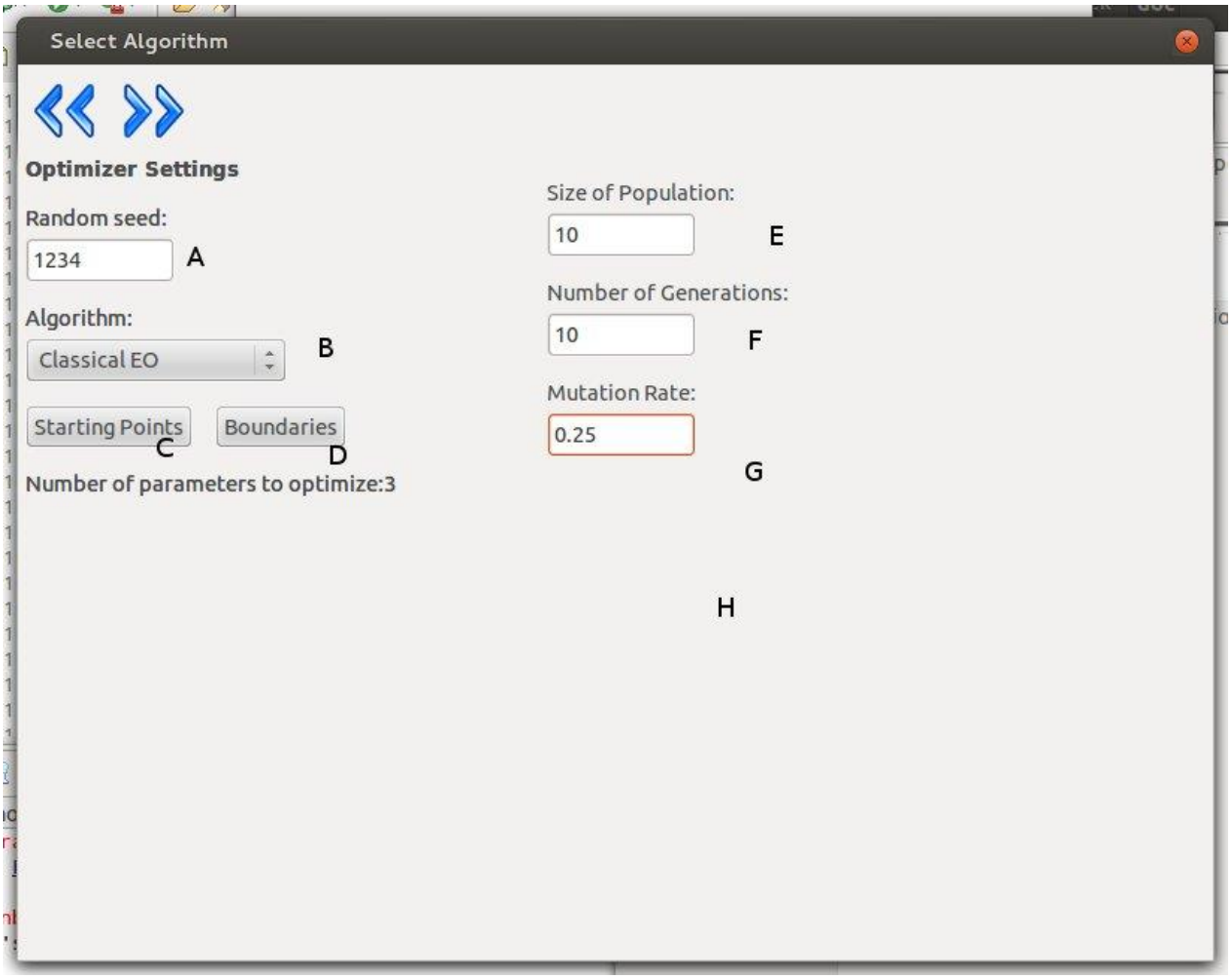
Layer 3.1

This window can be opened by „A” on layer 3.



A	The number of stimuli used during the stimulation. It assigns each stimuli to a trace, so setting this valu to e.g 4 means there must be 4 input trace.
B	Pressign this will create the input fields to the given number of stimuli.
C	The amplitude values in nA
D	Here you can load a time sries from a txt which will be used as stimuli
E	Accepts the settings and returns to layer3

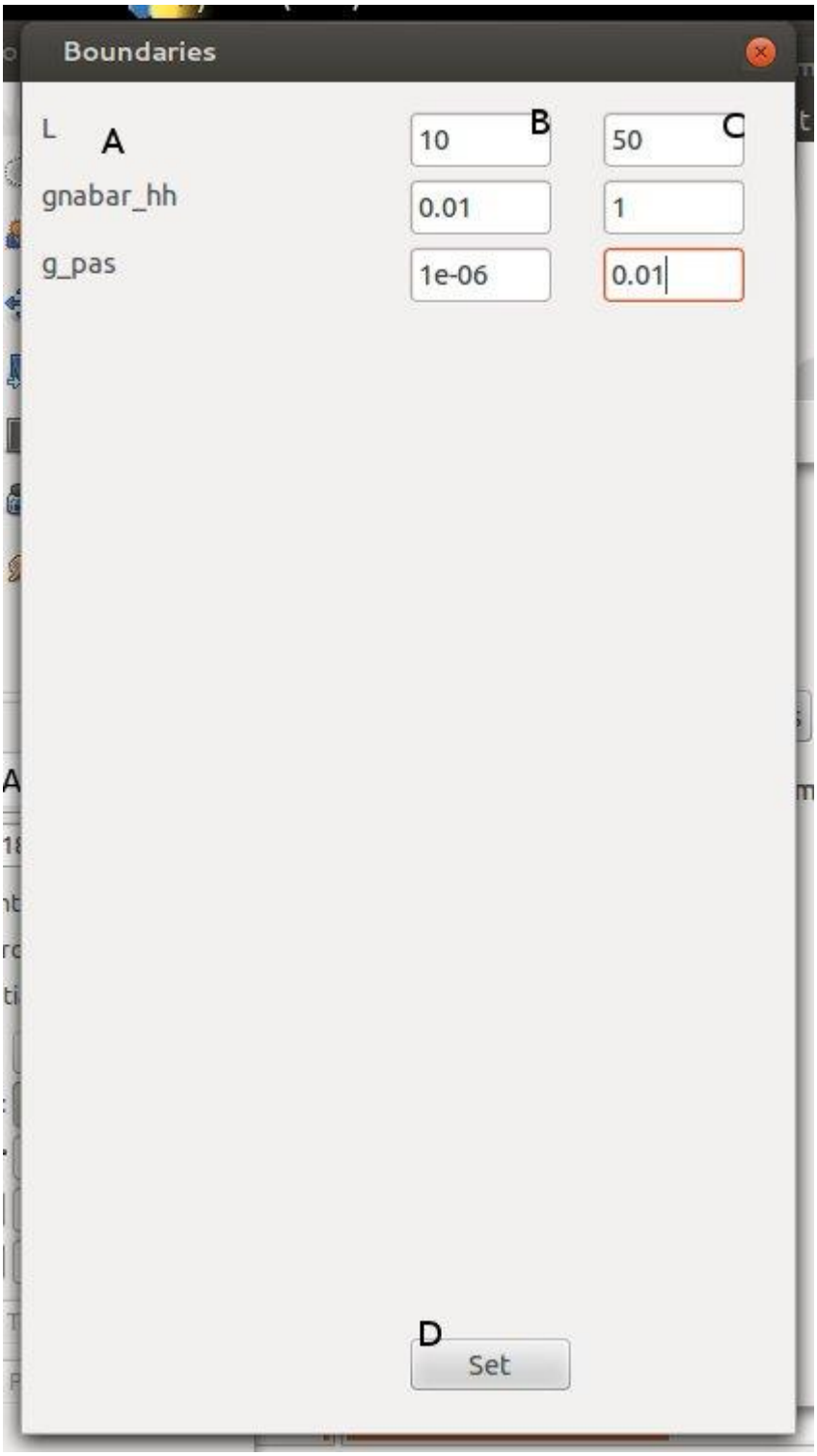
Layer 4



A	Random seed for the random generator.
B	The selected algoritmh. For detailed description see http://inspyred.github.io/ and http://docs.scipy.org/doc/scipy/reference/optimize.html
C	You can specify starting points for the algoritms
D	Bounds for the parameter values
E	Number of individuals used
F	Number of generations used
G	Mutation rate
H	Other algorithm may have different parameters, these are displayed dinamicaly, for their meaning see the previous links

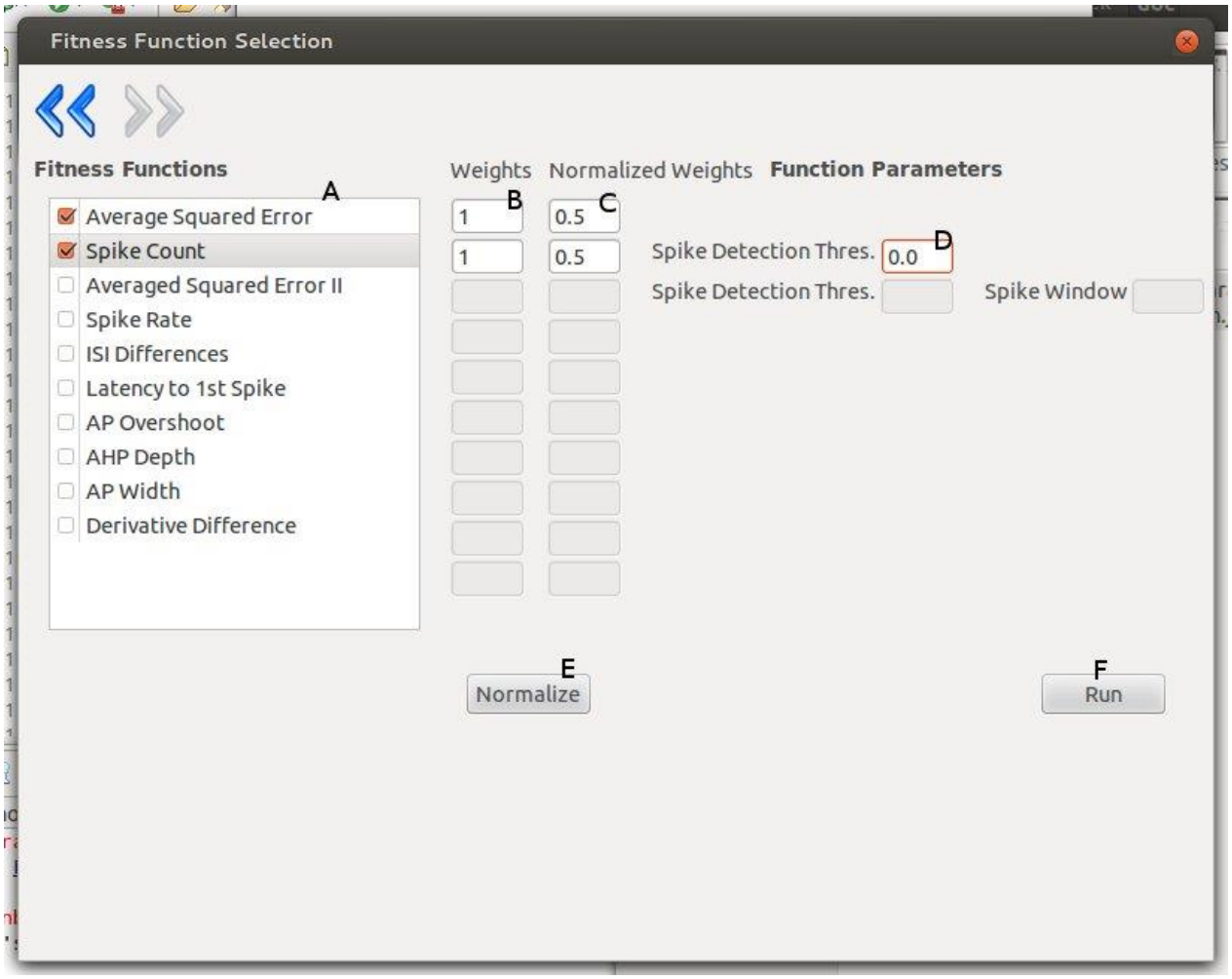
Layer 4.1

This can be accessed by „D” on layer 4.



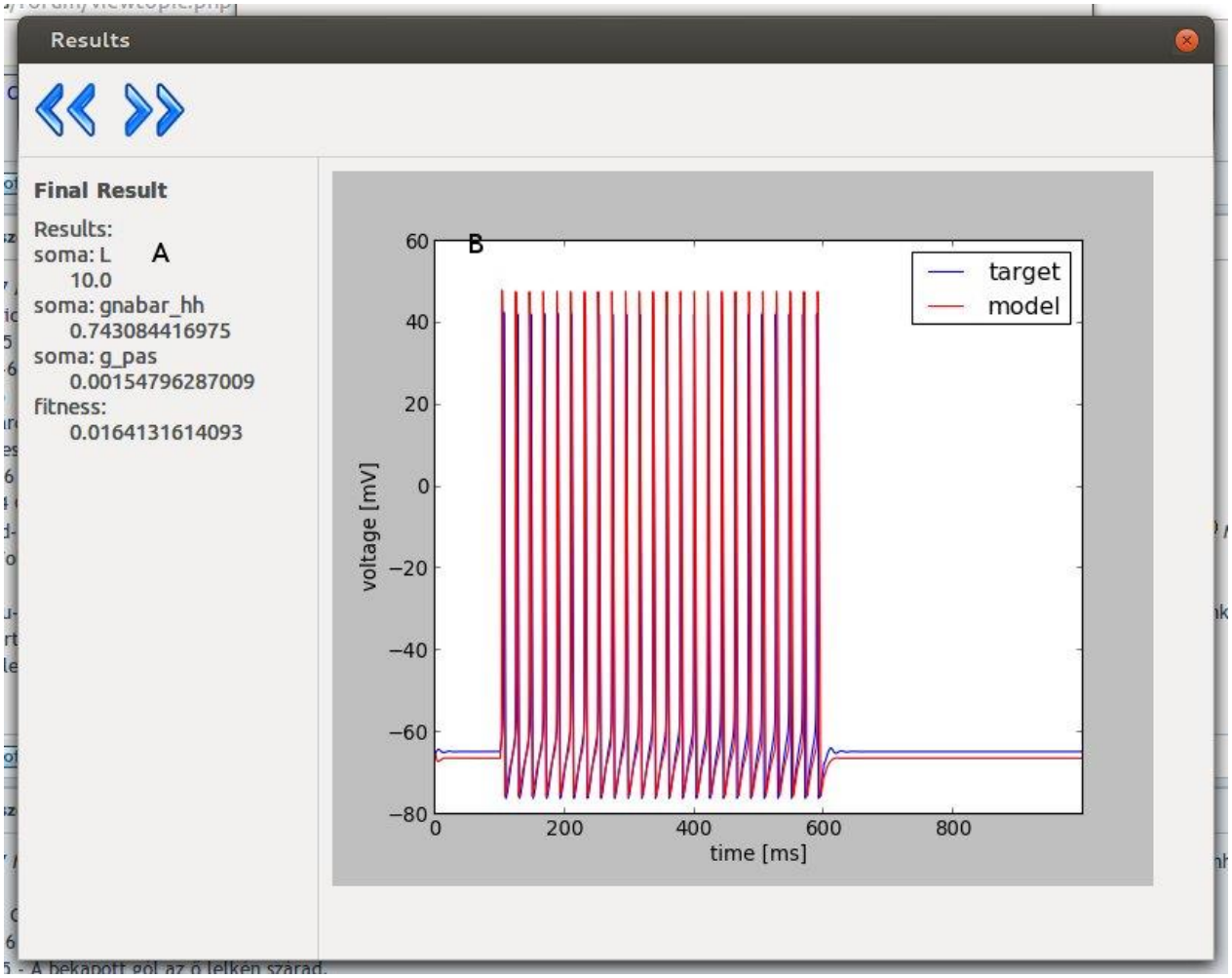
A	Parameters to optimize
B	Minimum values
C	Maximum values
D	Sets the bounds and returns to layer 4

Layer 5



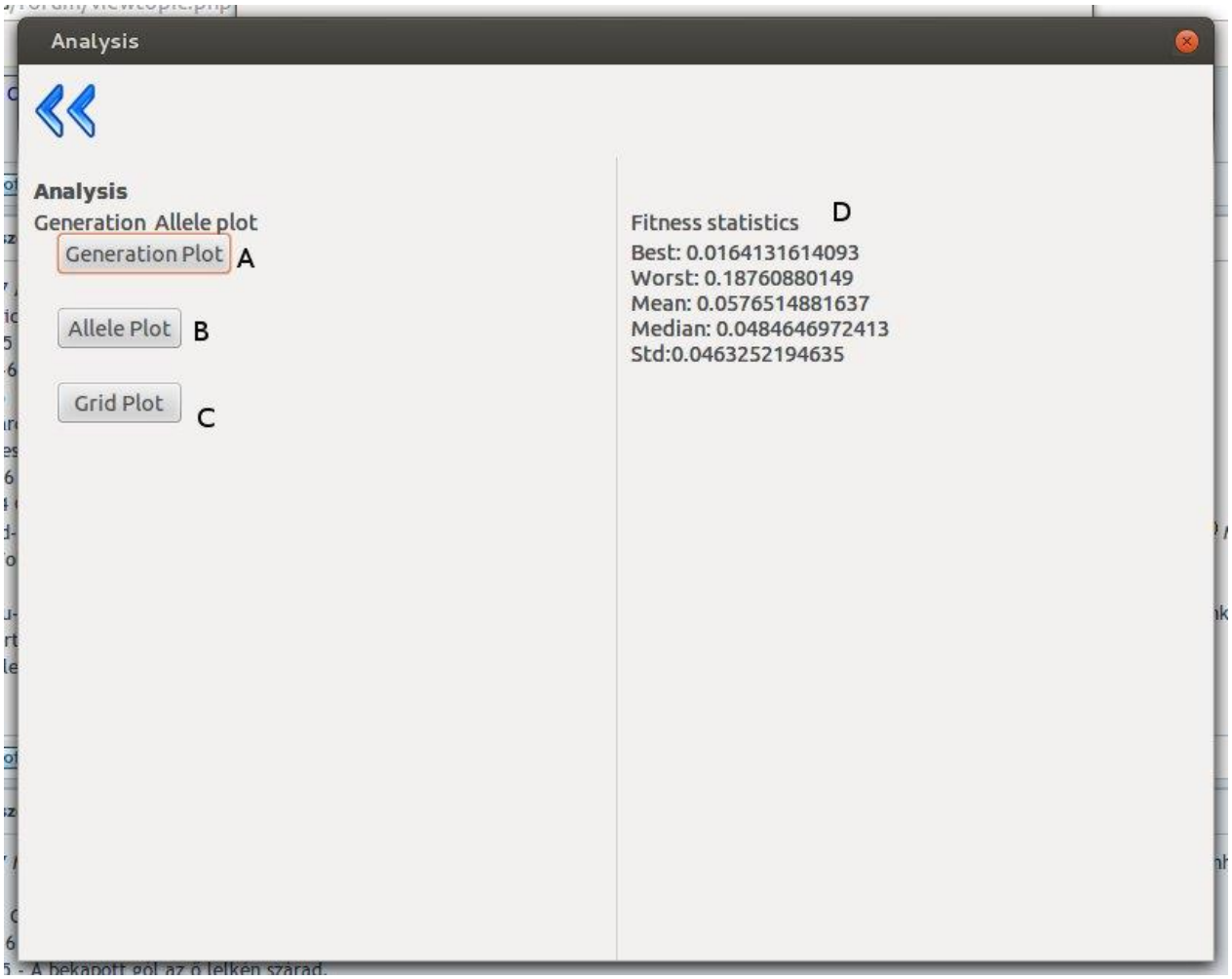
A	List of known error functions. By checking one you select itt o be used during the optimization. By selecting multiple functions you create a combinational error function.
B	Here you can specify the weights of the functions in the combination
C	The program will normalize the given wights and display them here
D	Error function related parameters
E	This will normalize the weights. You must press this befor pressing run, or you must enter values to the normalized column manually
F	Run the optimization. After pressed it will look like it is freezed but itt will display a dialog when finished. You can also see some debug output on the console

Layer 6



A	Here are the optimal parameters and the corresponding fitness value
B	Comparing the resulting trace to the input (in case of multiple input trace, this will concatenate them while displaying)

Layer 7



A	Plots the changes of the population during the generations. Only available for inspyred algorithms. It should thisplay a dialog if you used a different one and you want to use this function.
B	Plots the changes of the allels. Only for EO, not realy usefull. Only available for inspyred algorithms. It should thisplay a dialog if you used a different one and you want to use this function.
C	It plots the projection of each dimension of the search space. Under developement.
D	Some statistics about the results, only for inspyred algorithms.

Fitness functions

Our fitness functions are in the fitnessFunctions module, in the fF class, they have the following syntax:

def f_fun(self,model_trace,input_trace,args):

model_trace: the result of the current simulation

input_trace: the current input trace

args: dictionary for additional arguments

return: simply the corresponding fitness value

To use a new fitness function you must add itt to the self.calc_dict in the fF class.