**DSP Project Part 1**

**Moustafa Elsayed**

**201600848**

Table of contents:

* Introduction about Discrete Convolution.
* Introducing the Discrete Convolution GUI
* Showing some examples and Discussion about results
* Introduction about spectrum analyzer
* Introducing the spectrum analyzer GUI
* Showing some examples and Discussion about results

Discrete Convolution:

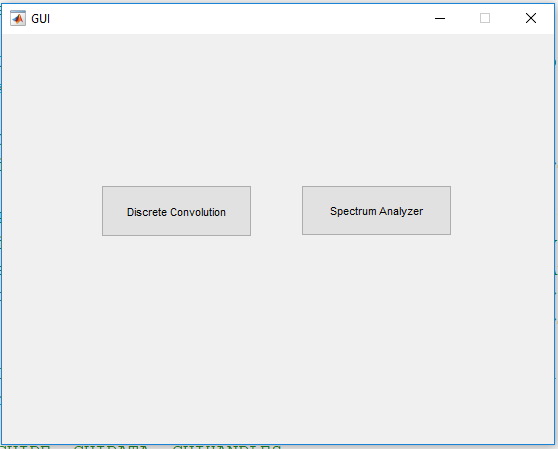
Introduction [1]:

Convolution is one of the most important concepts in signals, can be used to define the output that a LTI (Linear time invariant) system make whenever an input signal is given. It can be shown that a LTI system is completely identified by the impulse response of that system. In digital signal processing convolution can be represented as the sum of the shifted impulse response of the system. This response can also be represented as the shifted and scaled unit impulses. So in short, once the impulse response of an LTI system is known I can know the output of any input signal to this system.

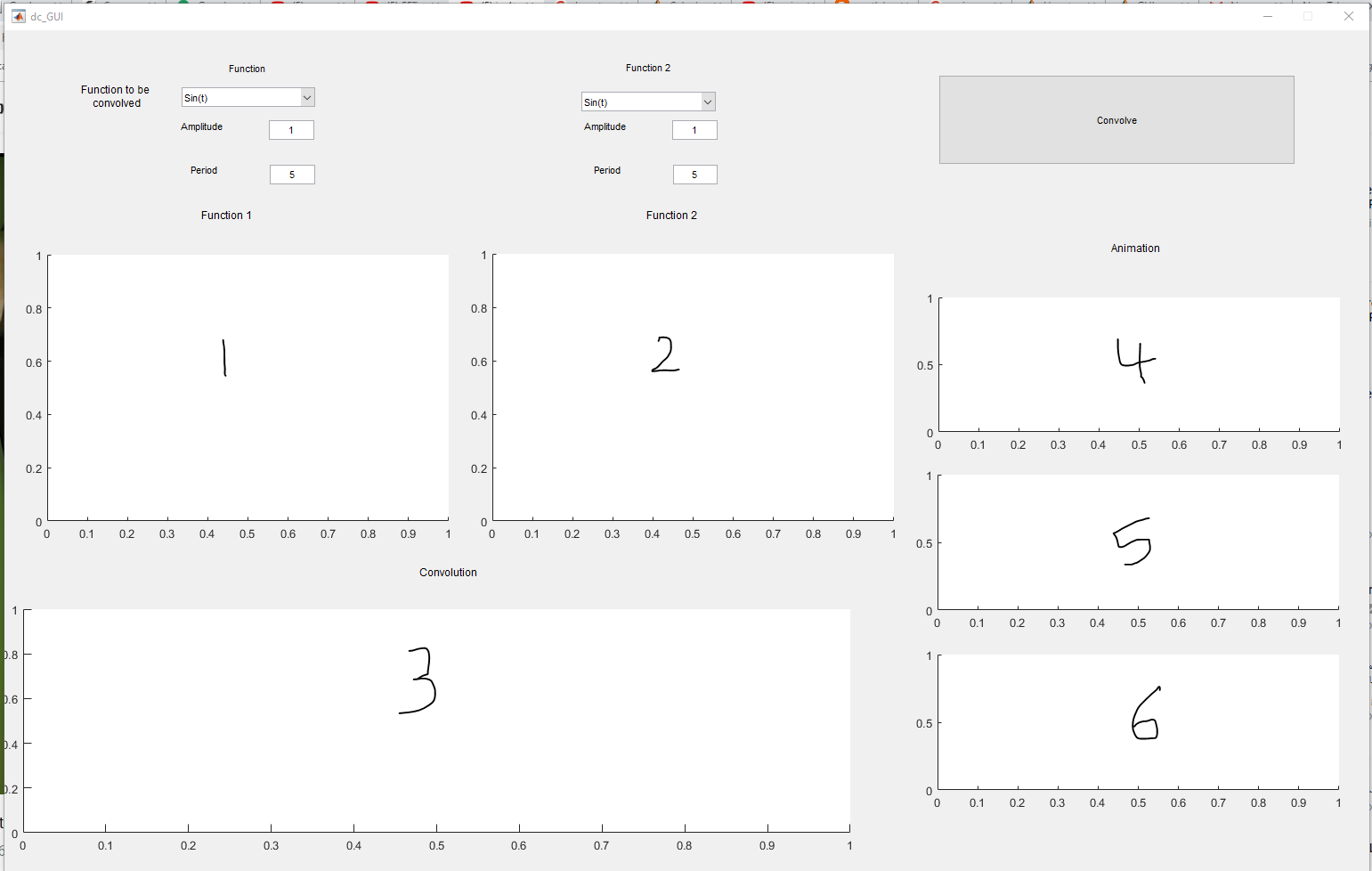
(f ∗ g)(n)= ∑ f (m) g (n-m) |m=-∞: m= ∞

Interface:

First the GUI asks the user whether he wants to enter to the discrete convolution or To the Spectrum Analyzer



If the user chooses to enter to the Discrete Convolution, a new GUI will show up



Here the user can adjust the 2 Parameters in each function (Amplitude and Period of each Function)

Features:

Axes (1): Plotting of the first function.

Axes (2): Plotting of the second function.

Axes (3): Plotting of the Convolution of those two function.

Axes (4), (5), and (6) are responsible of showing the animation of the convolution:

Axes (4): Plotting the moving version of the first function.

Axes (5): Plotting of non-moving function 2

Axes (6): Plotting the result of convolution in an animated way.

**Error Handling:**

* If the user entered a negative input to the parameters the code will automatically take the absolute of this parameter.
* If the user entered a 0 value for the period I made a default value of 1 second

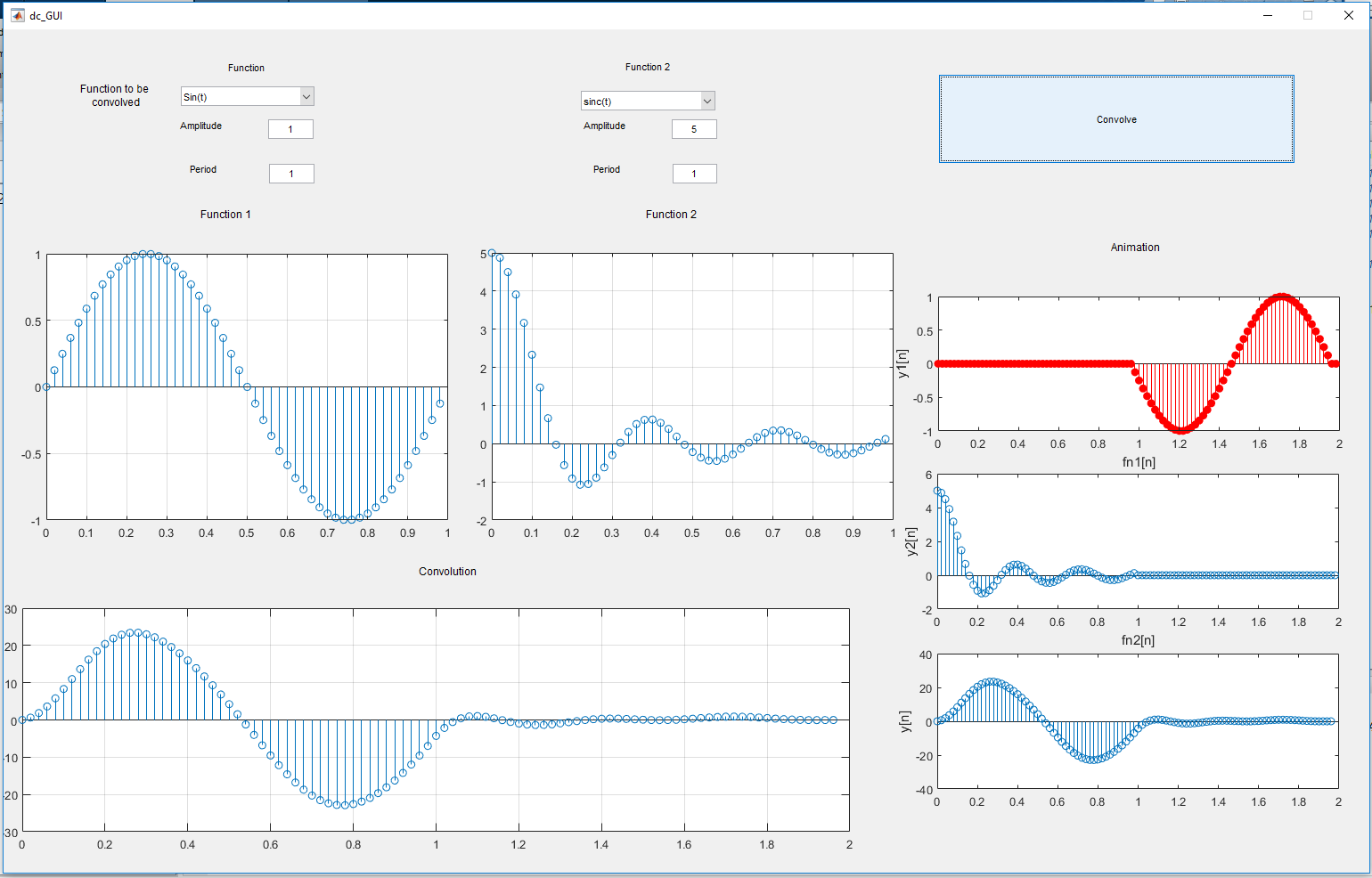
**Example 1:**

In this example I made two functions:

Function 1: sin (2\*pi\*t) Amplitude=1 and period = 1 second

Function 2: sinc (2\*pi\*t) Amplitude =5 and Period = 1 second

The Convolution axes there is the result of the convolution and in the right part there is an animation of the convolution



**Results:**

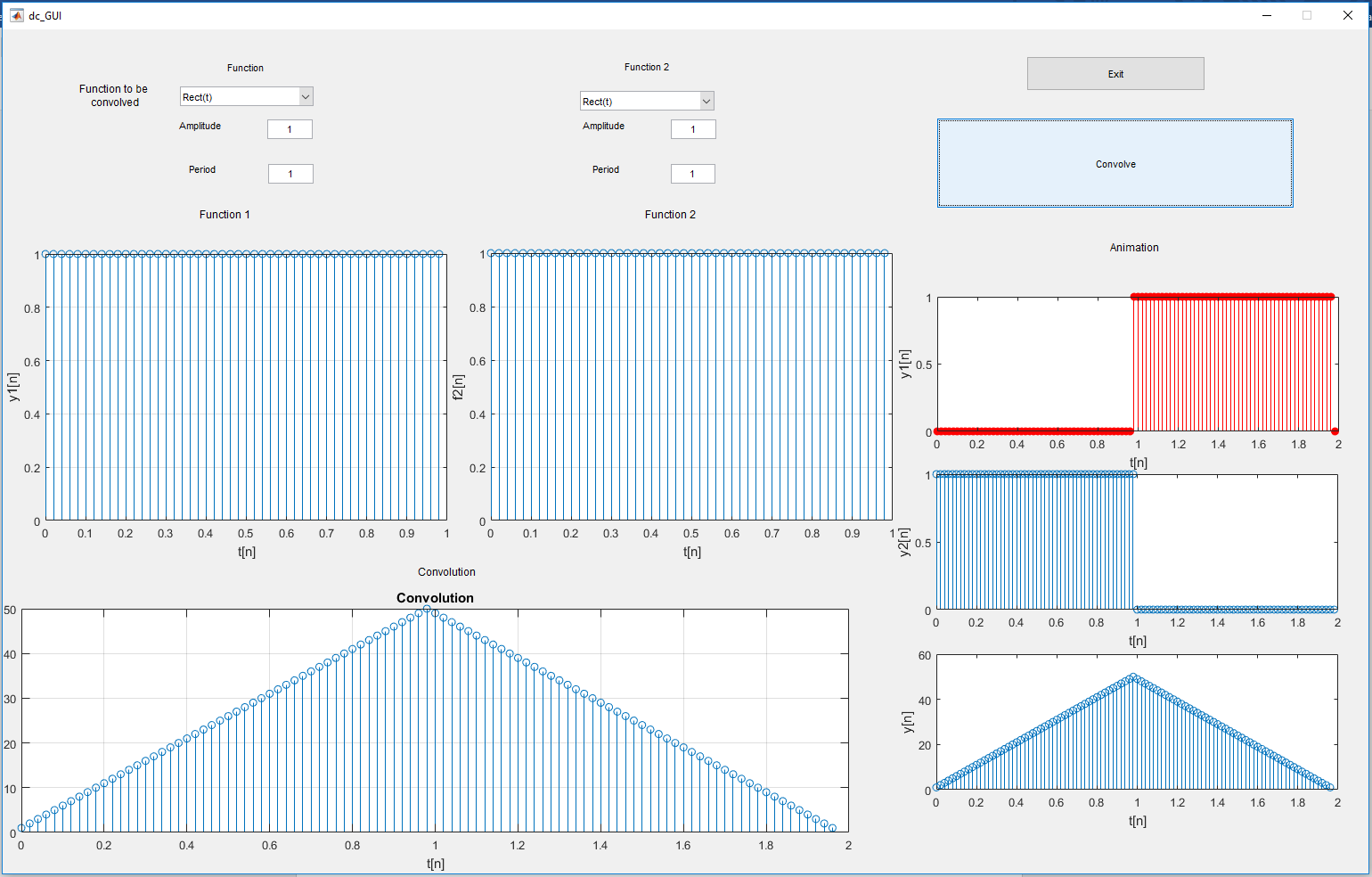
Here we can see that the sinc is a dying function and the convolution also acts this way. This is a logical because after shifting the sin for a while it is almost multiplying it with 0.

**Example 2:**

Here I convolved 2 rectangular pulses of the same width.

By inspection we can say that the output will be of a triangular shape as shifting function 1 from

-infinity: infinity will not make any change to the output unless it intersects with function 2. And the maximum intersection will happen when both on top of one another entirely and this happens when shifting happens with value of the whole function 2 time period, and it is here at time = 1 second.



Introduction about FFT-based (Fast Fourier transform) Spectrum analyzers [2]:

Spectrum analyzers are essential whenever we are dealing with signals because their ability to decompose any input signal to sines and cosines and when combining both of them we can represent it as an exponential functions. We decompose them to sines and cosines because those 2 functions are both orthogonal on each other and on themselves (unless it is the same function with same frequency) when. So we can represent a signal as a linear combination of those orthogonal functions with different coefficients. We use FFT instead of normal DFT because DFT has a complexity of O ( ). But thee FFT algorithm has a complexity of O (). FFT can be used on a large signal to minimize computation need. Also windowing must be taken into consideration in case we want to reduce leakage on a certain signal. For example hamming window has a low side-lopes but it as a trade-off it has a low resolution at the main lope as it is so wide.

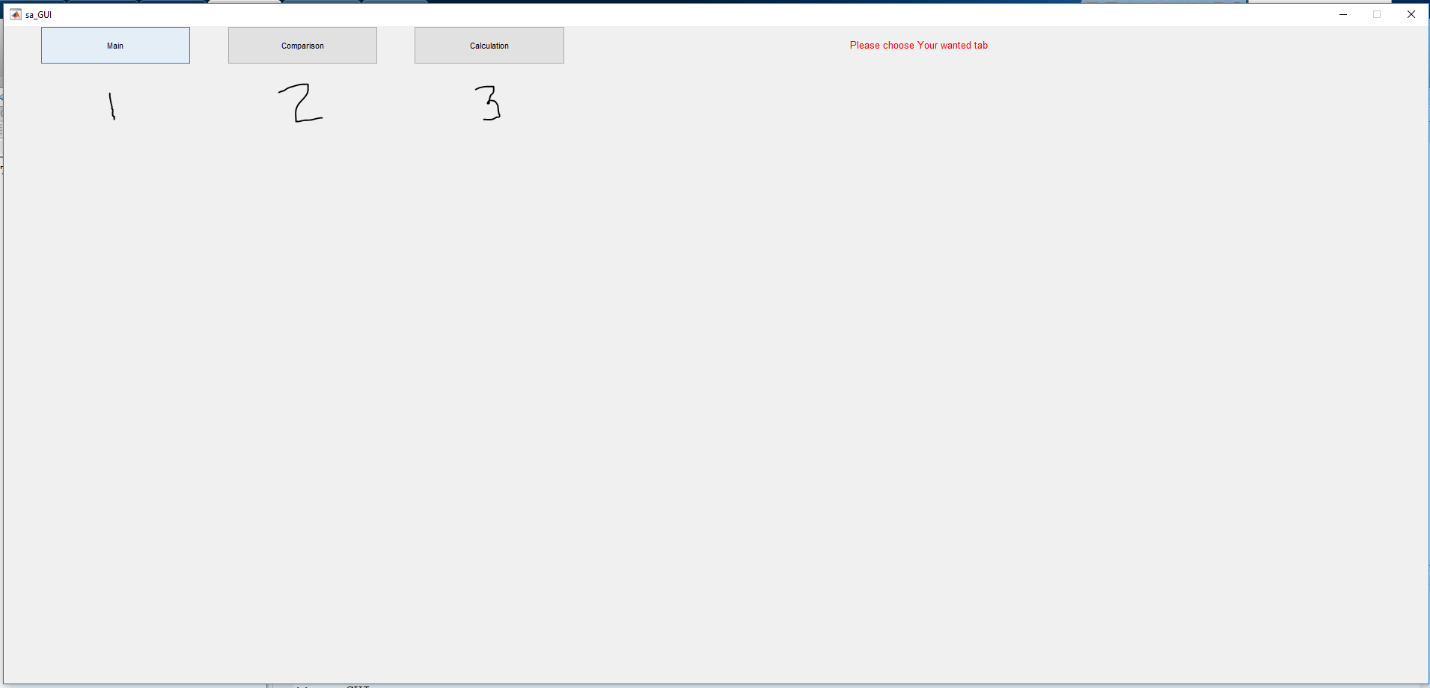
I used the MATLAB built in function fft (funciton) to get the output of the fft algorithm.

Introduction Spectrum analyzer GUI:

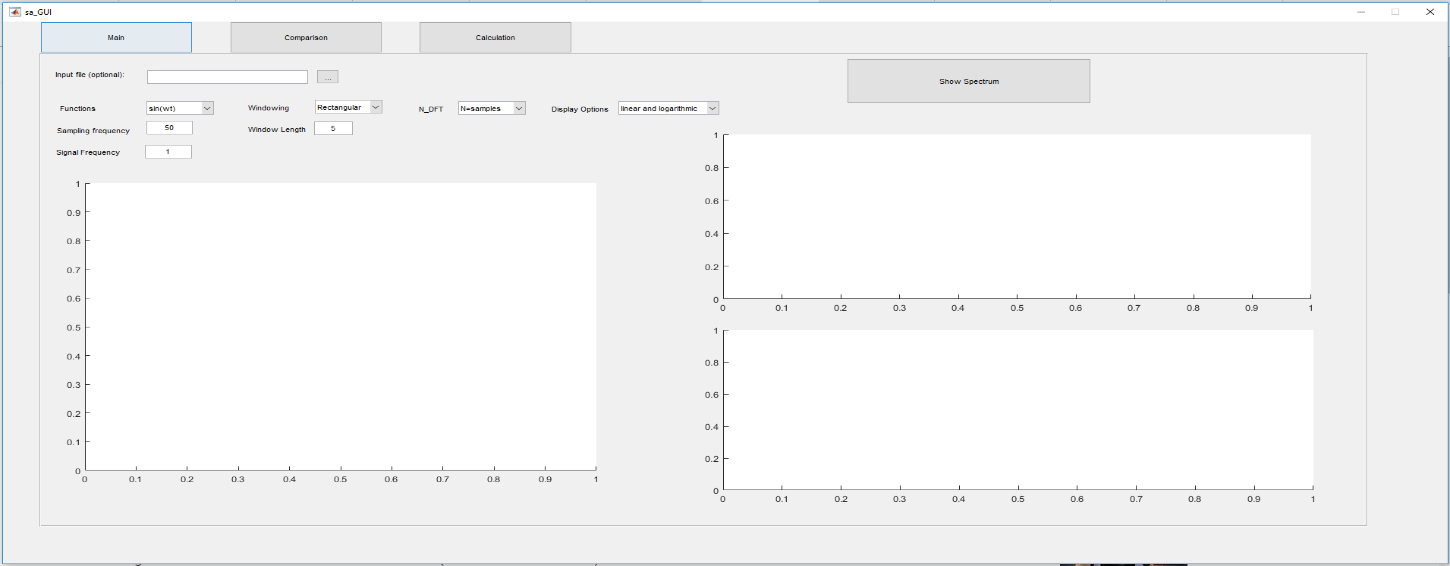
If the user uses the Spectrum analyzer this GUI will open

There are 3 Tabs available:

* Main
* Comparison
* Calculation



If the user clicked the main tab:

This will show up

Features:

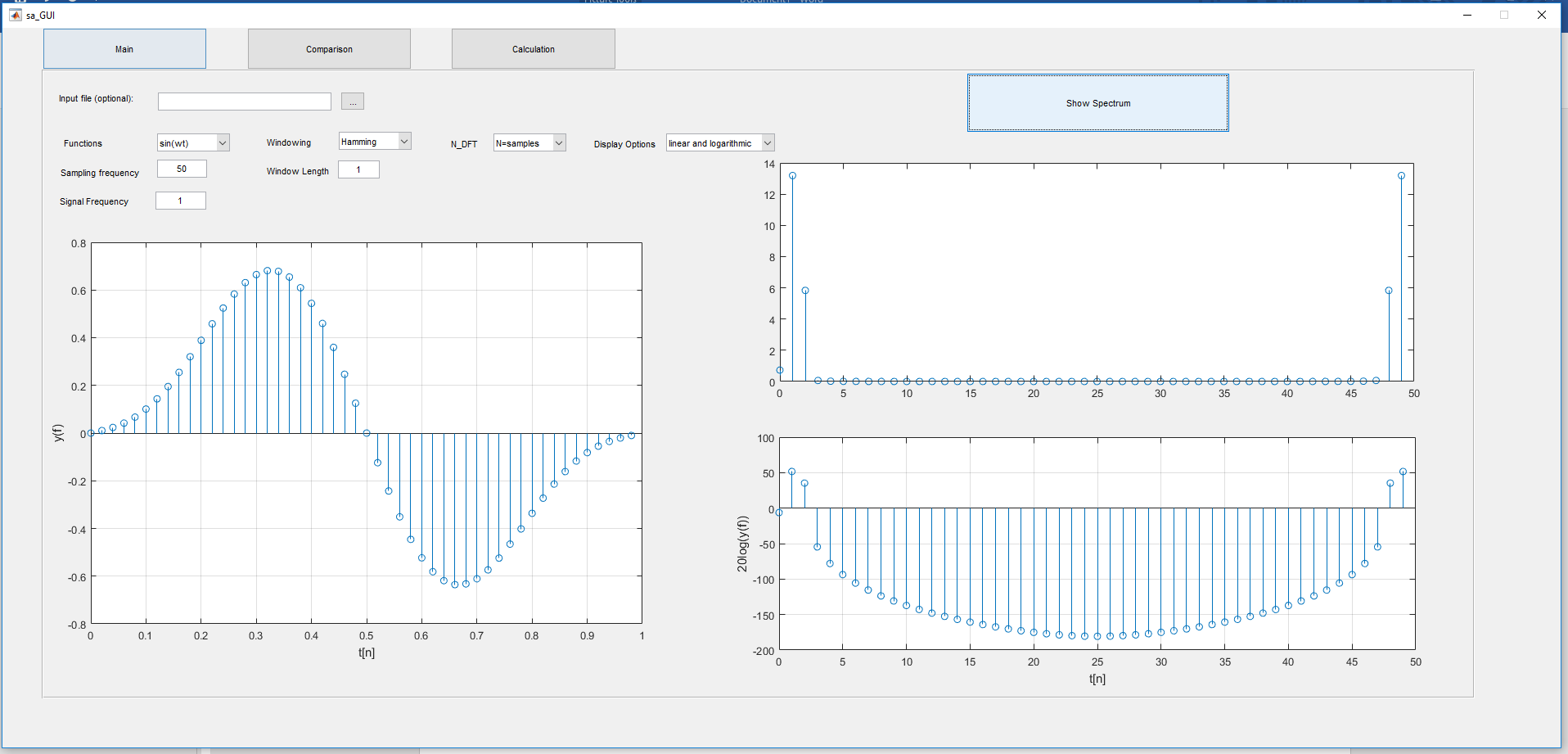
In this Tab the user has the freedom to adjust any of the parameters:

1. There are 4 options of functions (Sin(wt) – sinc(wt) - rectangular - Input function (this function is the can be any signal that the user can choose from his desktop)
2. There are 4 different window that can be applied to the function to see their different effects in the frequency domain. Those are (Rectangular – Triangular – Hanning - Hamming) windows and each one of them has its own different effect in the frequency with its pros and cons. And the user can specify the time period of this window
3. Then there 5 different choice of the number of points of DFT to be applied to the input signal (N= # of samples of the signal - 128 - 1024 - 32k - 64k)
4. Also the users can make different View of the GUI (Linear & Logarithmic – Real & imaginary – magnitude and phase)

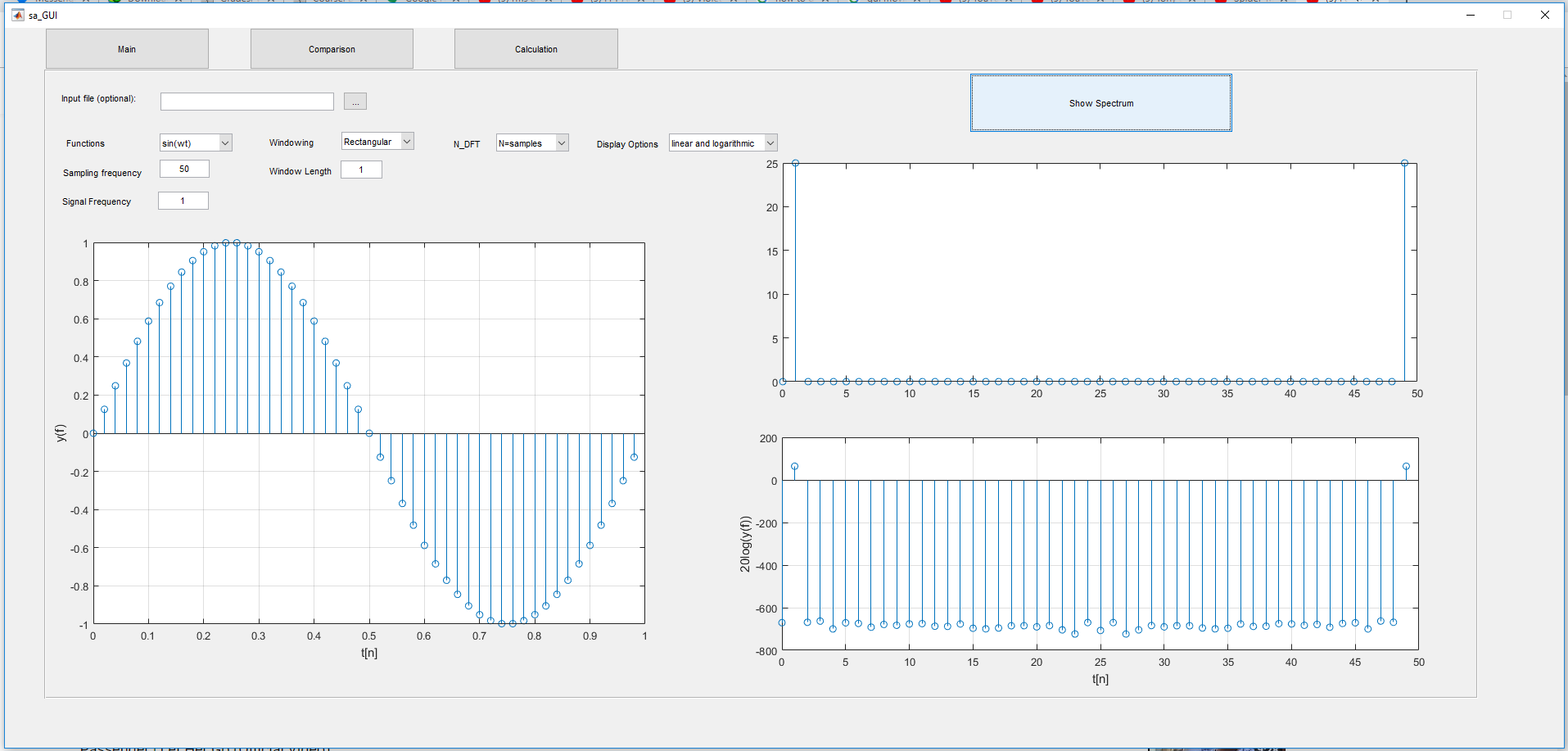
Here is an example:

I applied a hamming window of length 1 seconds and N-point DFT equal to the number of the samples of the sin function and a sampling frequency 50 Hz and chose the display to be linear and logarithmic

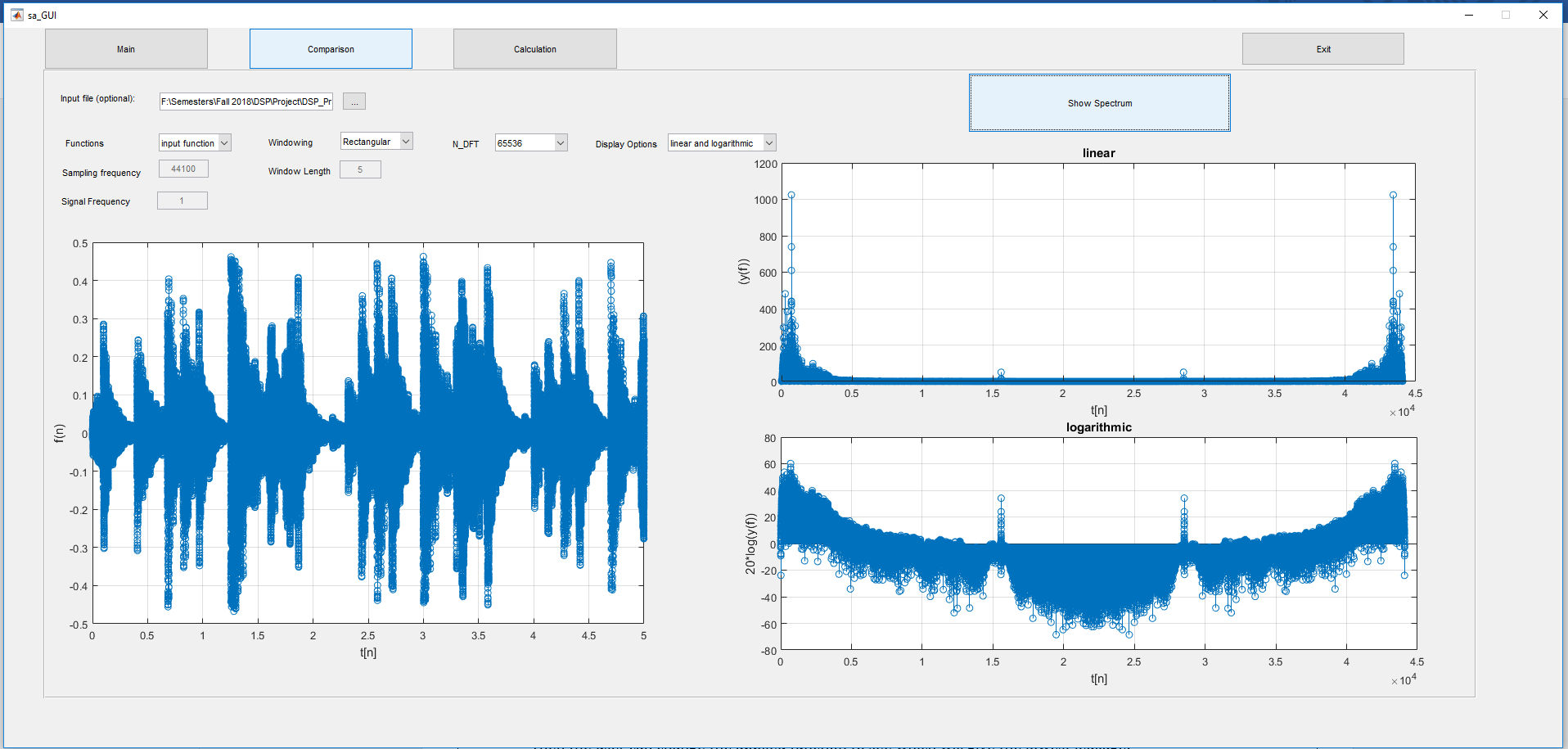
We can see that the hamming window causes a leakage because the main-lope is a lot wider and the zeros of the sinc that is resulting from this window in the time domain does not multiply in the leakage points unlike the rectangular window



But the rectangular window did not cause any leakage this is because the frequency the

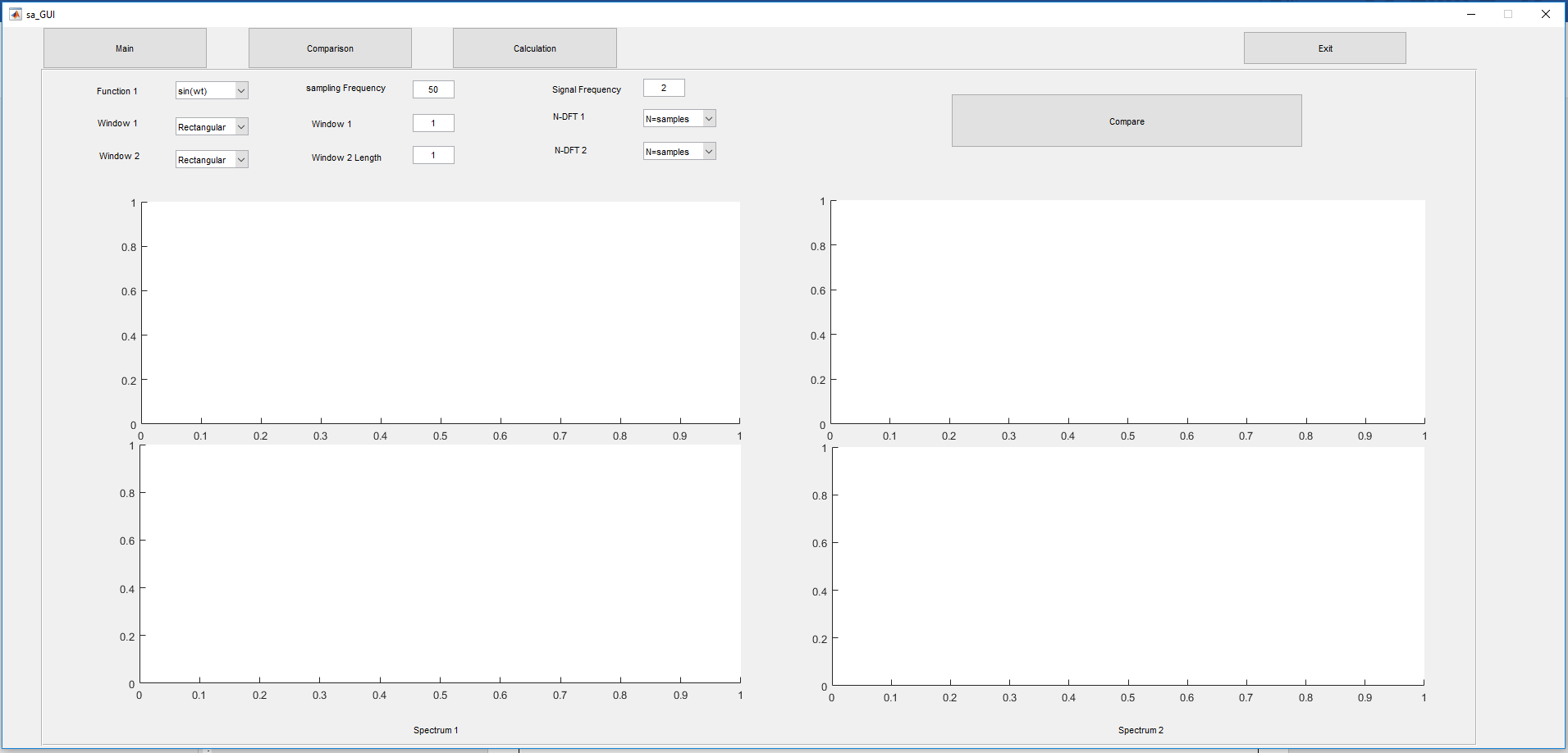


Also the user can input a signal of his own and choose “Input function from functions list”, but he cannot change any sampling or signal frequency. When the user chooses the input function the text field of the window time and sampling frequency to the sampling frequency of the audio file and 5 seconds for the window time.



Then the user can change the applied window to see which will give the lowest leakage.

The second Tab is Comparison Tab:



Features:

* Defining function to operate on (sin – sinc – Rectangular)
* The user can define the sampling and signal frequency.
* Changing the time of each window.
* Changing the N-point DFT of each window.
* Also he can Exit to the opening window

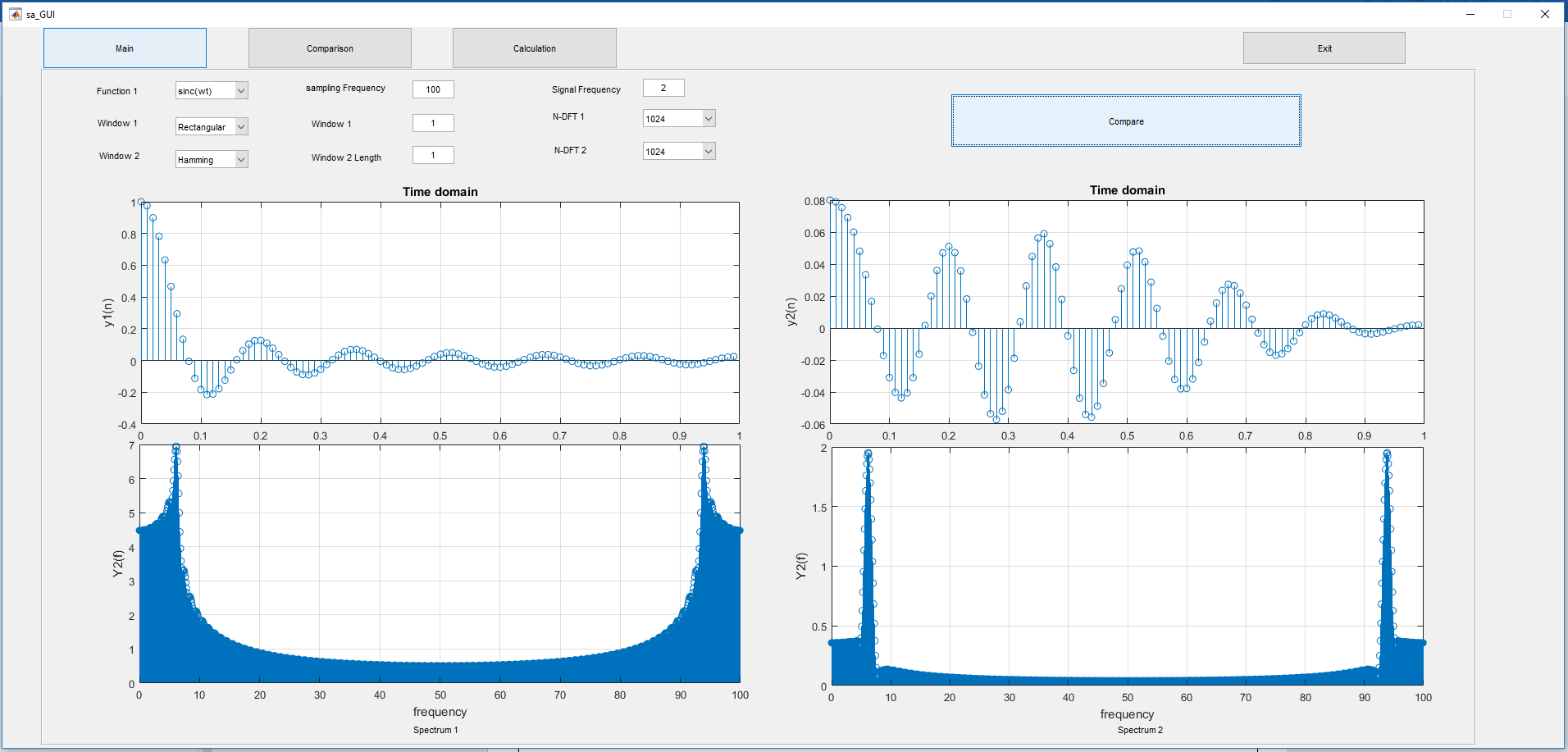
Error Handling:

* If the user entered a negative value the program will automatically take the absolute of the input value.

The purpose of this tab is to compare the effect of windowing the same function with two different window to see the characteristics of each window.

Example:

Here I used a sinc function with a sampling frequency 100 with a window length 1 second and applied 2 different windows rectangular and hamming windows.

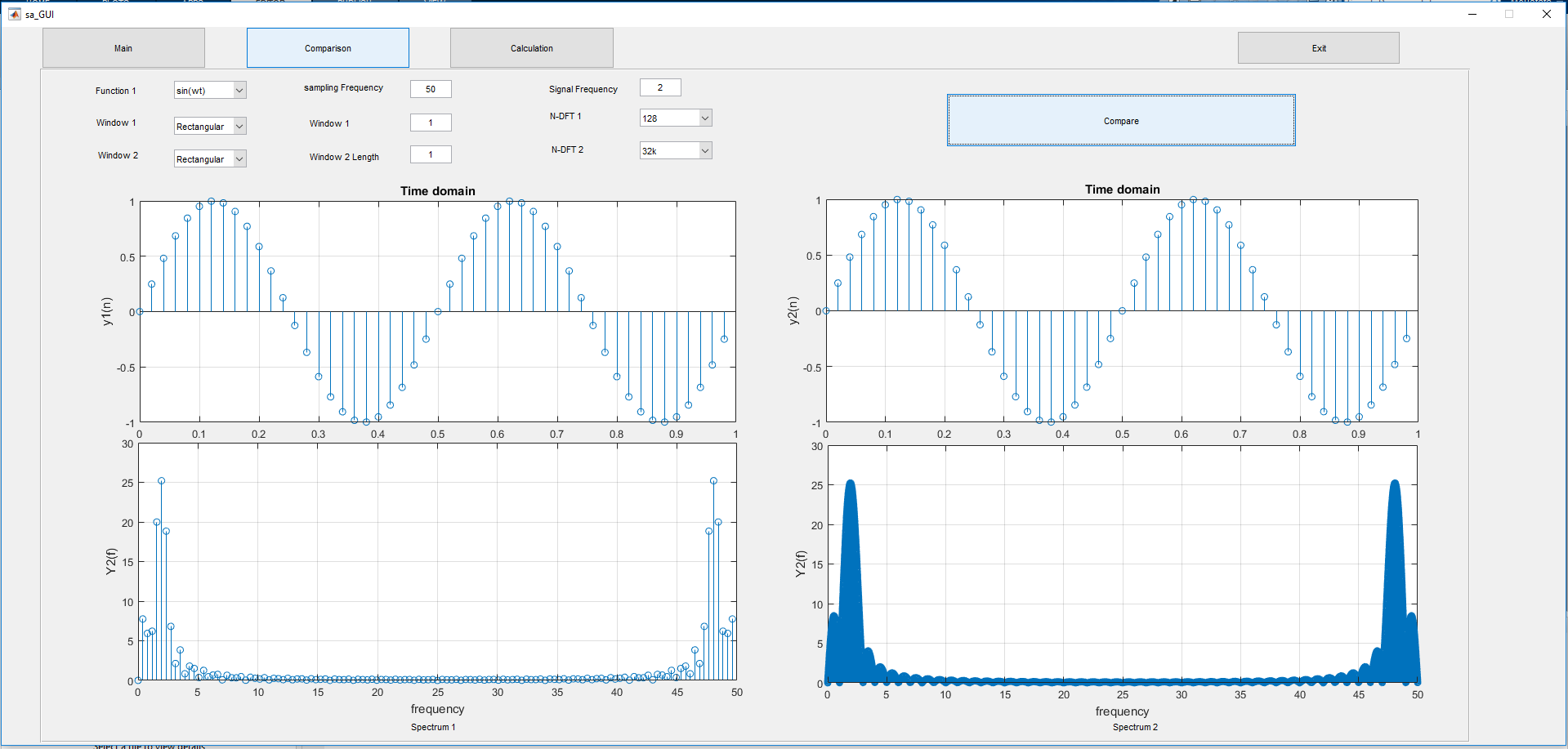


Results:

* We can see that the Rectangular window has bigger side-lopes leading to a big leakage problem unlike the hamming window that has an advantage here as it has very low side-lopes then minimizing the leakage.
* Also we can see that the hamming window effect has a magnitude of 0.08 so in the frequency domain it has a maximum of 2. On the other hand, the rectangular window has a higher amplitude. And this makes sense as the rectangular has no scaling effect in time domain but the hamming has very low magnitude at the first but then increases in the mid-point then decrease again.

Example 2:

Here I only changed the second N-DFT points and it is obvious that they are the same thing but with different resolution.



**References:**

[1] http://pilot.cnxproject.org/content/collection/col10064/latest/module/m10087/latest