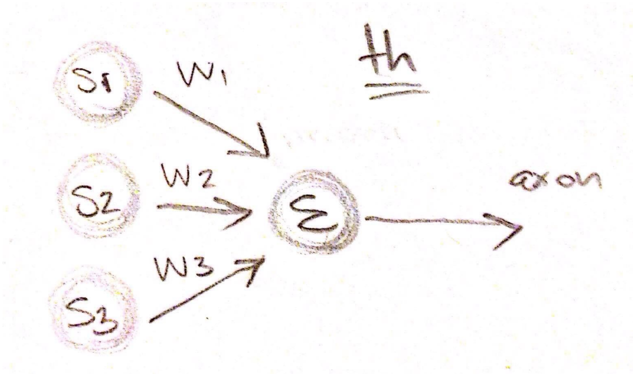
Phanitta Chomsinsap

Lab Report: Function Approximation using NN

1) Neuron (neuron.v)

Design



Neuron is a basic computation unit. The configuration of a neuron includes input synapses, weights, threshold, and output axon. The output of a neuron can be computed by taking a linear combination between the synapses and weights and compare the value to the threshold. If the sum is greater or equal to the threshold, axon will be 1, otherwise 0.

2) Testing the Functionality of Neuron Module (neuron_tb.v)

The test bench feeds in synapses value from 0 to 3 (binary values) into the neuron module. The threshold for this specific test is set to be 2 in decimal value (or 32-bit binary values). As you can see, the output ax remains 0 until the input is 2 in decimal value that it turns 1.

```
VSIM 205> run
# s0000
# ax0
# s0000
# ax0
# s0001
# ax0
run
# s0010
# ax1
# s0011
# ax1
VSIM 206> run
```

3) Neural Network for Square Pulse (nn_sv.sv)

Design

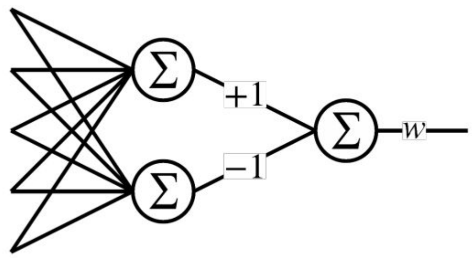
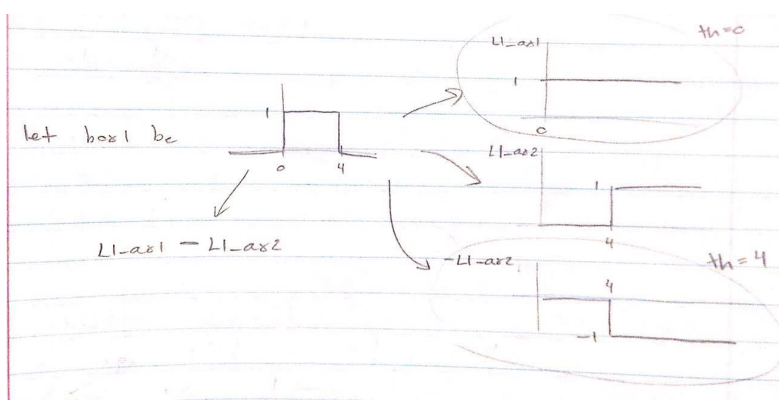
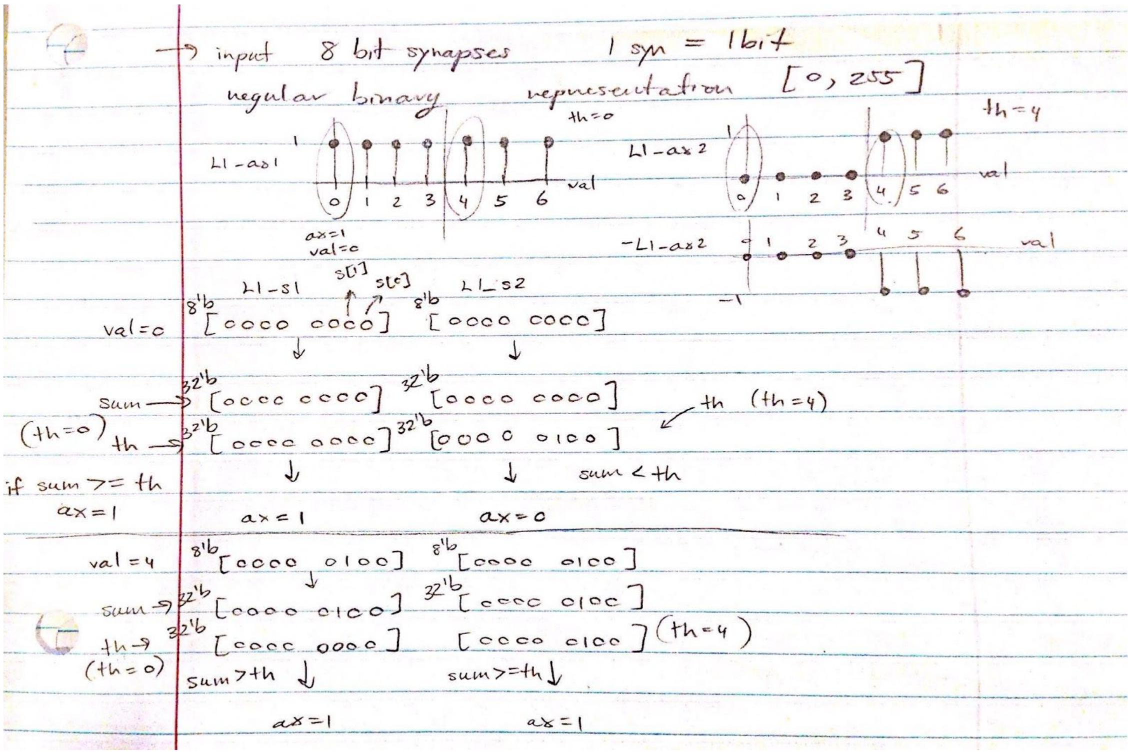


Figure 3: *Neuron Configuration for Square Pulse*

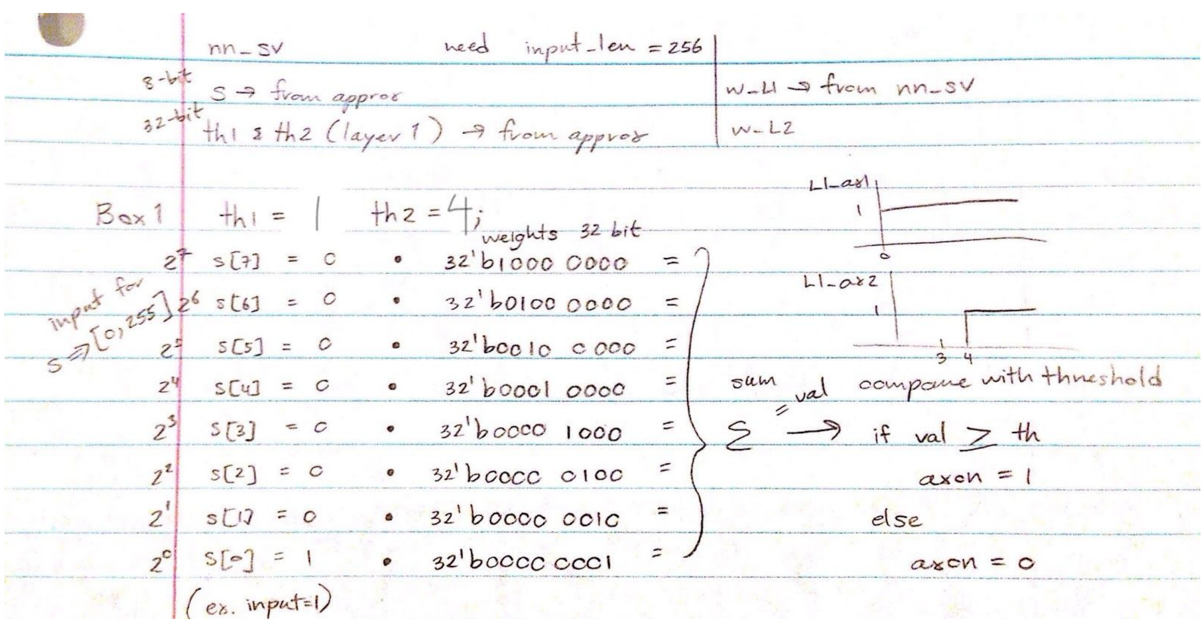
A square pulse can be constructed from the sum of two functions, in which each function is the output of the layer 1 of the network. For example, if the threshold is 0, and input synapses are 0 to 6, the output will always be 1 (**L1_ax1** -- output of first neuron in layer 1). In another case, if the threshold is 4, and input synapses are 0 to 6, the output will be 0 (for input synapses 0-3) and will be 1 (for input synapses 4-6) (**L1_ax2** -- output of second neuron in layer 1). If we let the weights for the 2nd layer of the neural network be +1 and -1, **L1_ax1** function will be the same, while **L1_ax2** will be flipped (**-L1_ax2**). By subtracting the two functions, a square pulse is formed (width = 4).



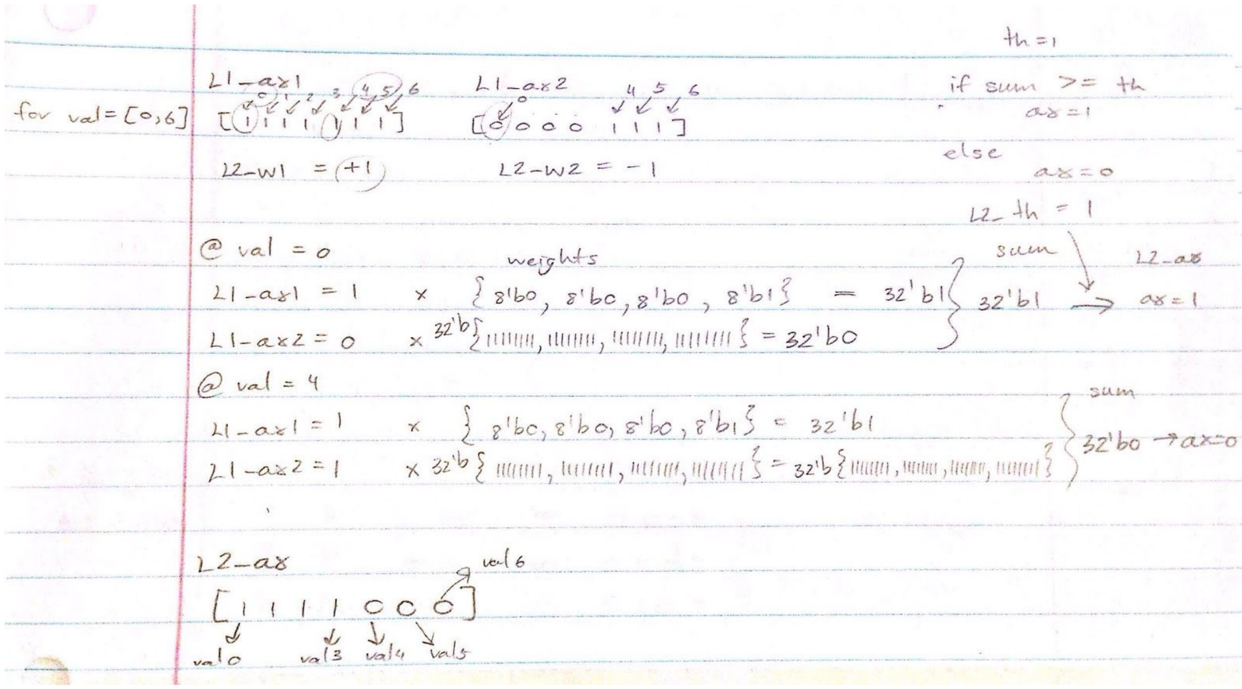


In order to construct a square pulse, two functions can be added. The top image shows how the calculation is done for each bit.

Below shows the weights chosen for layer 1 of the networks. The input synapses into layer 1 are chosen to be 8 bits to represent the input 0 to 255 of the unknown function (Figure 5). Weights are 32 bits in 2's complement representation, since one of the weights to layer 2 requires to be negative.



The calculation for forming a square pulse is explicitly shown below. The threshold of the second layer is chosen to be 1 (32 bits). Let the **L1_ax1** be 1, and **L1_ax2** be 0. If the respective weights are +1 and -1 (in 2's complement), after multiplying the weights the results will be 1 and 0 (32 bits) respectively. After summing up the results, if the sum is 1, then the axon will output 1, otherwise it will output 0.



4) Function Approximation using Neural Network (approx_sv.sv)

Design

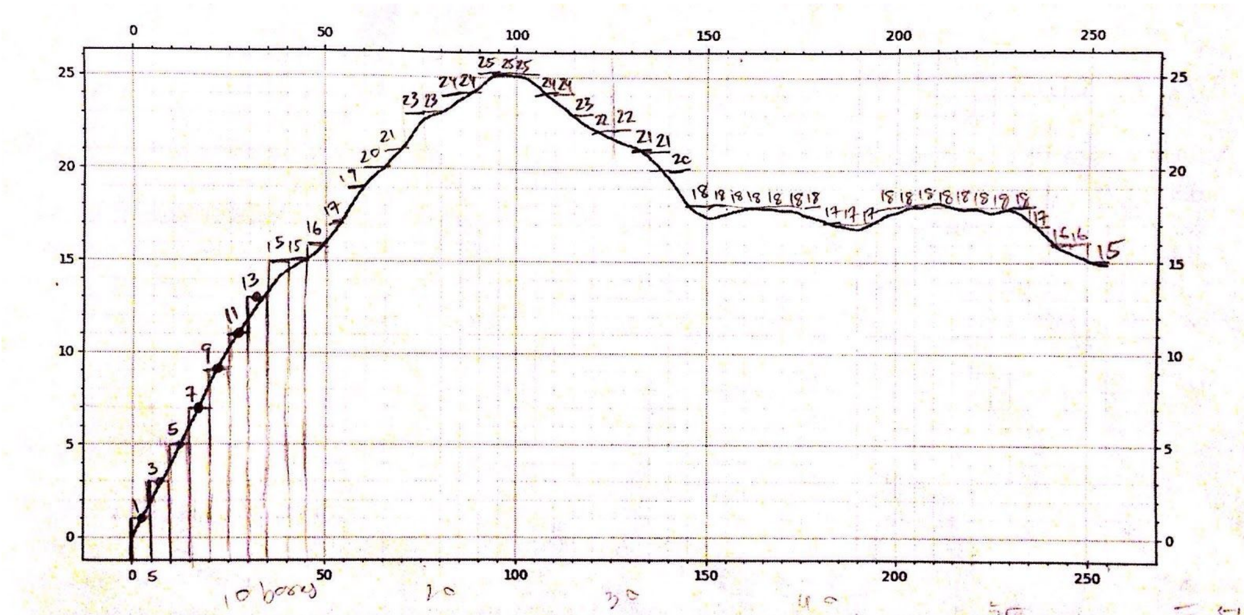
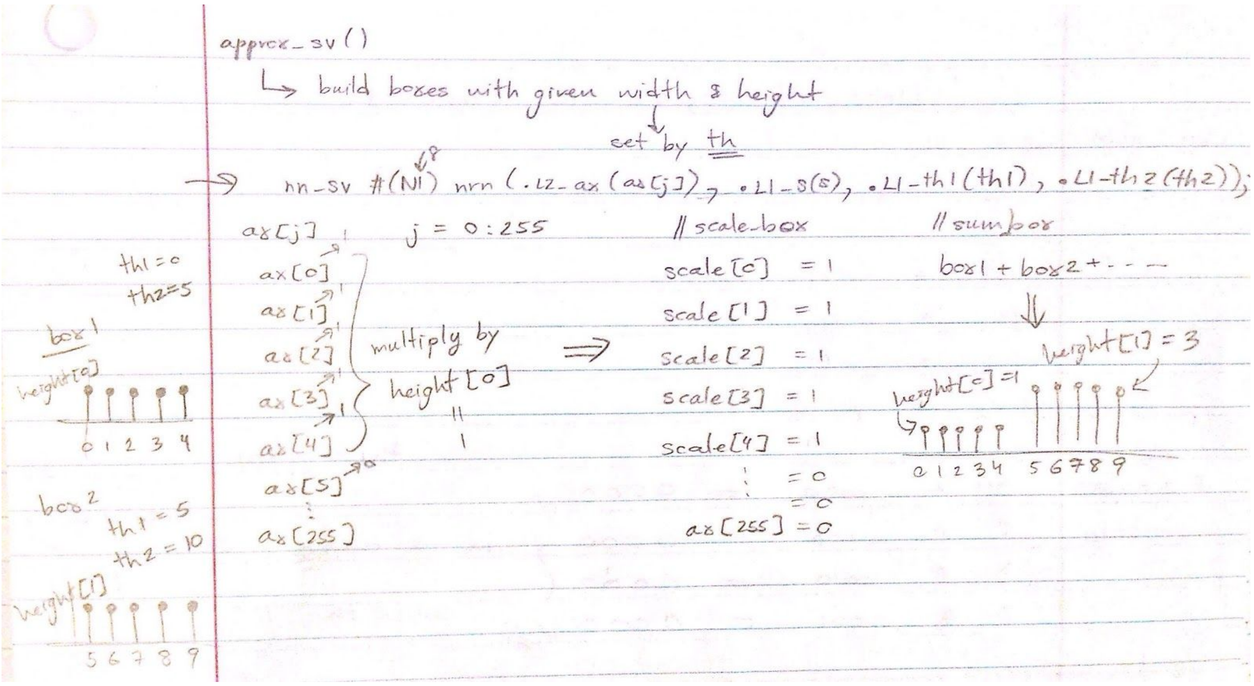


Figure 5: *Some 'Unknown' Function*

The function approx_sv.sv receives one output for every input synapses, and thresholds (for both neuron) provided. Axon outputs of the nn_sv.sv are stored into an array to construct a function of a single square pulse. The nn_sv.sv function is called 51 times, to build a total of 51 square pulses. Each square pulse is then multiplied by a height which I approximated from a graph. Finally, I summed up all square pulses to approximate the Unknown Function in Figure 5 from the lab instruction.



Waveform of a Square Pulse

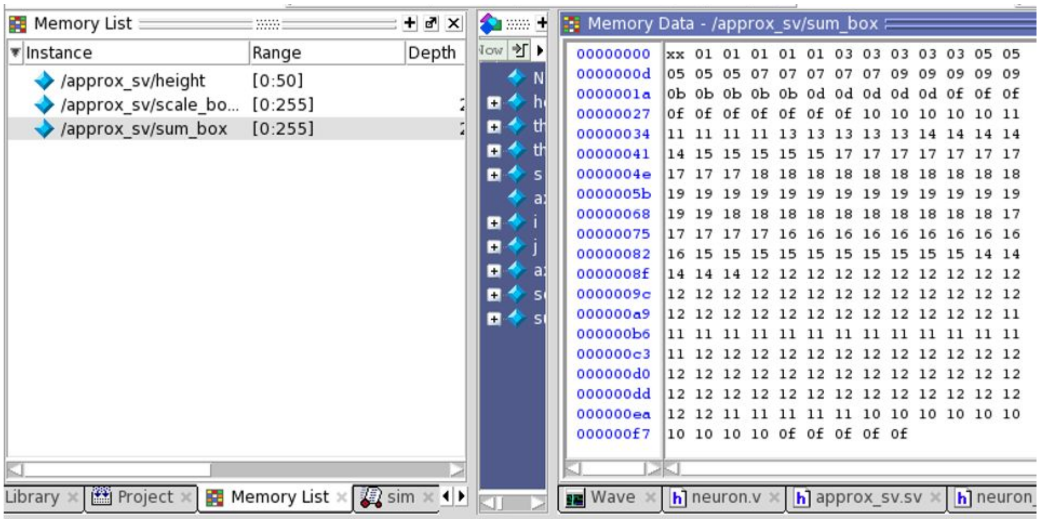
I chose to approximate the function by an interval of 5 for the inputs. The output axon is obtained at every 30 unit delay.

(First box -- th1 = 0 & th2 = 5)

(Second box -- th1 = 6 & th2 = 10)

(Third box -- th1 = 11 & th2 = 15)

Results



The final output of the unknown functions are shown in the memory data of sum_box variable here in hexadecimal. The numbers are then converted to decimal representation in order to plot the final results. The first memory data (xx) can be ignored because it represents the uninitialized value.