**Automation Laboratory**

Parameter Estimation: Frequency

**Master’s in IT**

Submitted by: **Suraj Sanath Kumar Bharadwaj**

Frankfurt University of Applied Sciences

**Contents**

[**Abstract** 2](#_Toc408103)

[**1.** **Introduction** 3](#_Toc408104)

[**2.** **Solution Strategy** 4](#_Toc408105)

[**3. Description of Implementation**: 7](#_Toc408106)

[**3.1** **Algorithm Implementation** 7](#_Toc408107)

[**3.2 GUI Implementation** 10](#_Toc408108)

[**4. Test Strategy** 12](#_Toc408109)

[**4.1 Generating Analog Noisy signal** 12](#_Toc408110)

[**4.2 Sampling Analog Signal and Storing in Excel file** 13](#_Toc408111)

[**4.3 Using Algorithm to Determine Fundamental Frequency** 13](#_Toc408112)

[**5. Test Scenarios** 14](#_Toc408113)

[**6. Conclusion** 19](#_Toc408114)

[**7. References** 20](#_Toc408115)

# **Abstract**

The real world applications mostly constitutes of periodic signals, which can be modeled as sum of signals whose frequencies are integral multiples of fundamental frequency. Hence estimating fundamental frequency becomes critical.

Among various estimation methods parameter estimation methods stands out for its accuracy. In this project a parametric estimation algorithm to determine fundamental frequency has been implemented.

The method is based on Non Linear Least Square Estimator and cost function. For the presentation and user interface a GUI has been created in python. The performance is evaluated by test strategy of generating an analog signal and sampling it at a specific sampling frequency, storing the values in an external file and applying the algorithm for the values read from this file.

# **Introduction**

The real world applications mostly constitutes of periodic signals, which can be modeled as sum of signals whose frequencies are integral multiples of fundamental frequency. Hence estimating fundamental frequency becomes critical.

Due to this, numerous methods have been proposed to estimate this fundamental frequency. These methods range from correlation methods to parametric estimation methods. Although parametric estimation methods have high computational complexity and are time consuming, they are much more accurate.

Few known parametric estimation methods are:

* Maximum Likelihood
* Non Linear Least Square
* Harmonic Summation

The main difficulty in estimating the fundamental frequency is that a non-linear optimization problem has to be solved. No closed-form solution is available, and we, therefore, have to search for the global optimizer. This search for the optimizer is often performed using the following steps.

1. The cost function is evaluated on a grid and one or several candidate optimizers are selected on this grid. Often, the grid is uniform since a part of the cost function can then be evaluated efficiently using an FFT algorithm.

2. The candidate optimizers are refined using, e.g., interpolation methods, line searches, or derivative-based methods.

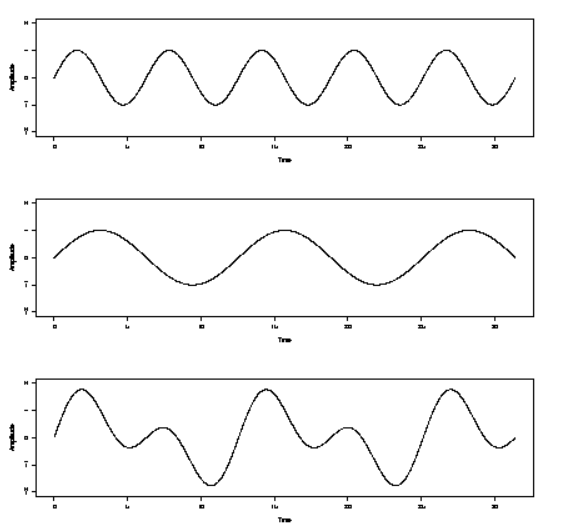
3. For the parametric methods, model order estimation has to be performed when the model is unknown to reduce the risk of estimating an integer multiple or division of the true fundamental frequency. Often, this problem is also referred to as pitch halving/doubling or as octave errors. Estimating an unknown model often means that we have to repeat the first two steps above for every candidate model, thus increasing the computational complexity significantly.

The model order has been fixed for this implementation.

# **Solution Strategy**

**Non Linear Least Square Method**

Any periodic signal z(n) can be written as a sum of harmonic components. Each harmonic component is a simple sinusoid with an amplitude, frequency and phase. The frequencies of these signals are integral multiple of fundamental frequency.



**Figure[[1]](#footnote-1)**: Addition of sinusoids with different frequencies.

* The signal model for any periodic signal is:

**z(n) = l(n) = lcos(ω­o l n + φ1)** …..(2.1)

where,

**Al**  is real amplitude of the lth harmonic

**φ1**is phase of lth harmonic

**ω­o**is fundamental frequency in radians/sample

**L** is number of harmonics order.

* The method of Least Squares is the basis of NLS method. The important step involved is to find the error which is the difference of the data and the signal model generated with specific model parameters.
* The principle is to minimize the squared error or find the parameters which minimizes squared error.

…(2.2)

**where,**

**e(n)** is error

**x(n)** is periodic signal

**z(n)** is mathematical model for the signal

**Steps of NLS in detail:**

* The mathematical model for signal is formulated using the equation:

**z(f) =**

**z(f)=[cos(2ft) + jsin(2ft)]** ….(2.3)

* The real and imaginary part of the signal model is separated.
* The least square error is:

**Te = [x - Zl (ω­o) αl]T [x - Zl (ω­o) αl]** ….(2.4)

Where estimate of **αl given ω­o,**

**αl̂̂̂̂̂̂ = [ZLT(ω­o) ZL(ω­o)]-1 ZLT(ω­o ) x** ….(2.5)

* The NLS cost function is

**JNLS(ω­o,l)=xTZl(ω­o)[Z­lT(ω­o)]-1 Z­lT(ω­o)x** ….(2.6)

* The fundamental frequency that maximizes this cost function (2.6) minimizes the least square error (2.4). Hence the argmax of the cost function is determined. The frequency corresponding to this argument gives the fundamental frequency. The equation is:

**ω­***̂o,***,L = argmax {xT ZL(ω­o) [ZLT(ω­o) ZL(ω­o)]-1 ZLT(ω­o) x}** ….(2.7)

**Selecting the Grid Size:**

* The minimum and maximum index for the grid is set. The grid is divided from minimum to maximum index in steps of appropriate grid size.
* This grid size is important so that we neither make grid so fine that we make too many unnecessary cost function evaluations which leads to more computational complexity, nor so coarse that we under sample and miss the required value.
* The indices obtained are further divided by the length of data to form the grid.
* The frequencies in this grid is looped to evaluate the cost function and estimate fundamental frequency.

**FLOWCHART:**

Min and Max indices are set for the grid

Indices are created from Min to Max in steps of grid size

Grid is obtained by dividing indices by total length of data

i < length of grid?

N

Y

Mathematical/sinusoidal model is created for corresponding frequency value for index of loop(i) in the grid

Cost Function is calculated

Argmax of cost function is found

Get frequency at the index

# **3.** **Description of Implementation**:

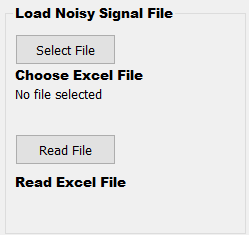
The estimation of fundamental frequency using NLS method is implemented in Python 3.7, and GUI has been created using QT designer 5.11.2 , PYQt5 on windows.

## **3.1 Algorithm Implementation**

**Phase 1: Loading and reading the data.**

* The file which contains the sampled values is selected and the file is read when Read File button is clicked and the values from file are stored into a variable. The select file and read file status will be updated.
* Reading the file content is a tricky part here. **read\_excel** function from pandas library is used which will return data as dataframe. This dataframe is converted to a matrix and a transpose is taken and finally stored.
* The above mentioned functionality has been implemented in the **def loadFile()** and

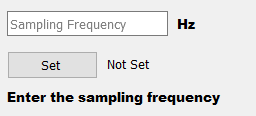
**def readFile()**



**Figure:** A part of GUI for selecting the file and reading its contents.

**Phase 2: Setting the sampling frequency.**

* The given sampling frequency is entered in the line field and it is set. Once the sampling frequency is set, the status of set sampling frequency is updated. The status gets updated from “Not Set” to “Sampling Frequency Set: \_\_\_\_\_\_\_ Hz”. Based on this sampling frequency we are setting the frequency Bounds for our frequency estimation algorithm.
* The lower bound is set to 1 and upper bound is set to (sampling frequency)/3 because making pitch grid for range from 1 to sampling frequency is time consuming.
* The upper bound is fixed to sampling frequency by 3 because we know that sampling frequency has to match the nyquist criteria(Fs > 2).
* The above mentioned functionality has been implemented in the **def setSamplingFrequency().**



**Figure:** A part of GUI for Setting the Sampling frequency

**Phase 3: Plotting the given signal**

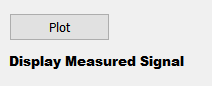
* Based on Sampling frequency steptime is calculated using formula:

**steptime =1/ Sampling Frequency**.

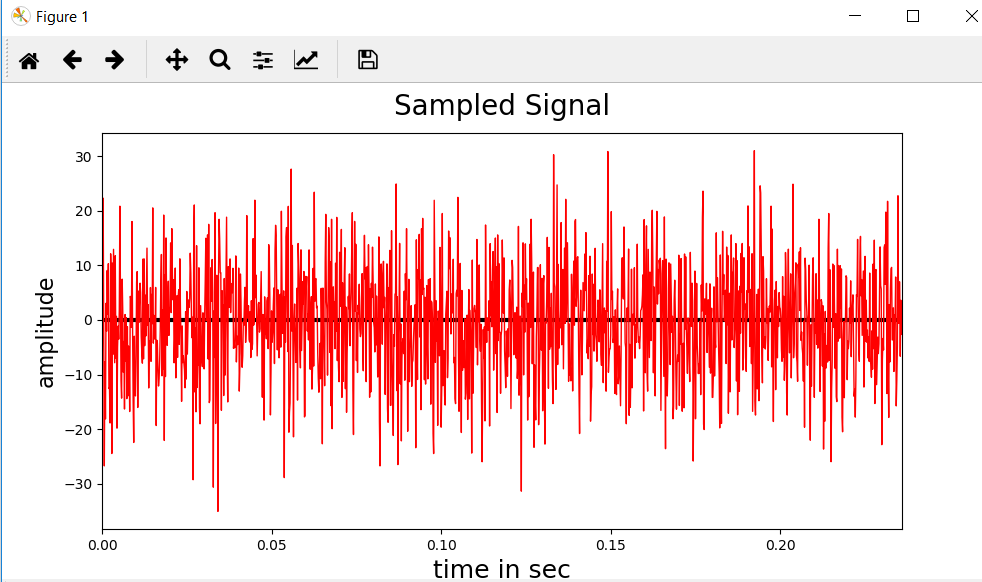
* An array for time index of size Nx1 is created in steps of steptime, where N is number of samples given. Calculation of duration of signal is done using formula:

**Time duration= No of samples\*steptime.**

* X axis limit is set to the calculated Time duration.
* The input signal vs steptime is plotted. This is achieved with the Plot Button. A new window for figure opens showing the plot.
* The above mentioned functionality has been implemented in the **def plotInputSignal()**



**Figure:** A part of GUI for Plotting the data



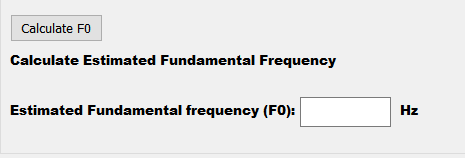
**Figure:** A part of GUI for displaying the measured signal.

**Phase 4: Estimating Fundamental Frequency (Main Algorithm)**

* First step is storing the length of signal in a variable.
* Then minimum index and maximum index is calculated by multiplying the length of signal with lower and upper frequency bounds respectively calculated in **def** **setSamplingFrequency()**.
* An array containing upper bound number of values starting from minimum index to maximum index is created.
* The obtained indices are further divided by length of signal and store in an array to form a full pitch grid.
* An empty array of length same as length full pitch grid with NaN values to store the cost functions is created.
* A pitch grid array similar to full pitch grid is created with an extra value to include the possibility of considering the upper bound as well.
* Next we loop through all the frequencies in the pitch grid to create a sinusoidal model using the formula: **sinusoidal Matrix = ,** Separate the real and imaginary components of sinusoidal model, cost function is calculated based on formula:

costFunctionMatrix(0,i) = real((data'\*sinusoidalMatrix)\*(sinusoidalMatrix\data));

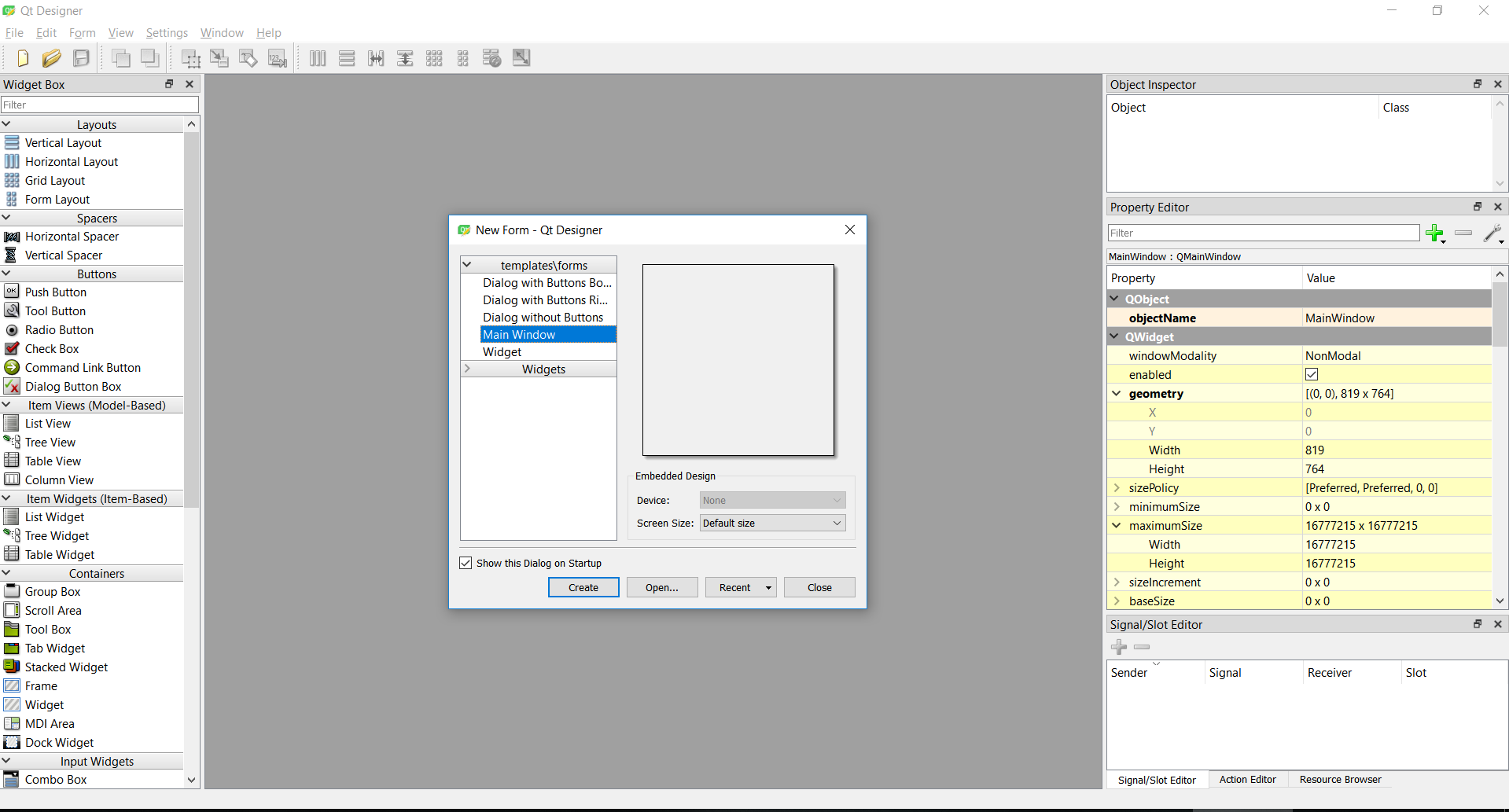
* The cost function is computed on a uniform (Fourier) grid.
* The index which has maximum value in cost function is found and stored.
* The fundamental frequency is estimated from the full pitch grid at this index.
* The calculate F0 button performs the estimation and result is shown in the empty box in GUI.
* The above is implemented in the **def estinmateFundamentalFrequency().**



**Figure:** A part of GUI for Calculating F0

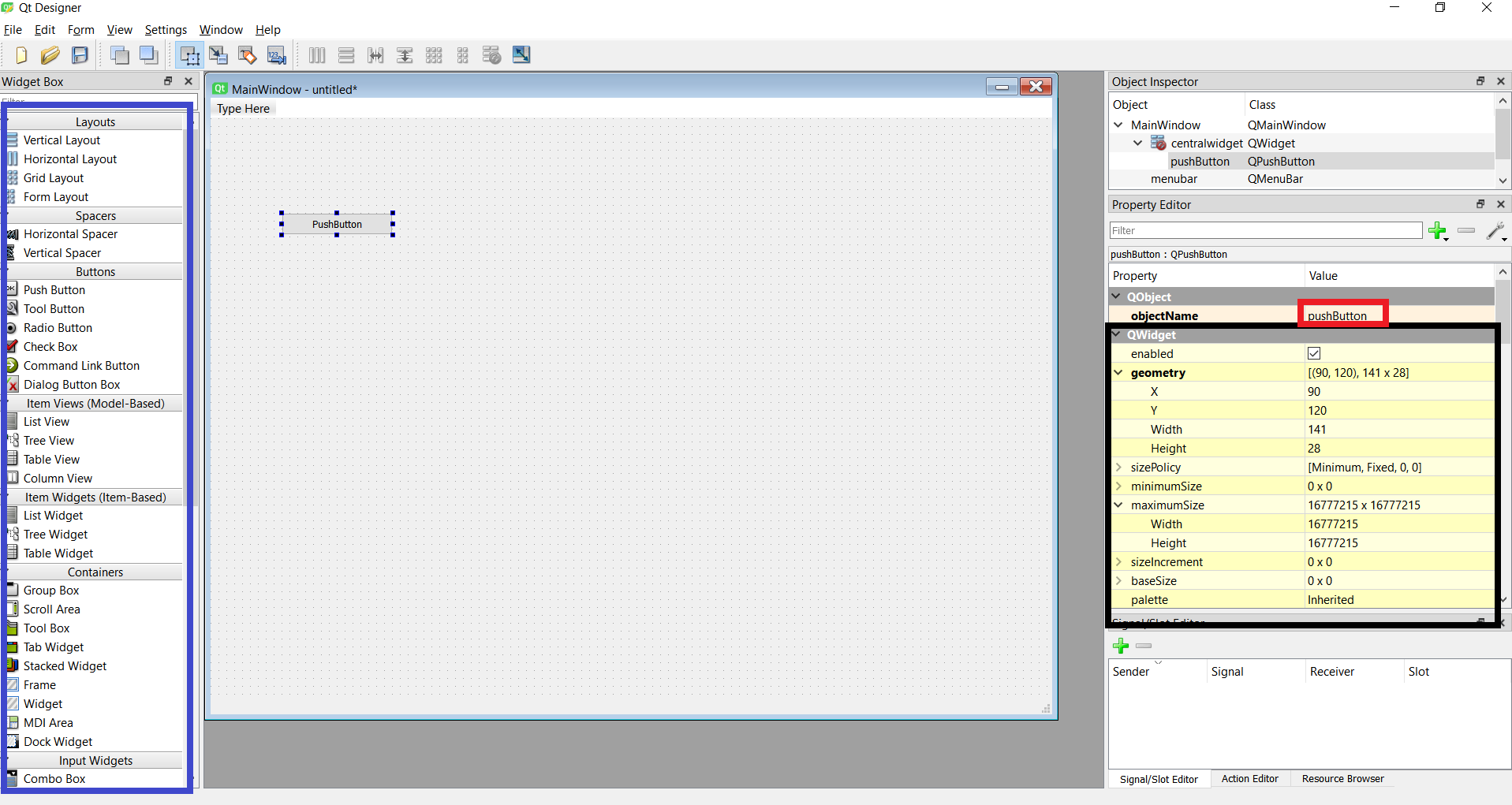
## **3.2 GUI Implementation**

The GUI has been created using Qt Designer application which is user friendly and has various options.



**Figure**: This is the starting screen of Qt Designer.

* Select Main window and click create.



3

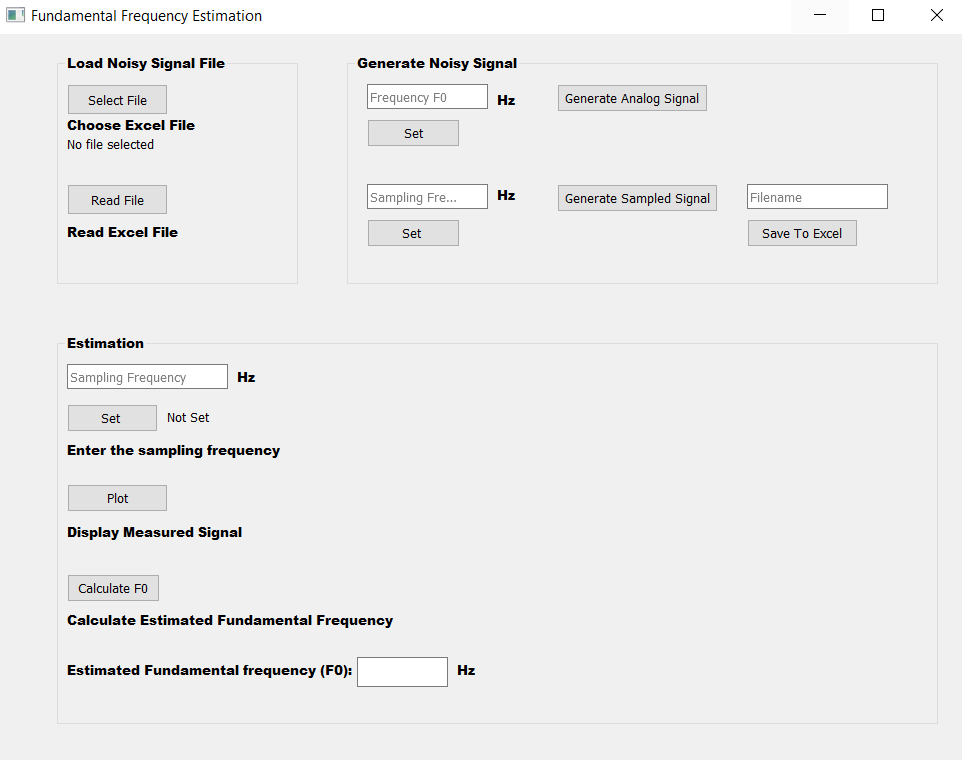
2

1

**Figure**: A main window which has a button.

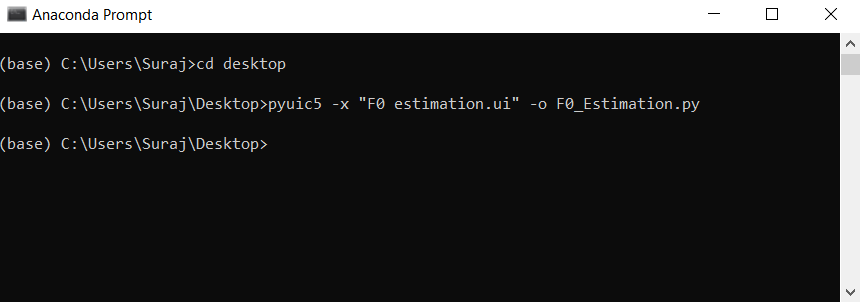
1. Elements which can be dragged and dropped to the window it has layouts, buttons, sliders, labels, etc..
2. These are properties of elements where we can adjust size of element, set name, set height and width, for labels placeholder text can be set and many more properties are there.
3. Object name can be set to make it meaningful when this GUI is converted into code.

* Push buttons, labels, Line Edit, Group box, Main window have been used in this project. The resulting GUI is shown below.



**Figure:** GUI for the Project

* After creating this GUI, the file will be saved as .ui file. It has to be converted to .py file.
* Go to the directory where .ui file is located .This is done using following command in command prompt displayed below.



* Once the file is generated, the callback functions can be linked to the respective elements by opening file in any python IDE.

# **4. Test Strategy**

# **4.1 Generating Analog Noisy signal**

* To test the implementation we need data. So first step is to generate a continuous signal with known fundamental frequency. The fundamental frequency can be set from the GUI. The number of cycles is fixed to 140, amplitude value is fixed to 2, and number of samples (N) to 140000. The calculation of the time duration of signal for a fundamental frequency is done using formula:

**Time duration=No of cycles / Fundamental Frequency** ......(4.1)

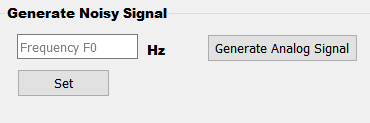
* Create an array t of size 1xN based on time duration.

**t=numpy.linspace(0,time duration, N)** ..…(4.2)

* So continuous analog signal is obtained by:

**xcont=amplitude\*sin(2\*pi\*FundamentalFreq\*t)** ..,,,.(4.3)

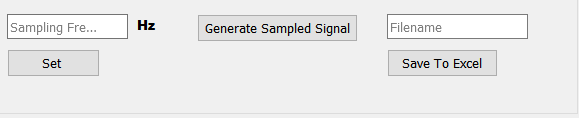
* To this continuous signal, noise is added using random function. A target noise of 10 dB is used.
* Generate Analog Signal Button in GUI generates the above signal.
* The above functionality is implemented in the **def generateNoisyAnalogSignal().**



**Figure:** A part of GUI to generate noisy analog signal.

## **4.2 Sampling Analog Signal and Storing in Excel file**

* Next step is to sample this continuous signal with a Sampling frequency which can be set in Sampling frequency field in GUI. After the sampling frequency is set, a sampled signal is generated by clicking the Generate Sampled signal button from GUI.
* This sampled signal is stored in the first row of an excel file.
* The above is implemented in the **def generateNoisySampledSignal().**



**Figure:** A part of GUI to generate noisy sampled signal base on sampling freq and saving it to file.

## **4.3 Using Algorithm to Determine Fundamental Frequency**

* When the Select File button is clicked browsing window appears, where desired file can be selected.
* The contents of the above selected file will be read when the Read File button is clicked.
* The Sampling Frequency in Hz has to be entered in the line Field and the Set button has to be clicked.
* The signal that has been read will get plotted according to the sampling frequency when the Plot button is clicked.
* The fundamental frequency will be estimated when Calculate F0 button is clicked.
* Result will be displayed in line Field in Hz.

# **5. Test Scenarios**

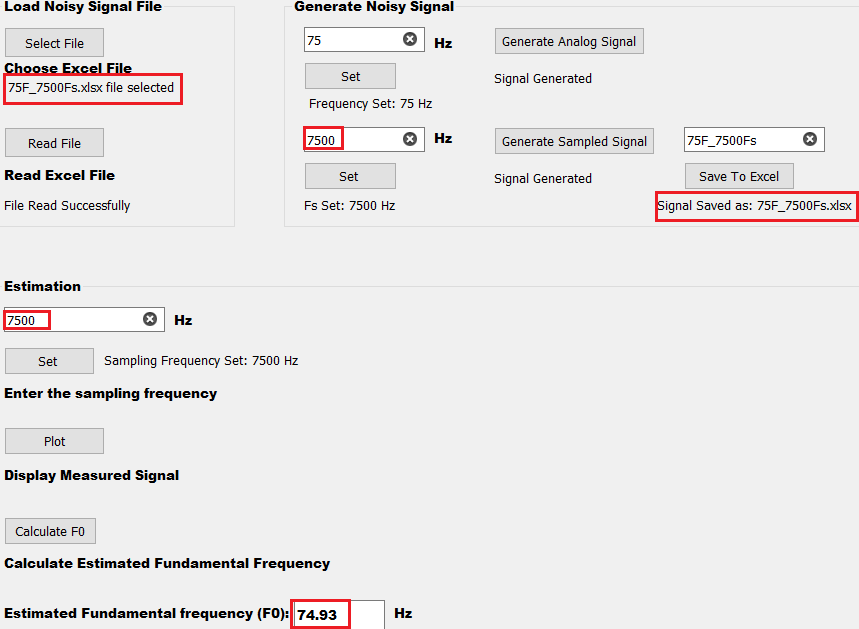
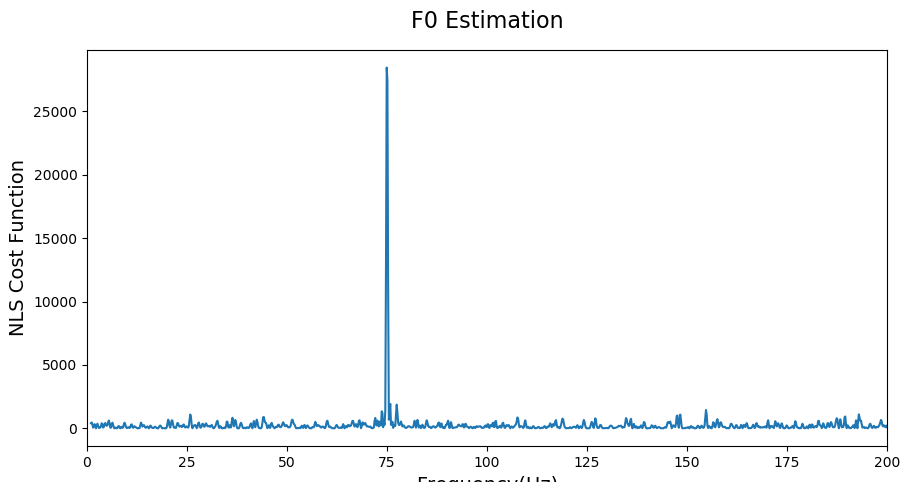
The procedure to follow the carry out Test Cases is described in Test Strategy discussed above. The test files are stored with the naming convention <Fundamental frequency>F\_<Samplingfrequency>Fs.

**Scenario 1: 0 < F0 < 100 Hz, Fs=100\*F0**

* Test Case1: F0 = 75Hz, Fs=7500Hz

*Description*: Using actual Fs sampling frequency which was used to generate the sampled signal.

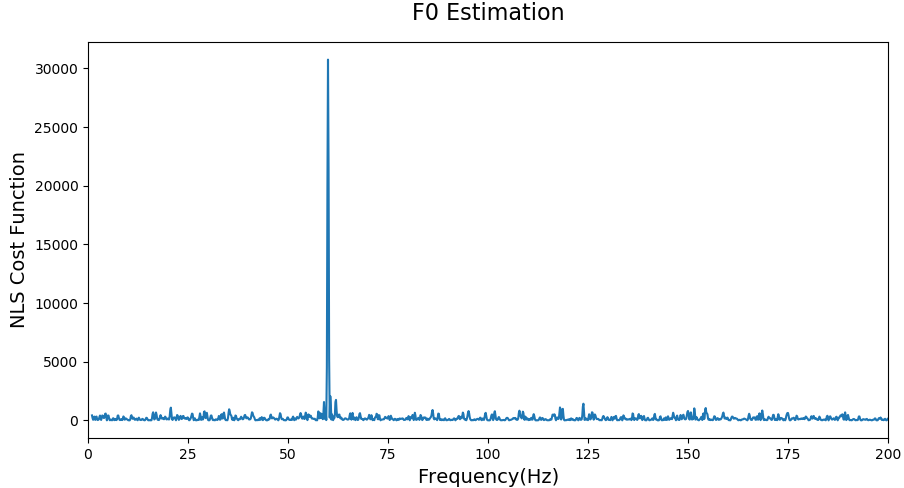
*Test* *Result*: The value of estimated Fundamental Frequency (F0) will be closer to the actual F0.

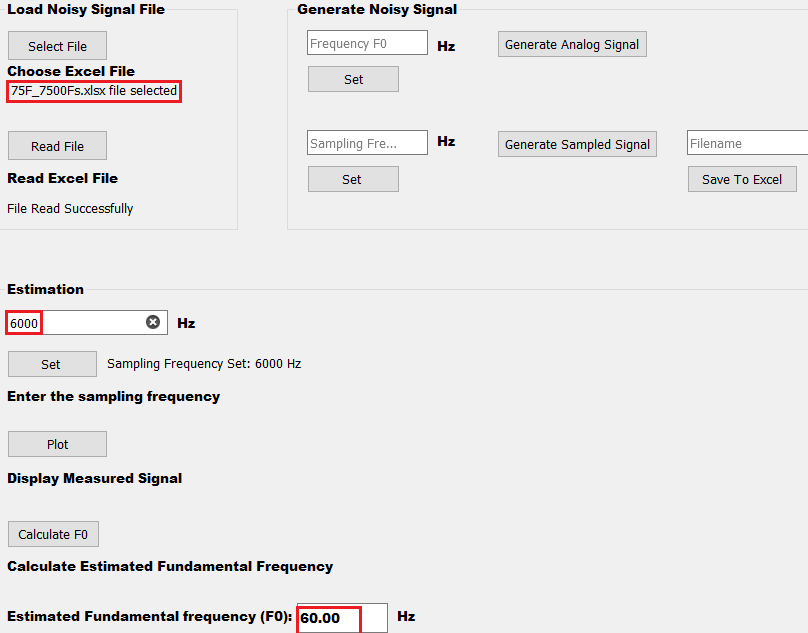


Note: Xlimit set(0,200)

* Test Case2: F0 = 75Hz, Fs=6000Hz

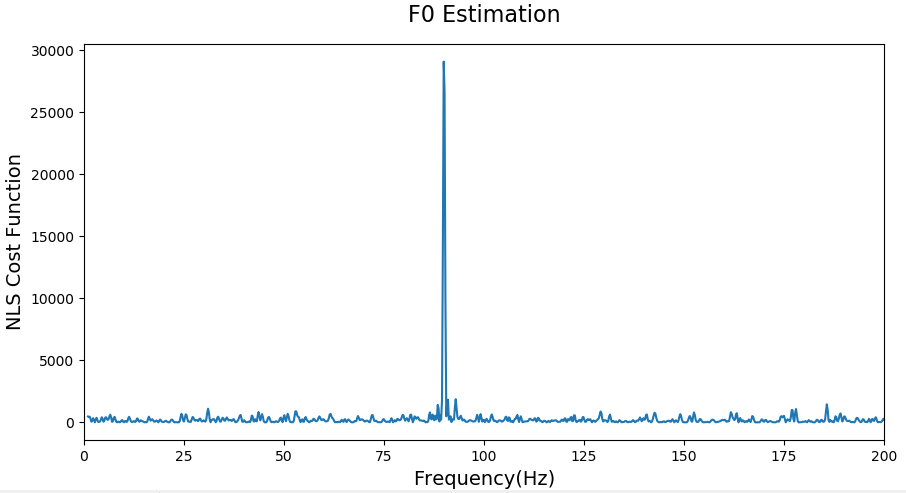
*Description*: Using Fs sampling frequency lesser than the actual Fs which was used to generate the sampled signal.

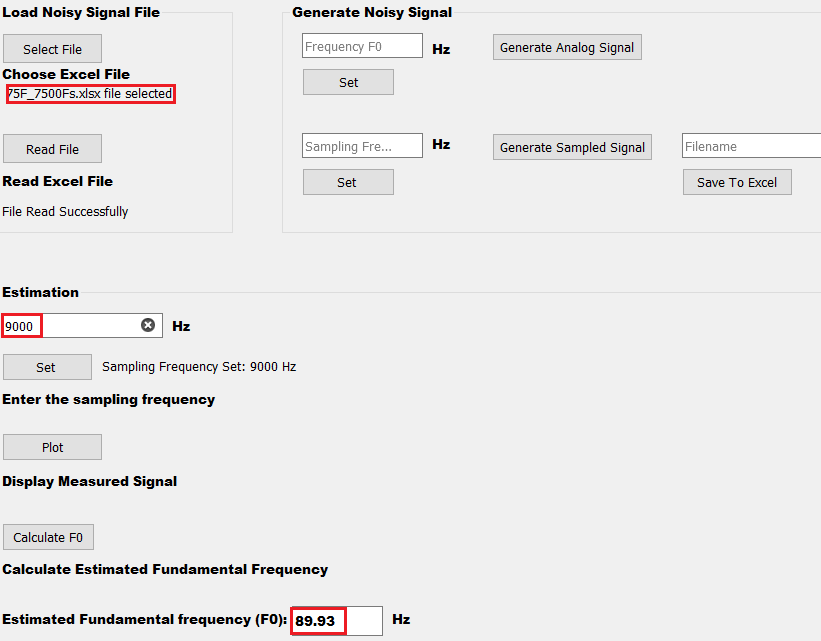
*Test Result*: The value of estimated Fundamental Frequency (F0) will be lesser than the actual F0.

Note: Xlimit set(0,200)

* Test Case3: F0 = 75Hz, Fs=9000Hz

*Description*: Using Fs sampling frequency greater than the actual Fs which was used to generate the sampled signal.

*Test Result*: The value of estimated Fundamental Frequency (F0) will be greater than the actual F0.



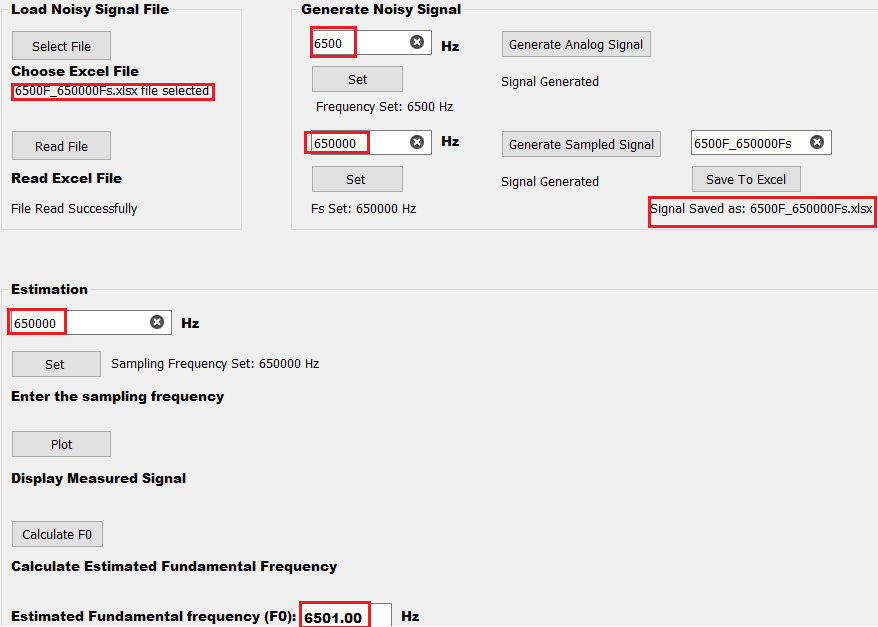
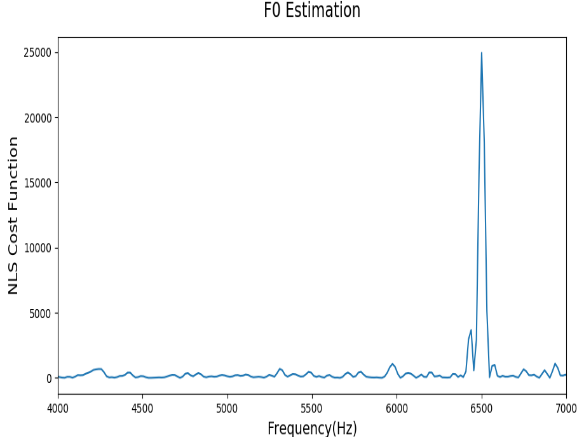
***Summary of Scenario 1:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **F0 (Hz)** | **Fs(Hz)** | **Estimated F0 (Hz)** |
| **1** | 75 | 7500 | 74.93 |
| **2** | 75 | 6000 | 60.00 |
| **3** | 75 | 9000 | 89.93 |

**Scenario 2: 100 < F0 <10 kHz, Fs=100\*F0**

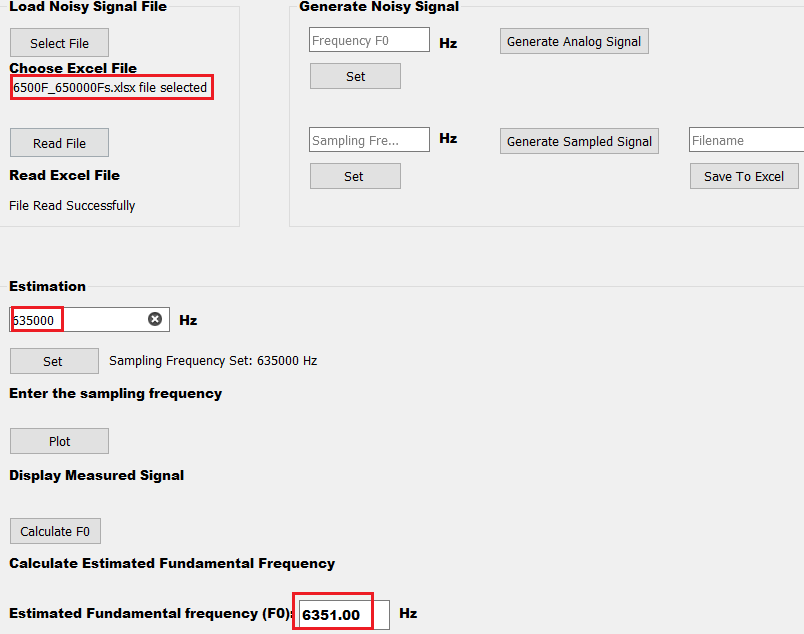
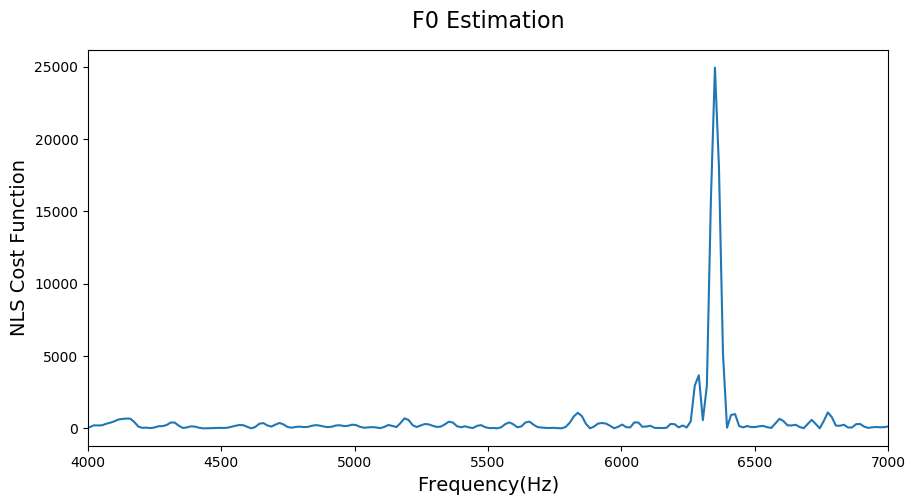
* Test Case1: F0 = 6500Hz, Fs=650 kHz

*Description*: Using actual Fs sampling frequency which was used to generate the sampled signal.

*Test Result:* The value of estimated Fundamental Frequency (F0) will be closer to the actual F0.

* Test Case2: F0 = 6500Hz, Fs=635 kHz

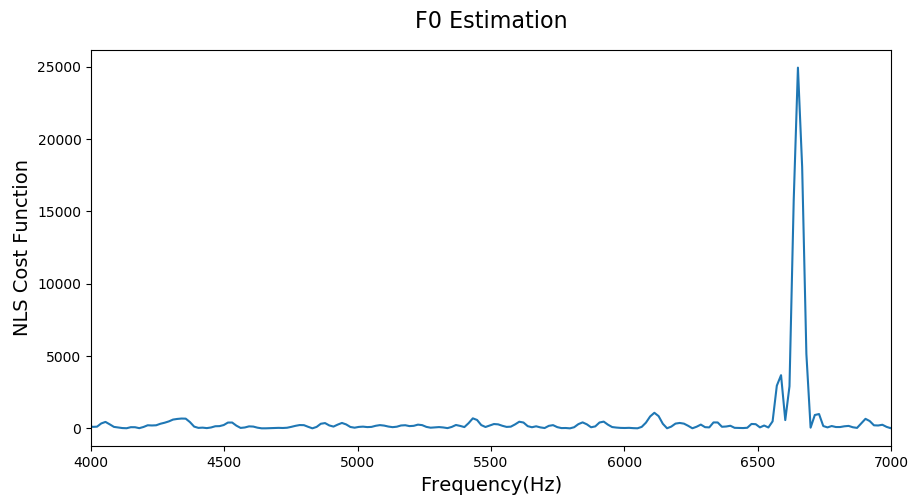
*Description*: Using Fs sampling frequency lesser than the actual Fs which was used to generate the sampled signal.

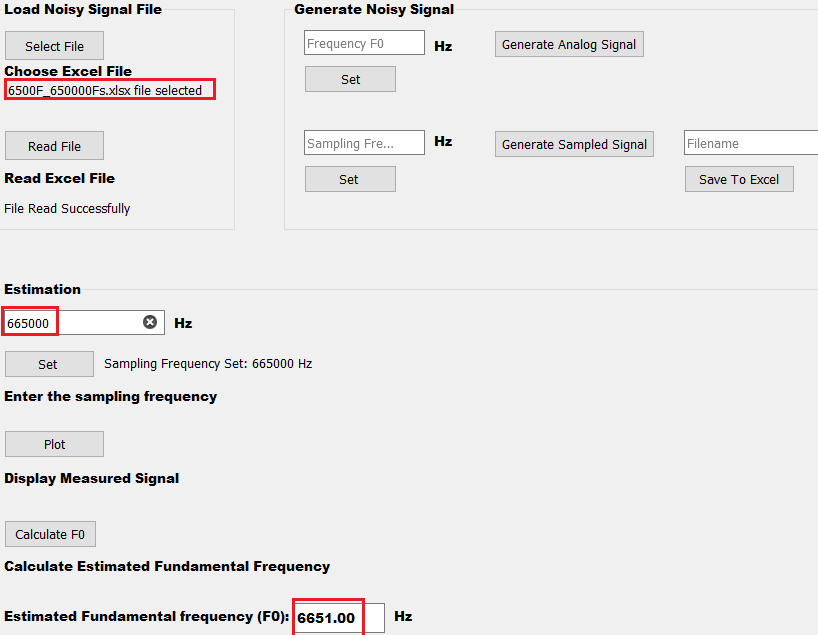
*Test Result*: The value of estimated Fundamental Frequency (F0) will be lesser than the actual F0.

Note: xlimit has been set(4000,7000)

* Test Case3: F0 = 6500Hz, Fs=665 kHz

*Description*: Using Fs sampling frequency greater than the actual Fs which was used to generate the sampled signal.

*Test Result*: The value of estimated Fundamental Frequency (F0) will be greater than the actual F0.

 Note: xlimit has been set(4000,7000)

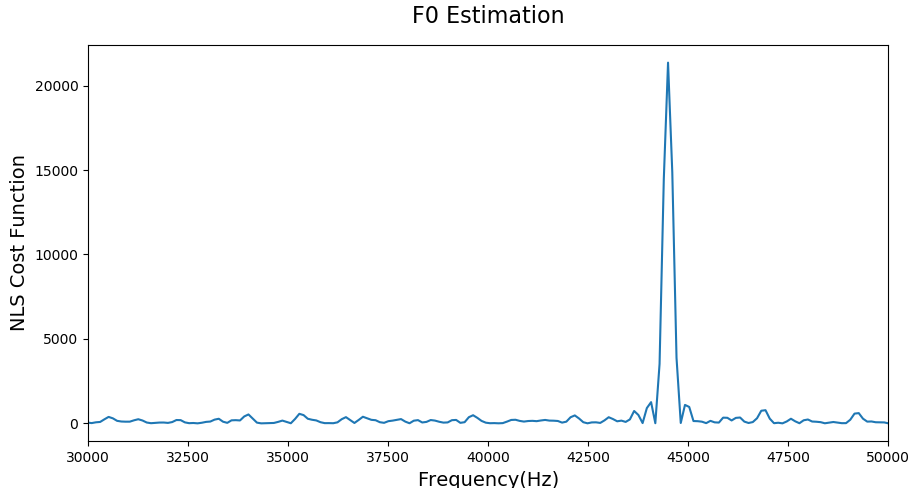
***Summary of Scenario 2:***

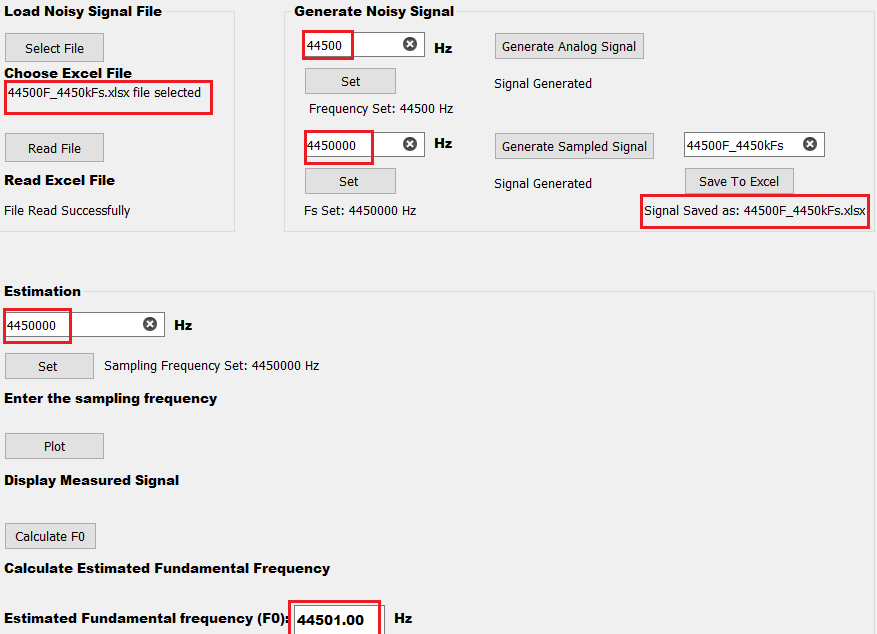
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **F0 (Hz)** | **Fs(kHz)** | **Estimated F0 (Hz)** |
| **1** | 6500 | 650 | 6501.00 |
| **2** | 6500 | 635 | 6351.00 |
| **3** | 6500 | 665 | 6651.00 |

**Scenario 3: 10 kHz < F0, Fs=100\*F0**

* Test Case1: F0 = 44.5 kHz, Fs=4450 kHz

*Description*: Using actual Fs sampling frequency which was used to generate the sampled signal.

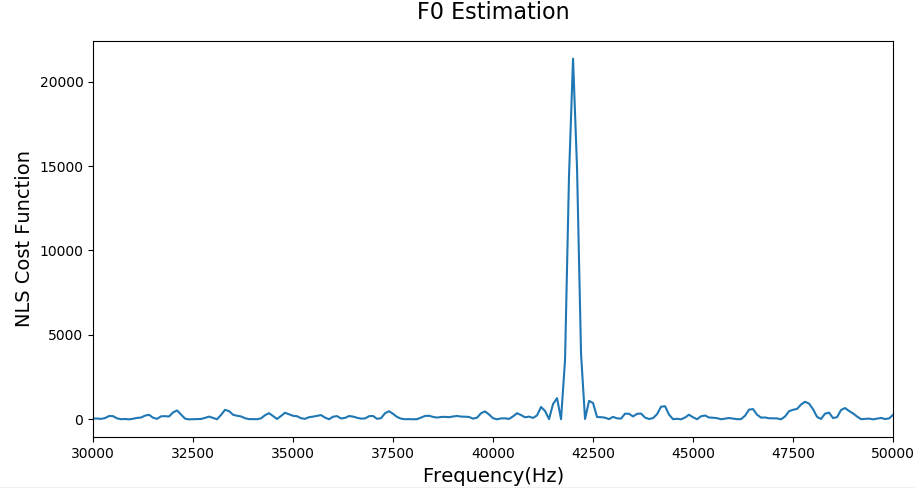
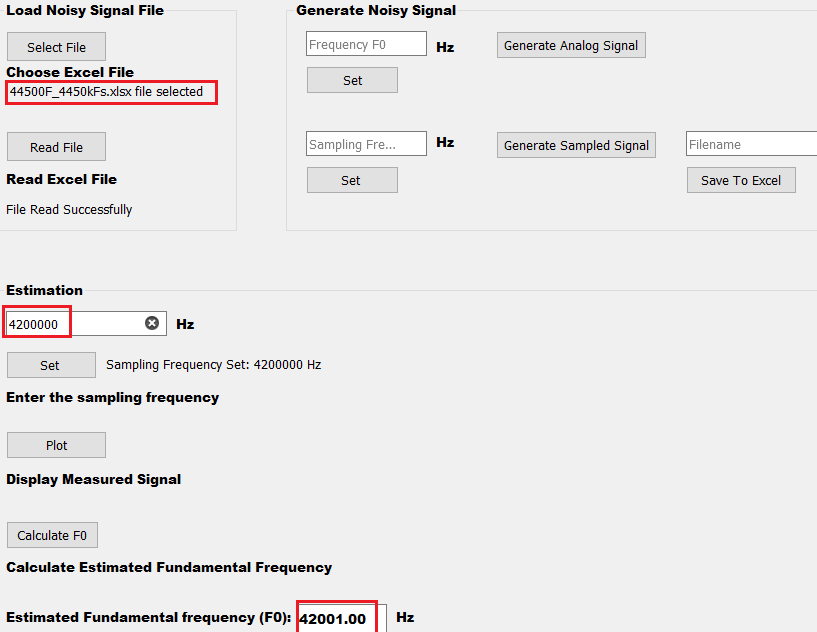
*Test Result*: The value of estimated Fundamental Frequency (F0) will be closer to the actual F0.

Note: xlimit has been set(30000,50000)

* Test Case2: F0 = 44.5 kHz, Fs=4200 kHz

*Description*: Using Fs sampling frequency lesser than the actual Fs which was used to generate the sampled signal.

*Test Result:* The value of estimated Fundamental Frequency (F0) will be lesser than the actual F0.

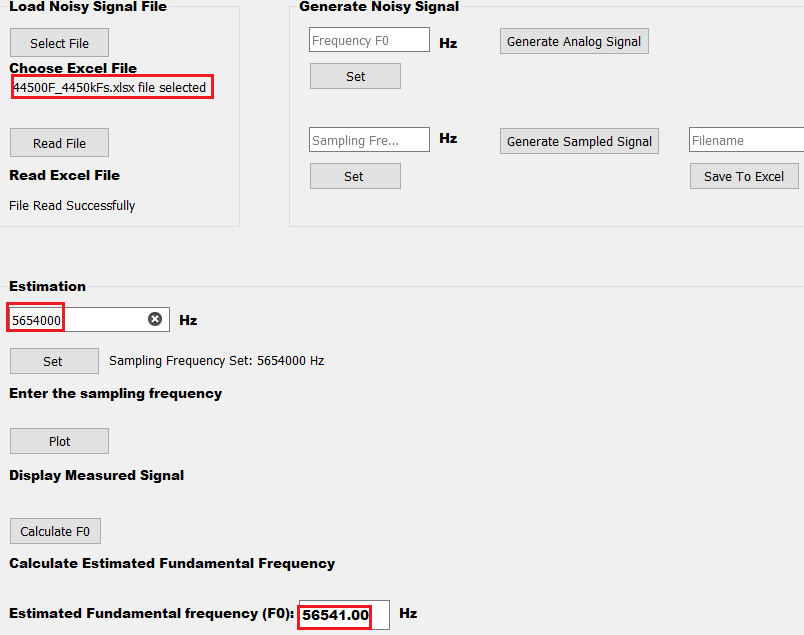


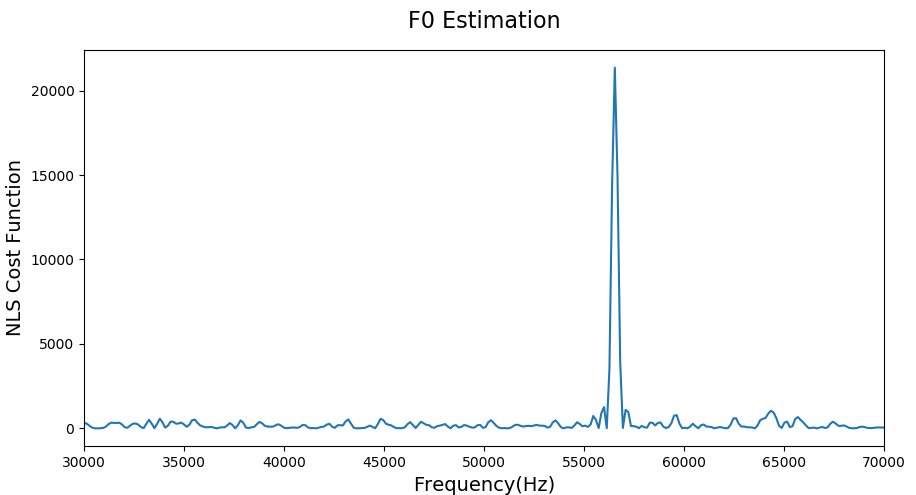
Note: xlimit has been set(30000,50000)

* Test Case3: F0 = 44.5 kHz, Fs= 5654 kHz

*Description*: Using Fs sampling frequency greater than the actual Fs which was used to generate the sampled signal.

*Test Result*: The value of estimated Fundamental Frequency (F0) will be greater than the actual F0.

****



***Summary of Scenario 3:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **F0 (kHz)** | **Fs(kHz)** | **Estimated F0 (kHz)** |
| **1** | 44.5 | 4450 | 44.501 |
| **2** | 44.5 | 4200 | 42.001 |
| **3** | 44.5 | 5654 | 56.541 |

# **6. Conclusion**

The algorithm has been implemented successfully and has provided good estimates for all range of frequencies. It has also been tested for different sampling frequencies and this estimator is better compared to autocorrelation method as it doesn’t require any pre-processing. The NLS method is statistically efficient if the noise is modelled as white and Gaussian and much more accurate than the autocorrelation based methods. But a drawback of this algorithm is that it takes time to calculate the cost function matrix as it involves matrix multiplication with an inverse matrix. This method can be made faster if implemented using recursive solver for Toeplitz-plus-Hankel systems for computing NLS cost function.

# **7. References**

[1] Fast fundamental frequency estimation: Making a statistically efficient estimator computationally efficient

http://vbn.aau.dk/da/publications/fast-fundamental-frequency-estimation(c9604a90-5140-40fa-b973-7feea1fa3ea7).html

[2] Fast Fundamental Frequency Estimation using Least Squares - Jesper Kjær Nielsen https://www.youtube.com/watch?v=F0XgU-9ERp4

[3] Signal Processing using Python 1

https://www.youtube.com/watch?v=t3AEweWweSI

1. http://web.science.mq.edu.au/~cassidy/comp449/html/ch03s03.html [↑](#footnote-ref-1)