|  |
| --- |
|  |
| SeesharpTools.JXI.DSP |
| user manual |
|  |
|  |
|  |

Document version

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Revisions | Revision person signature |
| 2017/11/11 | 1.0.0 | First Draft | 谢晓姣 |
| 2017/12/07 | 1.0.1 | Modified the Gaussian white method interface | 谢晓姣 |

Directory

[SeesharpTools.JXI.DSP 1](#_Toc16944)

[Directory 3](#_Toc1350)

[1. Conditioning 4](#_Toc17585)

[1.1. Conditioning Introduction 4](#_Toc29342)

[1.2. FIRFilter 4](#_Toc20929)

[1.2.1. FIR Filter Introduction 4](#_Toc12617)

[1.2.2. Example 5](#_Toc13517)

[1.3. IIRFilter 7](#_Toc14622)

[1.3.1. IIR Filter Introduction 7](#_Toc24578)

[1.3.2. Example 8](#_Toc28287)

[1.4. Synchronization 12](#_Toc7552)

[1.4.1. Synchronization Introduction 12](#_Toc5447)

[1.4.2. Example 12](#_Toc8964)

[1.5. Resample 14](#_Toc20551)

[1.5.1. Resample Introduction 14](#_Toc25595)

[1.5.2. Example 14](#_Toc14739)

[2. Generation 16](#_Toc7457)

[2.1. Generation Introduction 16](#_Toc16929)

[2.2. GaussianWhiteNoise 16](#_Toc11797)

[2.2.1. GaussianWhiteNoise Introduction 16](#_Toc15249)

[2.2.2. Example 16](#_Toc14470)

[3. JTFA 18](#_Toc17691)

[3.1. JTFA Introduction 18](#_Toc31213)

[3.1.1. Example 20](#_Toc23792)

[4. Measurement 22](#_Toc15343)

[4.1. Measurement Introduction 22](#_Toc8589)

[4.2. FrequencyResponseFunction 22](#_Toc22737)

[4.2.1. FrequencyResponseFunction Introduction 22](#_Toc27353)

[4.2.2. Example 22](#_Toc31922)

[4.3. HarmonicAnalysis 24](#_Toc24838)

[4.3.1. HarmonicAnalysis Introduction 24](#_Toc20831)

[4.3.2. Example 25](#_Toc10115)

[4.4. SquarewaveMeasurements 27](#_Toc29183)

[4.4.1. SquarewaveMeasurements Introduction 27](#_Toc9350)

[4.4.2. Example 27](#_Toc11498)

[4.5. ThirdOctaveAnalysis 29](#_Toc29185)

[4.5.1. ThirdOctaveAnalysis Introduction 29](#_Toc3430)

[4.5.2. Example 29](#_Toc10990)

[5. Spectrum 31](#_Toc13344)

[5.1. Spectrum Introduction 31](#_Toc8286)

[5.2. Example 31](#_Toc31897)

[6. LicenseBase 35](#_Toc7092)

# Conditioning

## Conditioning Introduction

Conditioning includes FIR Filter，IIR Filter，and Synchronization。

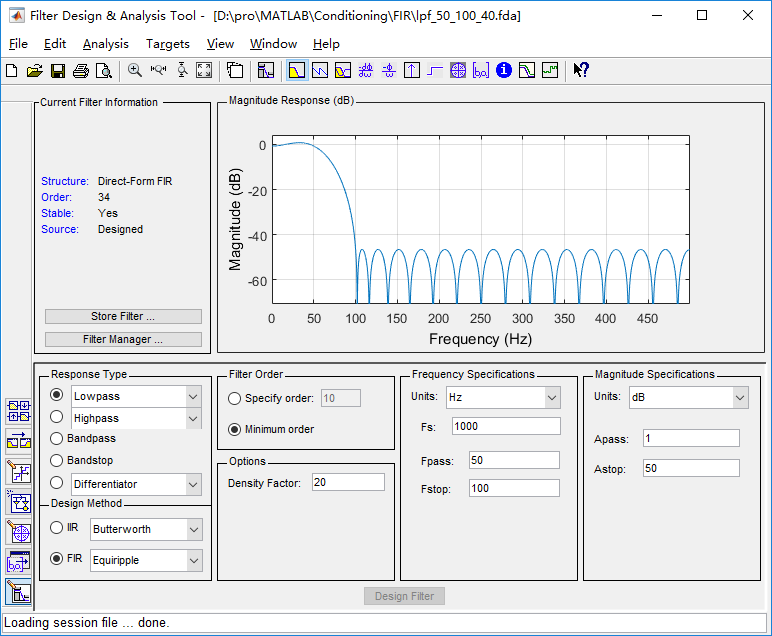
## FIRFilter

### FIR Filter Introduction

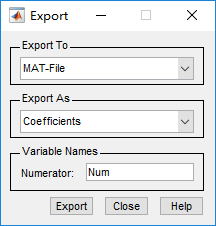
There are 2 steps to implementing a direct FIR filter:

1. Configure filter coefficients using the fir.Coefficients property
2. Call Filter to filter.

The coefficients of the filter can be designed using MATLAB's FDA tool. Then export the coefficients.



Pic 1.1 FIR Filter Design



Pic 1.2 Export FIR filter coefficients

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LPF

\* passFreq=50Hz

\* stopFreq=100Hz

\* 带外衰减40db

\* 采样频率1000Hz

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double[] coe = new double[] { 0.00194150175769345,- 0.00169793575520208,- 0.00426956799325628,- 0.00829350380976647,- 0.0132303432212115,- 0.0181544335557185,- 0.0217585679131809,- 0.0225134101549763,- 0.0189478446411358,- 0.00995128109977753,0.00489205937905203,0.0250974062726473, 0.0492006450550267, 0.0748863114260257, 0.0992878499995790, 0.119424371366942, 0.132695897659600, 0.137328720011526, 0.132695897659600 ,0.119424371366942 , 0.0992878499995790,0.0748863114260257,0.0492006450550267,0.0250974062726473, 0.00489205937905203,- 0.00995128109977753,- 0.0189478446411358,- 0.0225134101549763,- 0.0217585679131809,- 0.0181544335557185,- 0.0132303432212115 - 0.00829350380976647,- 0.00426956799325628,- 0.00169793575520208,0.00194150175769345 };

double[] sinWave1 = new double[1000];

double[] sinWave2 = new double[1000];

Generation.SineWave(ref sinWave1, 1, 0, 10, 1000);

Generation.SineWave(ref sinWave2, 1, 0, 120, 1000);

ArrayCalculation.Add(sinWave1, sinWave2, ref sinWave1);

double[] sinWaveFir = new double[1000];

var fir = new FIRFilter();

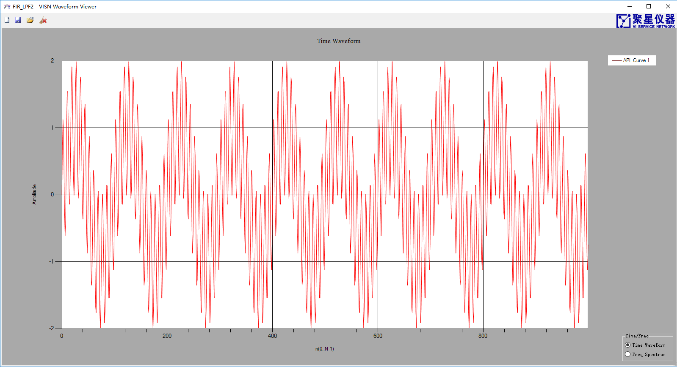
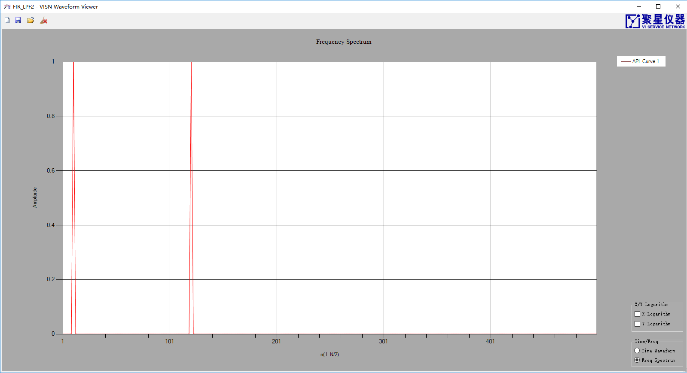
fir.Coefficients = coe;

fir.Filter(sinWave1, ref sinWaveFir);

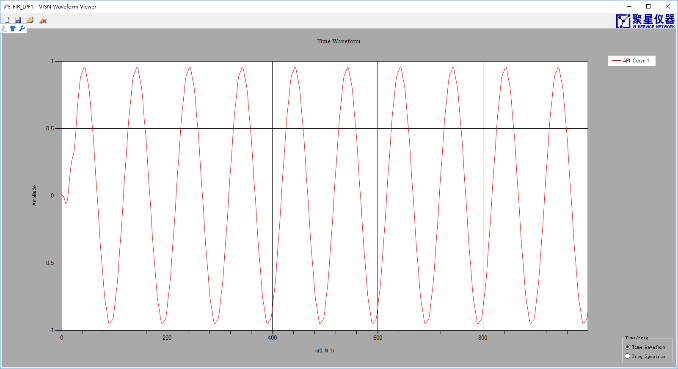
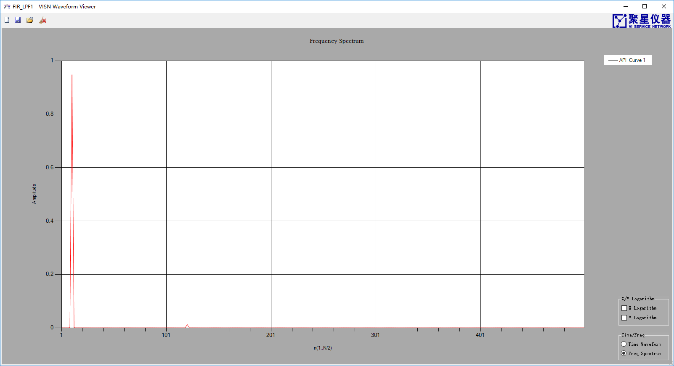
VISN\_Plot.PlotData("FIR\_LPF1", sinWaveFir);

VISN\_Plot.PlotData("FIR\_LPF2", sinWave1);

Results:



Pic 1.3 Pre-filtered signal and spectrum



Pic 1.4 Filtered signal and spectrum

**Note: After changing the filter coefficients, you need to call Reset() to clear the status register!**

## IIRFilter

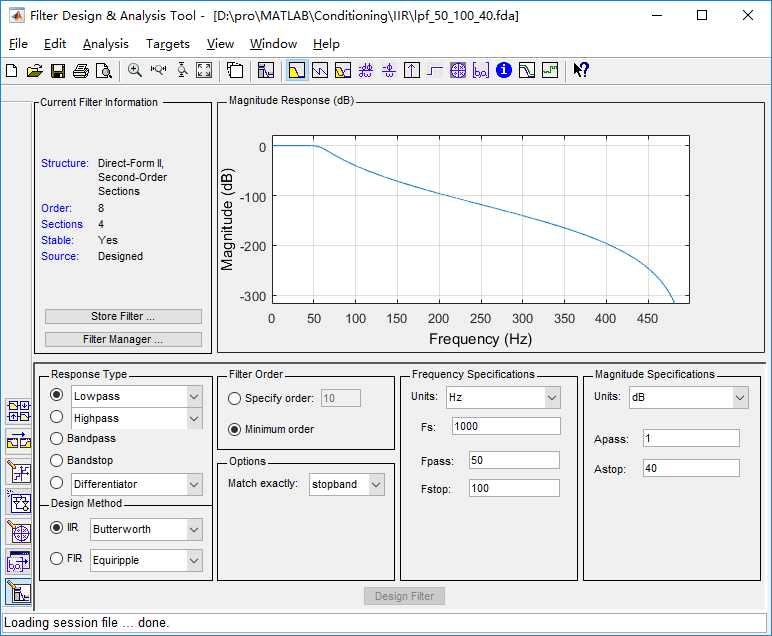
### IIR Filter Introduction

There are 2 steps to implementing a direct Type II IIR filter:

（1）Use SetCoefficients() to configure the filter coefficients.

（2）Call Filter to filter.

The coefficients of the filter can be designed with FDA tool from MATLAB. Then you can export the coefficients.



Pic 1.5 IIR Filter Design

After designing the filter, select Export in the File drop-down menu to get the matrix SOS and G.

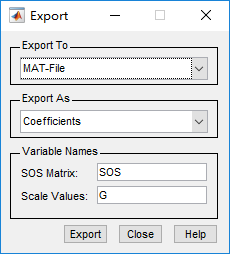
Use [b, a] = sos2tf (SOS) to obtain the numerator and denominator polynomial coefficients of the filter function, but with a difference factor k.

K=cumprod(G);

k=K(end);

The filtered output result filteredPWave can be obtained as follows:

filteredpWave=filter(b,a,pWave)\*k;



Pic 1.6 Export coefficient

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LPF

\* passFreq=50Hz

\* stopFreq=100Hz

\* 带外衰减40db

\* 采样频率1000Hz

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double[] sinWave1 = new double[1000];

double[] sinWave2 = new double[1000];

Generation.SineWave(ref sinWave1, 1, 0, 10, 1000);

Generation.SineWave(ref sinWave2, 1, 0, 120, 1000);

ArrayCalculation.Add(sinWave1, sinWave2, ref sinWave1);

double[] sinWaveIir = new double[1000];

double[] numerator = new double[] { 1, 8, 28, 56, 70, 56, 28, 8, 1 };

double[] demoniatorr = new double[] { 1,- 6.14833128577259 , 16.7106178322332, - 26.1940863217919 , 25.8773677909964 ,- 16.4863518823100 ,6.61072693847043, - 1.52457047404531 , 0.154752511303108 };

double k = 4.887073568907229e-07;

var iir = new IIRFilter();

iir.SetCoefficients(numerator, demoniatorr);

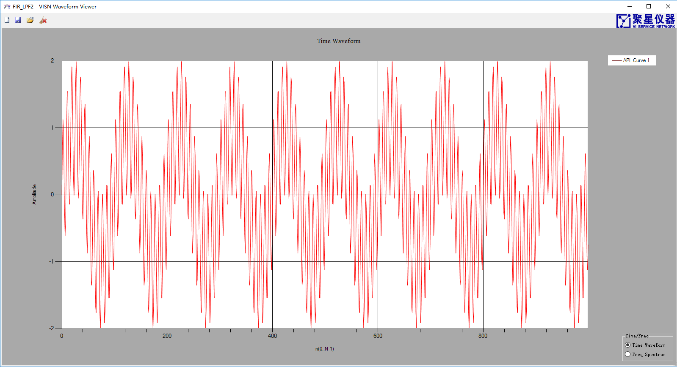
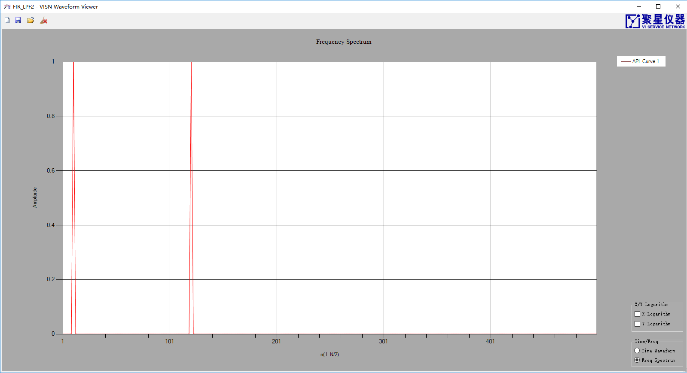
iir.Filter(sinWave1, ref sinWaveIir);

ArrayCalculation.MultiplyScale(ref sinWaveIir, k);

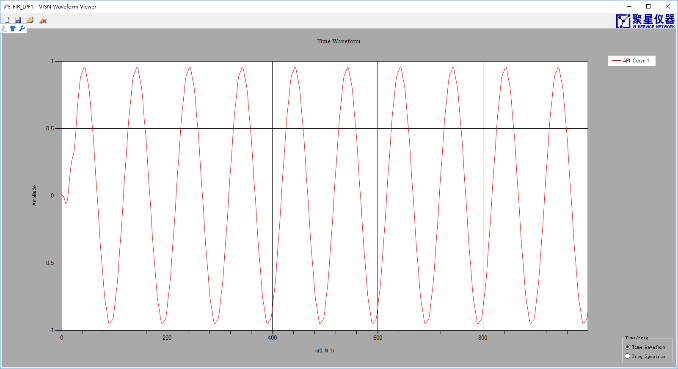
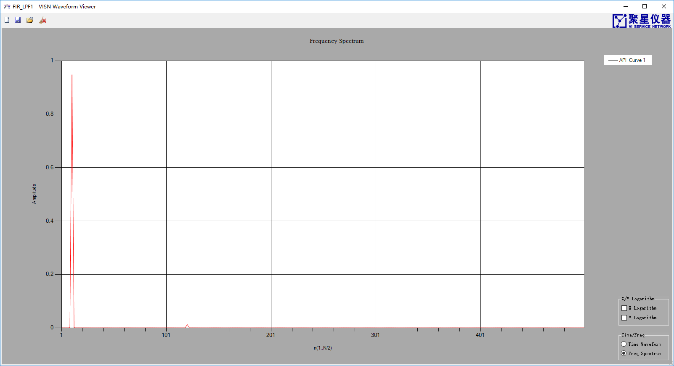
VISN\_Plot.PlotData("IIR\_LPF1", sinWaveIir);

VISN\_Plot.PlotData("IIR\_LPF2", sinWave1);

Results:



Pic 1.7 Pre-filtered signal and spectrum



Pic 1.8 Filtered signal and spectrum

using MathNet.Numerics.Data.Matlab;

using MathNet.Numerics.LinearAlgebra;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LPF

\* passFreq=0.2

\* stopFreq=0.3

\* 带外衰减40db

\* 采样频率1000Hz

\* SOS

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double[] sinWave1 = new double[1000];

double[] sinWave2 = new double[1000];

Generation.SineWave(ref sinWave1, 1, 0, 10, 1000);

Generation.SineWave(ref sinWave2, 1, 0, 350, 1000);

ArrayCalculation.Add(sinWave1, sinWave2, ref sinWave1);

string path = Environment.CurrentDirectory;

//清空系数矩阵

double [,] SOS = null;

double [] G = null;

//读.mat文件

Matrix<double> MATLAB\_SOS = MatlabReader.Read<double>(path+ @"\lpf\_0p2\_0p3\_40db\_SOS.mat", "SOS");

//创建滤波器系数数组

SOS = new double[MATLAB\_SOS.RowCount,MATLAB\_SOS.ColumnCount];

//获取滤波器系数

var tmp\_SOS = ((MathNet.Numerics.LinearAlgebra.Double.DenseMatrix)MATLAB\_SOS).Values;

//拷问数组

for (int i = 0; i < SOS.GetLength(0); i++)

{

for (int j = 0; j < SOS.GetLength(1); j++)

{

SOS[i, j] = tmp\_SOS[i + j\*( SOS.GetLength(1)-1)];

}

}

//读.mat文件

Matrix<double> MATLAB\_G = MatlabReader.Read<double>(path + @"\lpf\_0p2\_0p3\_40db\_SOS.mat", "G");

//创建滤波器系数数组

G = new double[ MATLAB\_G.RowCount];

//获取滤波器系数

var tmp\_G = ((MathNet.Numerics.LinearAlgebra.Double.DenseMatrix)MATLAB\_G).Values;

//拷问数组

Array.Copy(tmp\_G, G, MATLAB\_G.RowCount);

double[] sinWaveIir = new double[1000];

var iir = new IIRFilter();

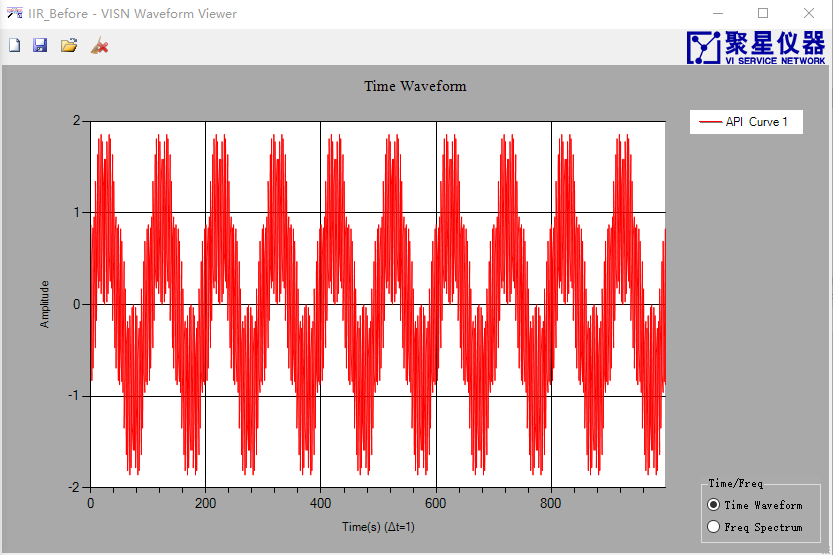
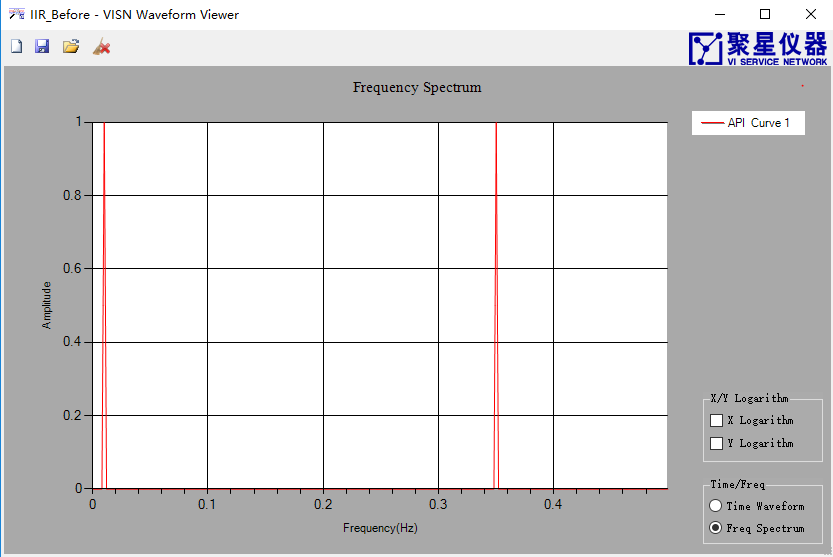
iir.IsSOS = true;

iir.SetCoefficients(SOS, G);

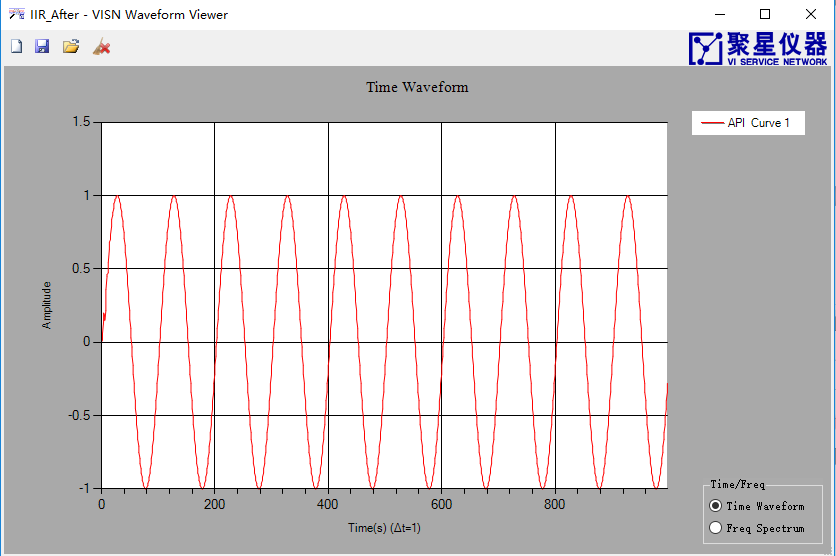
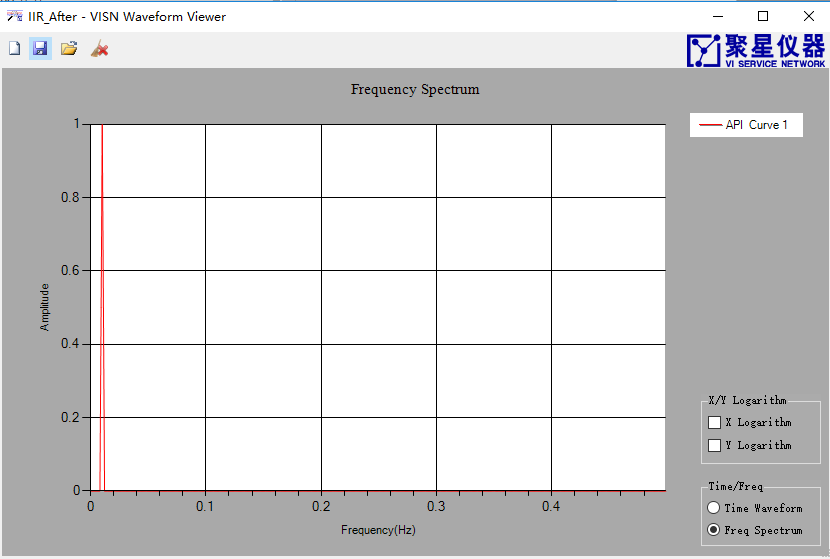
iir.Filter(sinWave1, ref sinWaveIir);

AnalogWaveformChart.Plot("IIR\_After", sinWaveIir);

AnalogWaveformChart.Plot("IIR\_Before", sinWave1);



Pic 1.9 Pre-filtered signal and spectrum



Pic 1.10 Filtered signal and spectrum

**Note: After changing the filter coefficients, you need to call Reset() to clear the status register!**

## Synchronization

### Synchronization Introduction

There is 1 step to achieve multi-channel data synchronization of the scan capture card:

（1）Call SyncWaveform(double[,] data, double ChannelShiftSamples) to synchronize the data. ChannelShiftSamples are sampling points.

### Example

double[] sinWave1 = new double[1000];

double[] sinWave2 = new double[1000];

Generation.SineWave(ref sinWave1, 1, 0, 10, 1000);

Generation.SineWave(ref sinWave2, 1, 360.0/2000, 10, 1000);

double[,] rawWaveform = new double[1000, 2];

ArrayManipulation.Connected\_2D\_Array(sinWave1, sinWave2, ref rawWaveform);

double[,] rawWaveformT = new double[rawWaveform.GetLength(1), rawWaveform.GetLength(0)];

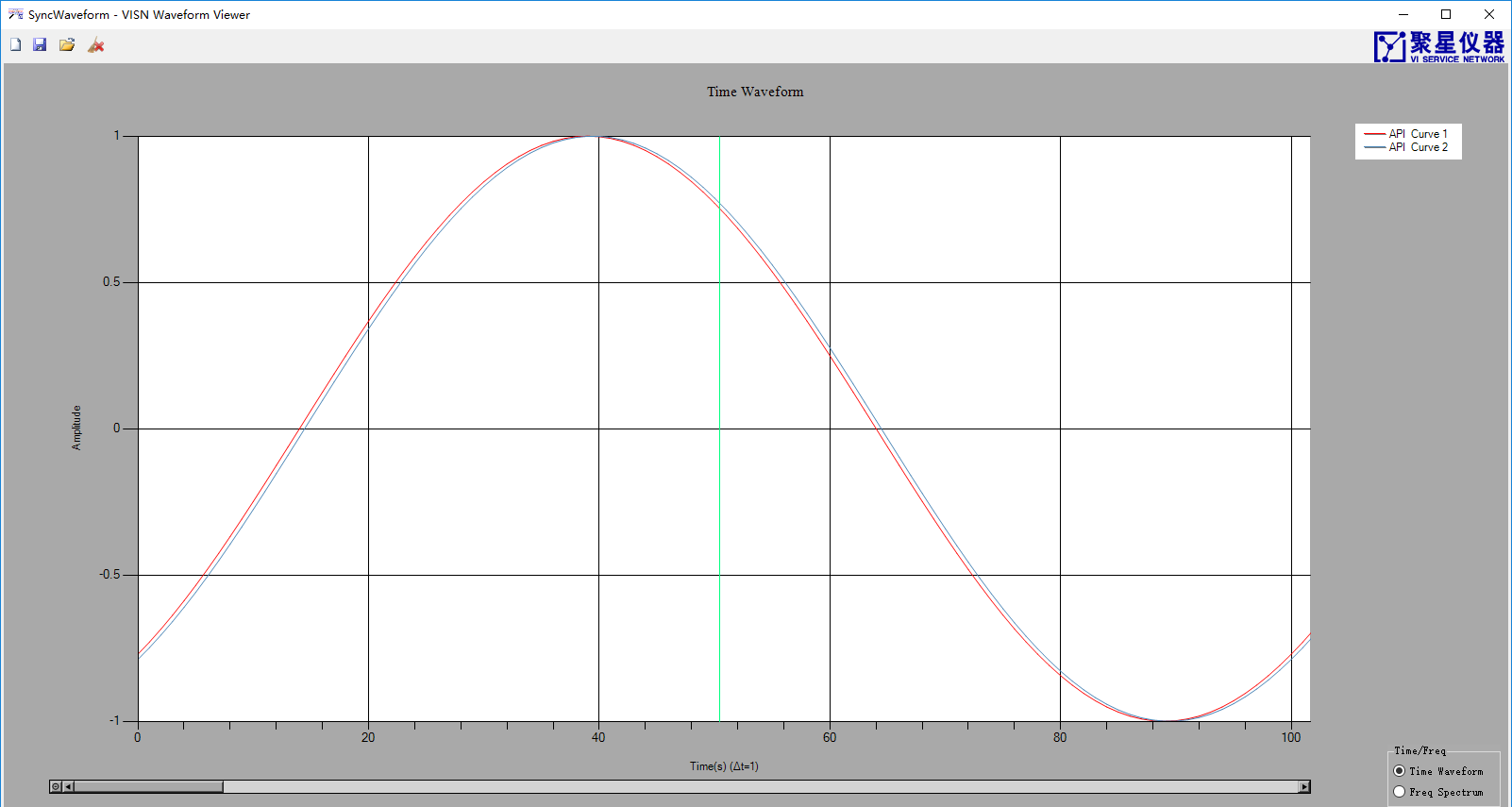
ArrayManipulation.Transpose(rawWaveform, ref rawWaveformT);

var syncWaveform = Synchronization.SyncWaveform(rawWaveformT, 0.5);

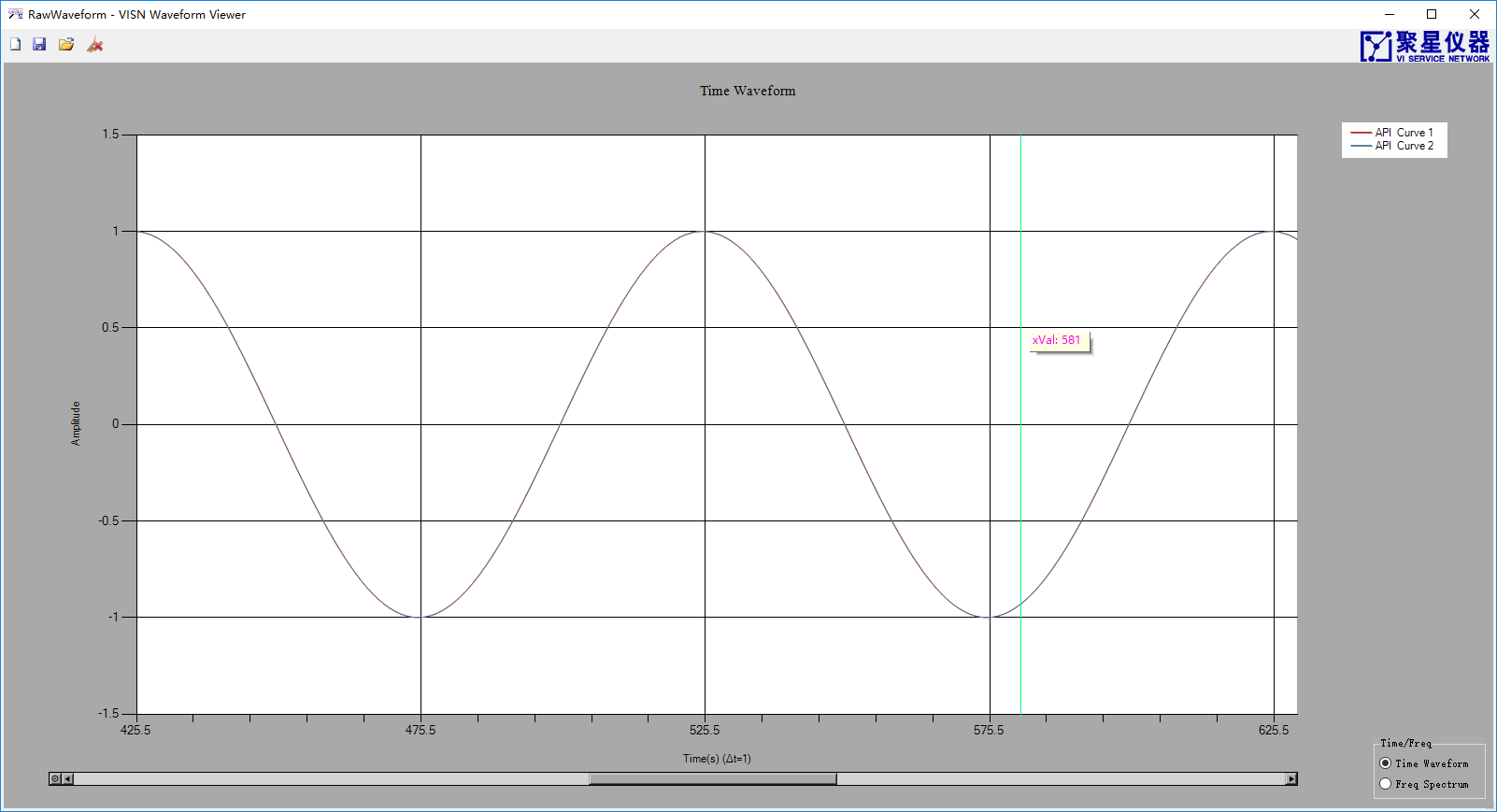
AnalogWaveformChart.Plot("RawWaveform", rawWaveformT);

AnalogWaveformChart.Plot("SyncWaveform", syncWaveform);

Result：



Pic 1.11Pre-filtered



Pic 1.12 Filtered

## Resample

### Resample Introduction

Provide multiple overwrites:

double[] ResampleWaveform(double[] data, double delay, double dt)

double[] ResampleWaveform(double[] data, double delay）

Complex[] ResampleWaveform(Complex[] data, double delay, double dt)

Where data is the input data, delay is the delay (the unit is the number of sampling points), and dt is the new sampling interval (the unit is the number of sampling points). For example, delay=0.5, dt=0.1, the sampling rate of the resampled data is the input signal, 10 times and offset by 0.5 sample points (in terms of the sample rate of the input data).

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 改变采样率，重采样后的信号与按照新采样率采样的信号只存

\*在一个时延

\* dt=0.3

\* delay=0;

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double dt = 0.3;

double[] sinWave1 = new double[1000];

double[] sinWave2 = new double[(int)((sinWave1.Length -2\*85)/dt)];

double sampleRate = 1000;

double freq = 10;

JY.DSP.Fundamental.Generation.SineWave(ref sinWave1, 1, 0, freq, sampleRate);

JY.DSP.Fundamental.Generation.SineWave(ref sinWave2, 1, 85\*360\*freq/sampleRate, freq, sampleRate/dt);

var ResWaveform2 = Resample.ResampleWaveform(sinWave1, 0, dt);

double[,] rawWaveform = new double[sinWave2.Length , 2];

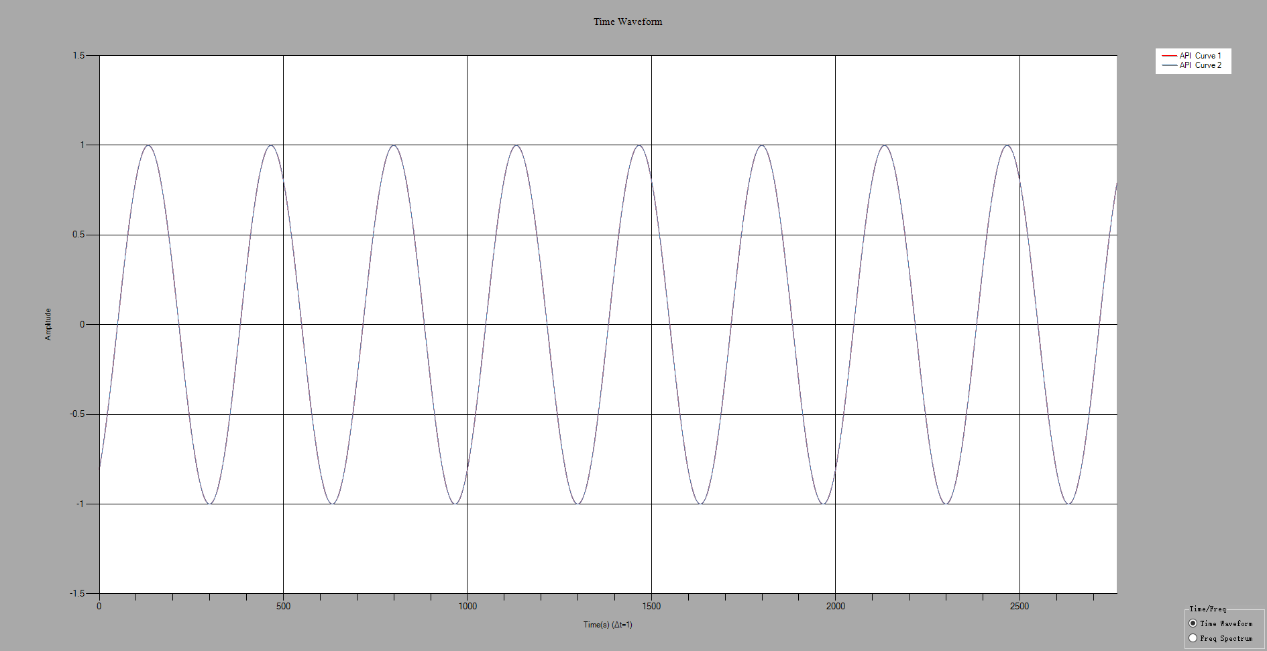
ArrayManipulation.Connected\_2D\_Array(ResWaveform2 , sinWave2, ref rawWaveform);

double[,] rawWaveformT = new double[rawWaveform.GetLength(1), rawWaveform.GetLength(0)];

ArrayManipulation.Transpose(rawWaveform, ref rawWaveformT);

AnalogWaveformChart.Plot("ResWaveform2 ", rawWaveformT);

Result：



Pic 1.13 Re-sampling

# Generation

## Generation Introduction

Generation includes the generation of Gaussian white noise.

## GaussianWhiteNoise

### GaussianWhiteNoise Introduction

If a noise’s instantaneous value complies with Gaussian distribution (normal distribution, u = 0), and its power spectral density is evenly distributed, it is called Gaussian white noise.

The expression of the probability density function of a normal distribution is as follows:

 Formula 2.1

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 产生服从标准正态分布的高斯白噪声

\* 标准差：1

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length = 1000000;

var noise = new double[Length];

int[] histgram=new int[500];

double[] histgramTemp = new double[histgram.Length];

double[] intervals = new double[500];

JXI.DSP.Generation.Generation.GaussianWhiteNoise(ref noise, 1);

ProbabilityStatistics.Histogram(noise, ref histgram, ref intervals);

for (int i = 0; i < histgram .Length ; i++)

{

histgramTemp[i] = (double)histgram[i];

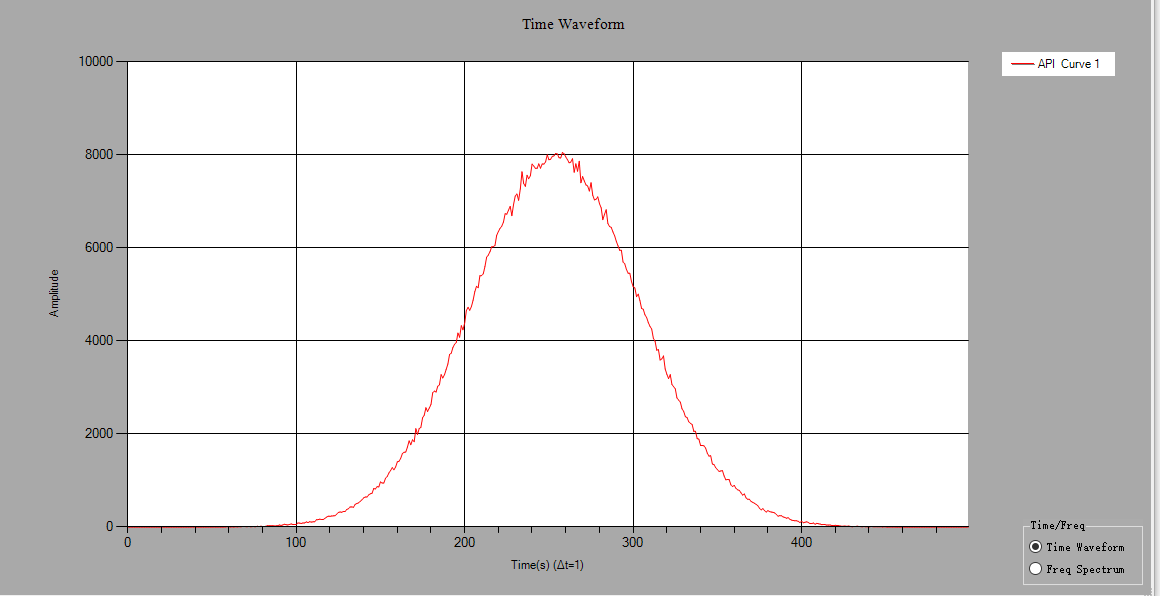
}

AnalogWaveformChart.Plot("histgram", histgramTemp);

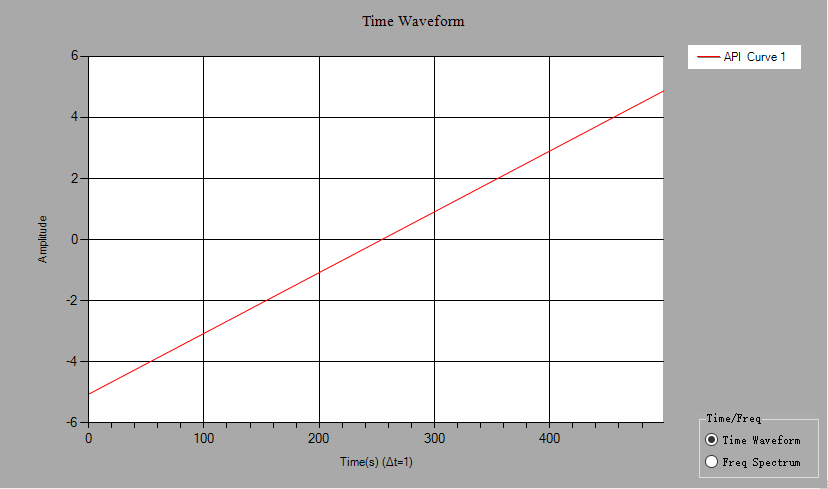
AnalogWaveformChart.Plot("intervals", intervals);

AnalogWaveformChart.Plot("noise", noise);

Result：



Pic 2.1 Histogram



Pic 2.2 Intervals

# JTFA

## JTFA Introduction

JTFA(Time-frequency joint analysis)，can obtain a two-dimensional array of time-frequency analysis of the signal, the row represents the frequency (increment), and the column represents time (increment) or intensity map (red is strong).

Assume that the time domain signal  is as shown in Pic 3.1, the frequency and amplitude change with time.



Pic 3.1 Time domain signal 



Pic 3.2 Window sliding

Assume that the length of the window is WindowLength, and the window sliding interval is WindowLength/4 (rounded). As shown in the figure, the time domain signal in the window is subjected to short time Fourier transform (STFT) to obtain a one-dimensional array representing the spectrum (i is the number of sliding windows, for complex signals, for real numbers), resulting in a two-dimensional array:



This results in a two-dimensional array containing time, frequency, and amplitude information.

**Note: When using the same task to perform multiple JTFA operations, if you do not want the saved data of the previous operation to affect the next operation, you need Reset(). The output spectrum and the intensity map have the abscissa as the frequency and the ordinate as the time.**

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 输入信号为实数

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length = 1000;

double[] sin = new double[Length];

double[] noise = new double[Length];

double freqSin = 10;

Generation.SineWave(ref sin, 1, 0, freqSin, 1000);

Generation.UniformWhiteNoise(ref noise, 0.05);

ArrayCalculation.Add(sin, noise, ref sin);

JTFATask task = new JTFATask();

task.ColorTable = JTFATask.ColorTableType.Rainbow;

task.WindowType = FFTWindowType.None;

task.FrequencyBins = 500;

task.SampleRate = 1000;

double[,] JTFASpectrum = new double[task.FrequencyBins / 2, 5];

double df;

task.GetJTFA(sin, ref JTFASpectrum,out df);

Bitmap image = new Bitmap(750, 250);

task.GetImage(JTFASpectrum, ref image);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 输入信号为复数

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length = 1000;

double[] sin = new double[Length];

double[] cos = new double[Length];

double freqSin = 10;

JY.DSP.Fundamental.Generation.SineWave(ref sin, 1, 0, freqSin, 1000);

JY.DSP.Fundamental.Generation.SineWave(ref cos, 1, 90, freqSin, 1000);

Complex[] signal = new Complex[Length];

for (int i = 0; i < Length; i++)

{

signal[i] = new Complex(cos[i], sin[i]);

}

JTFATask task = new JTFATask();

task.ColorTable = JTFATask.ColorTableType.Rainbow;

task.WindowType = FFTWindowType.None;

task.FrequencyBins = 500;

task.SampleRate = 1000;

double[,] JTFASpectrum = new double[task.FrequencyBins , 5];

double df;

double valueFreq = 1.0 / 2;

task.GetJTFA(signal, ref JTFASpectrum, out df);

Bitmap image = new Bitmap(750, 250);

task.GetImage(JTFASpectrum, ref image);

# Measurement

## Measurement Introduction

Measurement includes frequency response function, harmonic analysis, square wave measurement, 1/3 octave analysis.

## FrequencyResponseFunction

### FrequencyResponseFunction Introduction

Provides a system amplitude response and a phase response function;

①Amplitude response：double[] GetMagenitude(double[] inputWaveform, double[] outputWaveform, bool inDB, AverageParam Average)

②Phase response：double[] GetPhase(double[] inputWaveform, double[] outputWaveform,bool inDegree,AverageParam Average)

InputWaveform is the input system signal, outputWaveform is the inputWaveform signal after proceeding through the system. If inDB/inDegree is true, the system response is expressed in dB, and Average is the average parameter.

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FIR LPF

\* passFreq=50Hz

\* stopFreq=100Hz

\* 带外衰减40db

\* 采样频率1000Hz

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double Fs = 1000;

int freqLines = 500; //set frequency resolution here

double df = Fs / 2 / freqLines;

double[] coe = new double[] { 0.00194150175769345, -0.00169793575520208, -0.00426956799325628, -0.00829350380976647, -0.0132303432212115, -0.0181544335557185, -0.0217585679131809, -0.0225134101549763, -0.0189478446411358, -0.00995128109977753, 0.00489205937905203, 0.0250974062726473, 0.0492006450550267, 0.0748863114260257, 0.0992878499995790, 0.119424371366942, 0.132695897659600, 0.137328720011526, 0.132695897659600, 0.119424371366942, 0.0992878499995790, 0.0748863114260257, 0.0492006450550267, 0.0250974062726473, 0.00489205937905203, -0.00995128109977753, -0.0189478446411358, -0.0225134101549763, -0.0217585679131809, -0.0181544335557185, -0.0132303432212115 - 0.00829350380976647, -0.00426956799325628, -0.00169793575520208, 0.00194150175769345 };

double[] noiseInput = new double[1000];

double[] noiseAddon = new double[1000];

double[] FIROutput = new double[1000];

var fir = new FIRFilter();

fir.Coefficients = coe;

double[] analysisInWav = new double[freqLines \* 2];

double[] analysisOutWav = new double[freqLines \* 2];

double[] bodeMag = new double[freqLines];

double[] bodePhase = new double[freqLines];

double[] coherent = new double[freqLines];

int numOfAverage = 10;

FrequencyResponseFunction FRFAnalysis = new FrequencyResponseFunction();

FRFAnalysis.Average.Mode = AverageMode.RMS;

FRFAnalysis.Average.Number = numOfAverage;

FRFAnalysis.ResetAveraging = true;

for (int i = 0; i < numOfAverage; i++)

{

JY.DSP.Fundamental.Generation.UniformWhiteNoise(ref noiseInput, 0.01);

JY.DSP.Fundamental.Generation.UniformWhiteNoise(ref noiseAddon, 0.0001);

fir.Filter(noiseInput, ref FIROutput);

ArrayCalculation.Add(FIROutput, noiseAddon, ref FIROutput);

if (i == 0) FRFAnalysis.Reset();

FRFAnalysis.Analyze(noiseInput, FIROutput);

bodeMag = FRFAnalysis.GetMagenitude(true);

bodePhase = FRFAnalysis.GetPhase(true);

coherent = FRFAnalysis.GetCoherente();

}

bool averageDone = FRFAnalysis.AveragingDone;

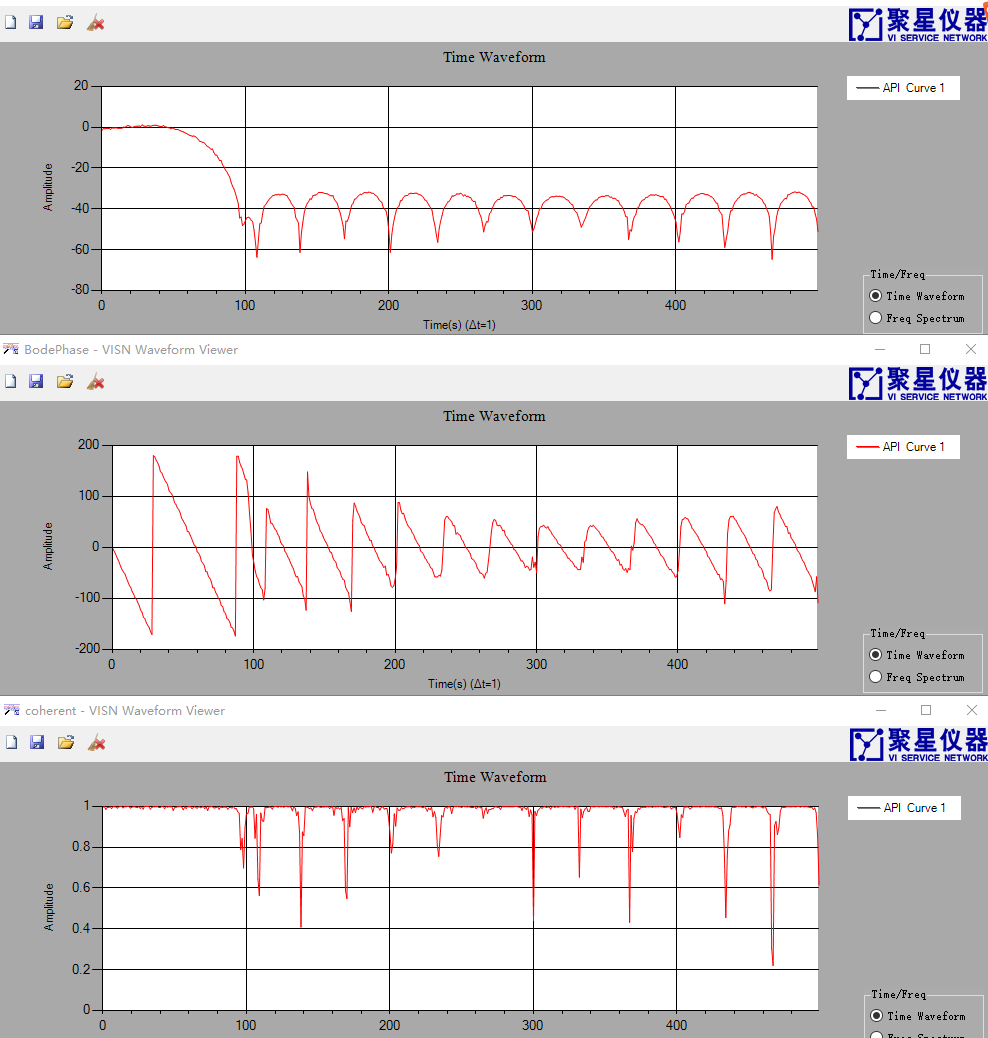
Console.WriteLine("averageDone:{0}", averageDone.ToString());

AnalogWaveformChart.Plot("BodeMagnitude", 0, 1 / df, bodeMag);

AnalogWaveformChart.Plot("BodePhase", 0, 1 / df, bodePhase);

AnalogWaveformChart.Plot("coherent", 0, 1 / df, coherent);

Results:



Pic 4.1 System's amplitude-frequency response, phase-frequency response, and correlations

## HarmonicAnalysis

### HarmonicAnalysis Introduction

Harmonic analysis includes ToneAnalysis、THDAnalysis、SINADAnalysis.

①ToneAnalysis: Calculate the amplitude, frequency and phase of the sine wave.

②THDAnalysis: Analysis of harmonic distortion.

③SINADAnalysis: Analysis of the ratio.

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tone分析 计算信号基波的幅度、相位、频率

\* 基波频率为20

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

var samplingRate = 1000;

var signalFrequency = 20;

var noiseLevel = 0.2;

var noise = new double[1000];

var signal = new double[1000];

JY.DSP.Fundamental.Generation.SineWave(ref signal, 4, 50, (double)signalFrequency, (double)samplingRate);

ArrayCalculation.AddOffset(ref signal, 2);

var dt = 1 / (double)samplingRate;

JY.DSP.Fundamental.Generation.UniformWhiteNoise(ref noise, noiseLevel);

ArrayCalculation.Add(signal, noise, ref signal);

AnalogWaveformChart.Plot("raw signal", signal);

double fundamentalFreq;

double phase;

double amplitude;

HarmonicAnalysis.ToneAnalysis(signal, dt, out fundamentalFreq,out amplitude,out phase);

Console.WriteLine("fundamentalFrequency={0}", fundamentalFreq);

Console.WriteLine("amplitude={0}", amplitude);

Console.WriteLine("phase={0}", phase);

**Results:**

fundamentalFreq=19.999813199835433

amplitude=3.9963377884507003

phase=50.079892166637848

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* THD分析

\* 基波频率为60

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

var samplingRate = 1000;

var signalFrequency = 60;

var noiseLevel = 0.2;

var noise = new double[1000];

var signal = new double[1000];

JY.DSP.Fundamental.Generation.SineWave(ref signal, 1, 50, (double)signalFrequency, (double)samplingRate);

ArrayCalculation.AddOffset(ref signal, 2);

var dt = 1 / (double)samplingRate;

JY.DSP.Fundamental.Generation.UniformWhiteNoise(ref noise, noiseLevel);

ArrayCalculation.Add(signal, noise, ref signal);

AnalogWaveformChart.Plot("raw signal", signal);

double fundamentalFreq;

double THD;

double[] componentsLevel = new double[0];

HarmonicAnalysis.THDAnalysis(signal, dt, out fundamentalFreq, out THD, ref componentsLevel);

Console.WriteLine("fundamentalFrequency={0}", fundamentalFreq);

Console.WriteLine("THD={0}", THD);

AnalogWaveformChart.Plot("Analysis", componentsLevel);

**Results:**

fundamentalFreq=60.0043143232529

THD=0.0362533770383877

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* SINAD分析

\* 基波频率为60

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

var samplingRate = 1000;

var signalFrequency = 60;

var noiseLevel = 0.2;

var noise = new double[1000];

var signal = new double[1000];

JY.DSP.Fundamental.Generation.SineWave(ref signal, 1, 50, (double)signalFrequency, (double)samplingRate);

ArrayCalculation.AddOffset(ref signal, 2);

var dt = 1 / (double)samplingRate;

JY.DSP.Fundamental.Generation.UniformWhiteNoise(ref noise, noiseLevel);

ArrayCalculation.Add(signal, noise, ref signal);

AnalogWaveformChart.Plot("raw signal", signal);

double fundamentalFreq;

double SINAD;

double[] componentsLevel = new double[0];

HarmonicAnalysis.SINADAnalysis( signal, dt, out fundamentalFreq, out SINAD, ref componentsLevel);

Console.WriteLine("fundamentalFrequency={0}", fundamentalFreq);

Console.WriteLine("SINAD={0}", SINAD);

AnalogWaveformChart.Plot("Analysis", componentsLevel);

**Results:**

fundamentalFreq=59.996122423727464

SINAD=1.0601596233046509

## SquarewaveMeasurements

### SquarewaveMeasurements Introduction

Square wave measurements include amplitude analysis, period analysis, phase analysis, level histogram, and duty cycle histogram.

①Amplitude analysis: void AmplitudeAnalysis(double[] waveform, out double highLevel, out double lowLevel)

②Cycle analysis: void PeriodAnalysis(double[] waveform, out double period, out double dutyCycleAvg,out double pulseCount, out double pulseMaxWidth, out double pulseMinWidth)

③Phase analysis: void PhaseAnalysis(double[] waveform, double[] waveformRef,out double phase)

④Level histogram: bool GetLevelHistogram(ref double[] histogramX, ref double[] histogramY, Histogram histogramConfig)

⑤Duty cycle histogram: bool GetDutyCycleHistogram(ref double[] histogramX,ref double[] histogramY,Histogram histogramConfig)

⑥Rise and fall time:TimeAnalysis(double[] waveform, out double risingSamples, out double fallingSamples);

Waveform is a square wave signal, waveformRef is a reference square wave signal, highLevel is high level, lowLevel is low level, period is period (unit is sampling point), dutyCycleAvg is duty ratio (0~1), pulseCount bit pulse Number, pulseMaxWidth is the maximum pulse width (in sample points), pulseMinWidth is the minimum pulse width (in sample points), phase is waveform relative to waveformRef, and risingSamples is rising.

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 产生频率为20Hz，采样率为1000Hz，幅值为2V，占空比为50%的方波

\* 测量方波的高电平、低电平、周期、相位

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double[] waveformTmp = new double[1000];

double[] waveformRef = new double[1000];

JY.DSP.Fundamental.Generation.SquareWave(ref waveformTmp, 2, 50, 20, 1000);

JY.DSP.Fundamental.Generation.SquareWave(ref waveformRef, 2, 50, 20, 1000);

double highLevel;

double lowLevel;

double period;

double dutycycle;

double phase;

double pulseCount;

double pulseMaxLength;

double pulseMinLength;

SquarewaveMeasurements.AmplitudeAnalysis(waveformTmp, out highLevel, out lowLevel);

SquarewaveMeasurements.PeriodAnalysis(waveformTmp, out period, out dutycycle,

out pulseCount, out pulseMaxLength, out pulseMinLength);

SquarewaveMeasurements.PhaseAnalysis(waveformTmp, waveformRef, out phase);

double[] levelHistogramX = new double[0];

double[] levelHistogramY = new double[0];

SquarewaveMeasurements.GetLevelHistogram(waveformTmp,ref levelHistogramX, ref levelHistogramY, new SquarewaveMeasurements.Histogram());

for (int j = 0; j < levelHistogramX.Length; j++)

{

Console.WriteLine("{0}:{1}", levelHistogramX[j], levelHistogramY[j]);

}

Console.WriteLine("highLevel:{0}", highLevel);

Console.WriteLine("lowLevel:{0}", lowLevel);

Console.WriteLine("period:{0}", period);

Console.WriteLine("phase:{0}", phase);

**Results:**

highLevel=2

lowLevel=-2

period=50

phase=0

## ThirdOctaveAnalysis

### ThirdOctaveAnalysis Introduction

1/3 octave analysis

### Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 产生50Hz的正弦信号

\* 做1/3倍频程分析

\* 时间平均方式为Fast

\* 加权滤波器的类型为A加权

\* 中心频率为50即第4频段（从0开始）的能量最高

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double duration = 2; //seconds;

double sampleRate = 51200; //you cannot change this

int dataLength = (int)(duration \* sampleRate);

double[] sin1 = new double[dataLength];

JY.DSP.Fundamental.Generation.SineWave(ref sin1, 1.414, 0, 50, sampleRate);

ThirdOctaveAnalysis analysis = new ThirdOctaveAnalysis();

analysis.AverageMode = Conditioning.TimeAveragingMode.Fast;

analysis.WeightingFilterType = WeightingType.AWeighting;

var result = analysis.Analyze(sin1, sampleRate);

double[] octaveLevels = new double[result.ThirdOctaveLevels.Length];

for (int i = 0; i < octaveLevels.Length; i++)

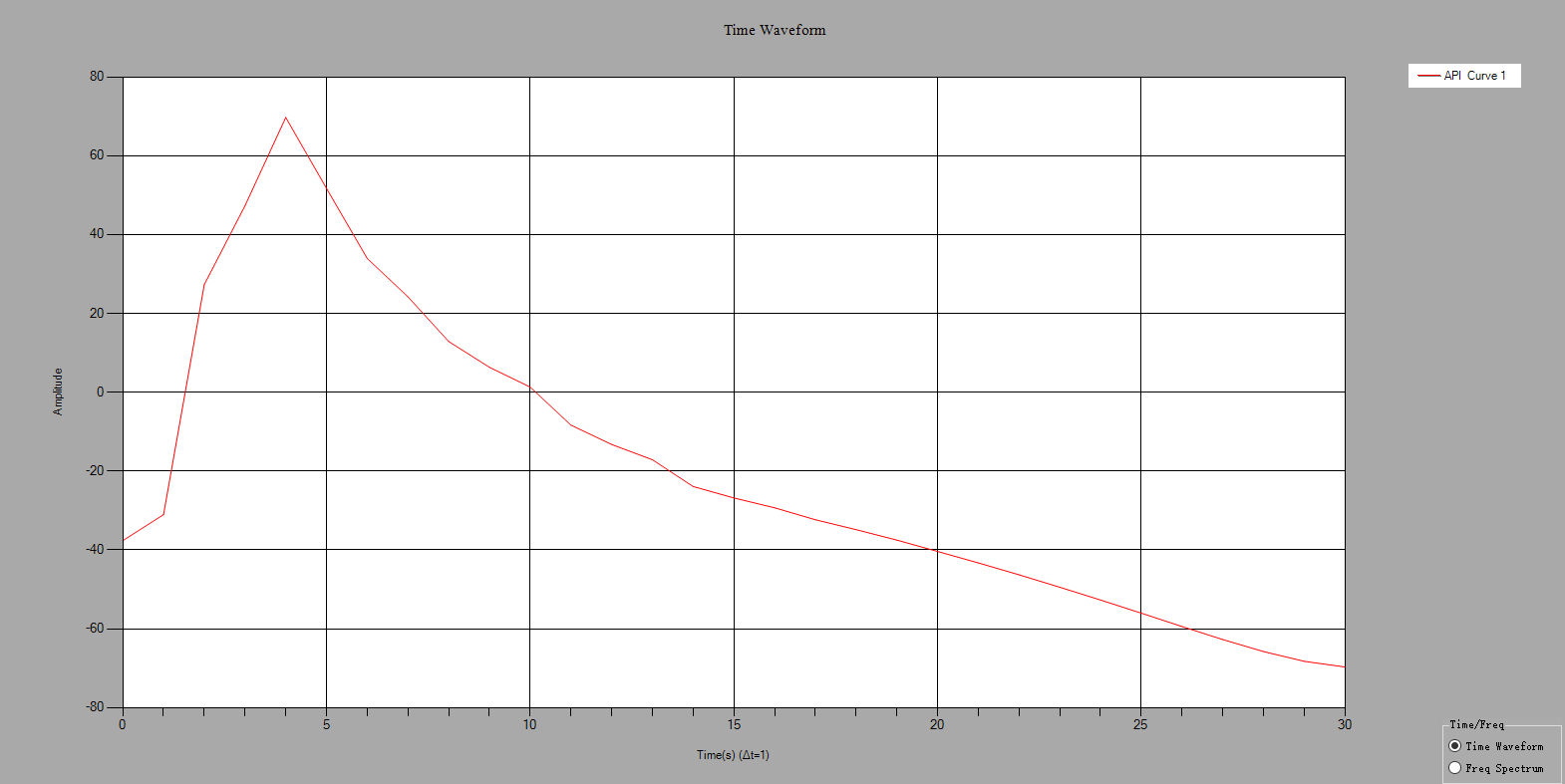
{

octaveLevels[i] = 20 \* Math.Log10(result.ThirdOctaveLevels[i]) + 100;

}

AnalogWaveformChart.Plot("Mode=Fast and WeightingFilterType=AWeighting", octaveLevels);

**Results:**



Pic 4.2 1/3 octave analysis

# Spectrum

## Spectrum Introduction

Fast Fourier transform, which can output amplitude spectrum and density spectrum, window type can be selected, support real/complex input, output unit has V2/V/dBV/dbmV/dbuV/W/dBW/dBm, resistance value defaults to 50 ohm .

**Note: The real signal is the FFT output line number is half of the input signal length, and the complex signal is the FFT output line number is the input signal length.**

## Example

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 产生有2V的直流分量10Hz，1Vpp的正弦波，并加上噪声

\* 实数FFT，不加窗，不平均，输出频谱单位为V

\* 验证峰值的有效值是不是2V和1/sqrt（2）=0.707V

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length = 1000;

double sampleRate = 1000;

double[] sin = new double[Length];

double[] noise = new double[Length];

double[] spec = new double[Length/2];

Generation.SineWave(ref sin, 1, 0, 10, sampleRate);

Generation.UniformWhiteNoise(ref noise, 0.1);

ArrayCalculation.Add(sin, noise, ref sin);

ArrayCalculation.AddOffset(ref sin, 2);

SpectrumTask \_task = new SpectrumTask();

\_task.InputDataType = InputDataType.Real;

\_task.SampleRate = sampleRate;

\_task.WindowType = FFTWindowType.None;

\_task.Average.Mode = SpectrumAverageMode.NoAveraging;

\_task.Average.WeightingType = SpectrumWeightingType.LinearMoving ;

\_task.Average.Size = 10;

\_task.Output.NumberOfLines = Length/2;

\_task.Unit.Type = SpectrumOutputUnit.V;

\_task.Unit.Impedance = 50;

\_task.Unit.IsPSD = false ;

// \_task.Commit();

\_task.GetSpectrum(sin, ref spec);

double df = \_task.SpectralInfomation.FreqDelta;

double count = \_task.SpectralInfomation.FFTCount;

double fftSize = \_task.SpectralInfomation.FFTSize;

double fStart = \_task.SpectralInfomation.FreqStart;

// VISN\_Plot.PlotData("Spectrum1",fStart,df,spec);

AnalogWaveformChart.Plot("Spectrum1",fStart, df, spec);

Peak[] a = new Peak[100];

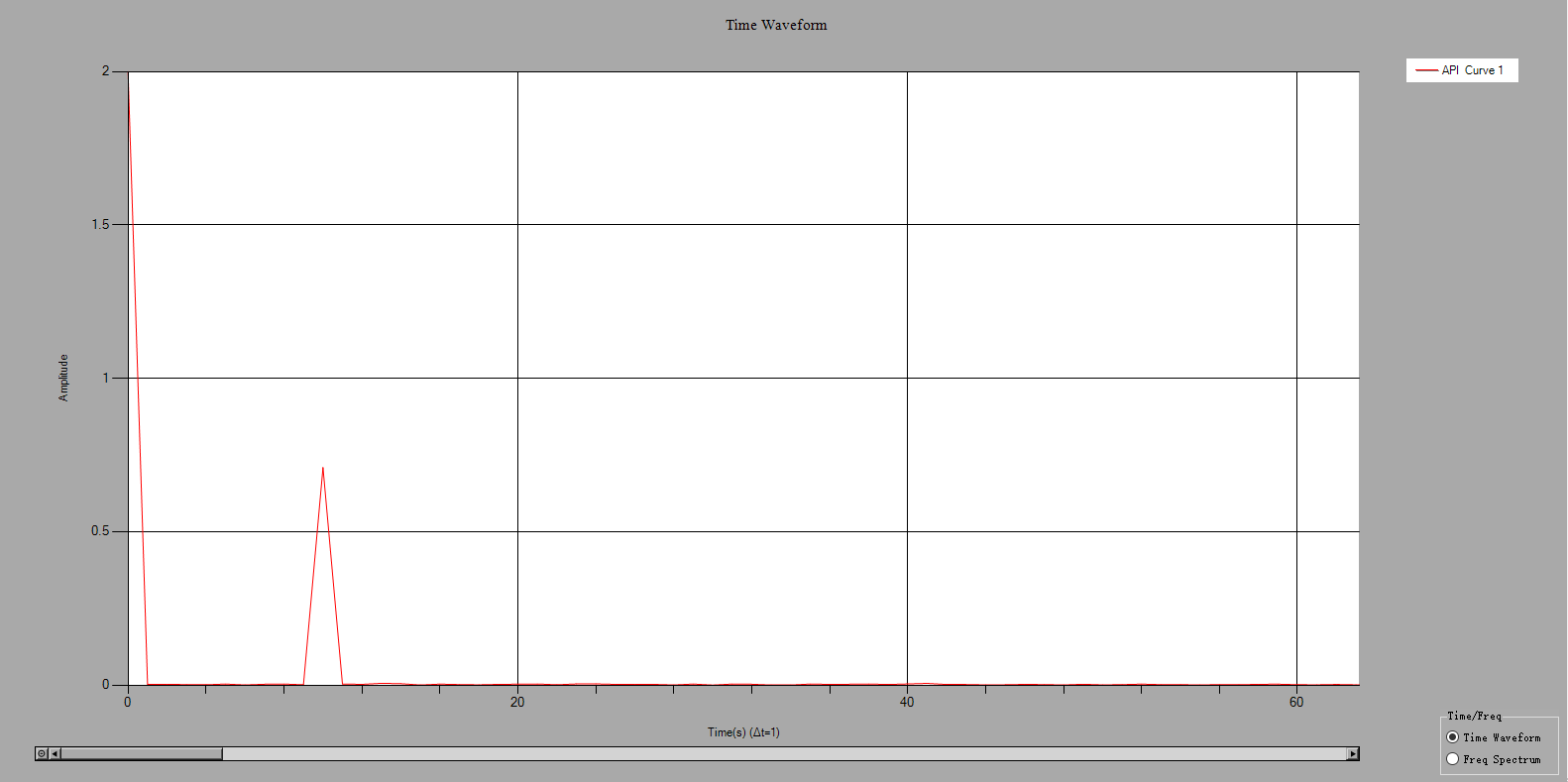
a=\_task.FindPeak(spec, 0.6, 0, 150);

Assert.IsTrue(Math.Abs((2 - a[0].PeakValue) / 2)< 0.01);//直流分量的幅值

Assert.IsTrue(Math.Abs((1/Math.Sqrt(2) - a[1].PeakValue)/ (1 / Math.Sqrt(2))) < 0.01); //正弦信号的有效值

Assert.IsTrue(Math.Abs((10 - a[1].PeakFrequency)/10) < 0.01);//正弦信号的频率

**Results:**



Pic 5.1 Real number FFT

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 产生复数信号，频率为10Hz，直流分量为2V，幅值1Vpp

\* 复数FFT，不加窗，不平均，输出频谱单位为V

\* 验证峰值的有效值

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Length = 1000;

double sampleRate = 1000;

double[] sin = new double[Length];

double[] cos = new double[Length];

double[] noise = new double[Length];

double[] spec = new double[Length ];

Generation.SineWave(ref sin, 1, 0, 50, sampleRate);

Generation.SineWave(ref cos, 1, 90, 50, sampleRate);

Generation.UniformWhiteNoise(ref noise, 0.05);

ArrayCalculation.Add(sin, noise, ref sin);

ArrayCalculation.Add(cos, noise, ref cos);

ArrayCalculation.AddOffset(ref sin, 2);

ArrayCalculation.AddOffset(ref cos, 2);

var signal = new Complex[Length];

for (int i = 0; i < Length; i++)

{

signal[i] = new Complex(cos[i], sin[i]);

}

SpectrumTask \_task = new SpectrumTask();

\_task.InputDataType = InputDataType.Complex;

\_task.SampleRate = sampleRate;

\_task.WindowType = FFTWindowType.None;

\_task.Average.Mode = SpectrumAverageMode.NoAveraging;

\_task.Average.WeightingType = SpectrumWeightingType.LinearMoving;

\_task.Average.Size = 10;

\_task.Output.NumberOfLines = Length;

\_task.Unit.Type = SpectrumOutputUnit.V;

\_task.Unit.Impedance = 50;

\_task.Unit.IsPSD = false;

\_task.GetSpectrum(signal, ref spec);

double df = \_task.SpectralInfomation.FreqDelta;

double count = \_task.SpectralInfomation.FFTCount;

double fftSize = \_task.SpectralInfomation.FFTSize;

double fStart = \_task.SpectralInfomation.FreqStart;

AnalogWaveformChart.Plot("Spectrum3", fStart, df, spec);

double f0Value = Math.Sqrt (8) ;

double f10Value = 1/Math.Sqrt(2);

Peak[] a = new Peak[100];

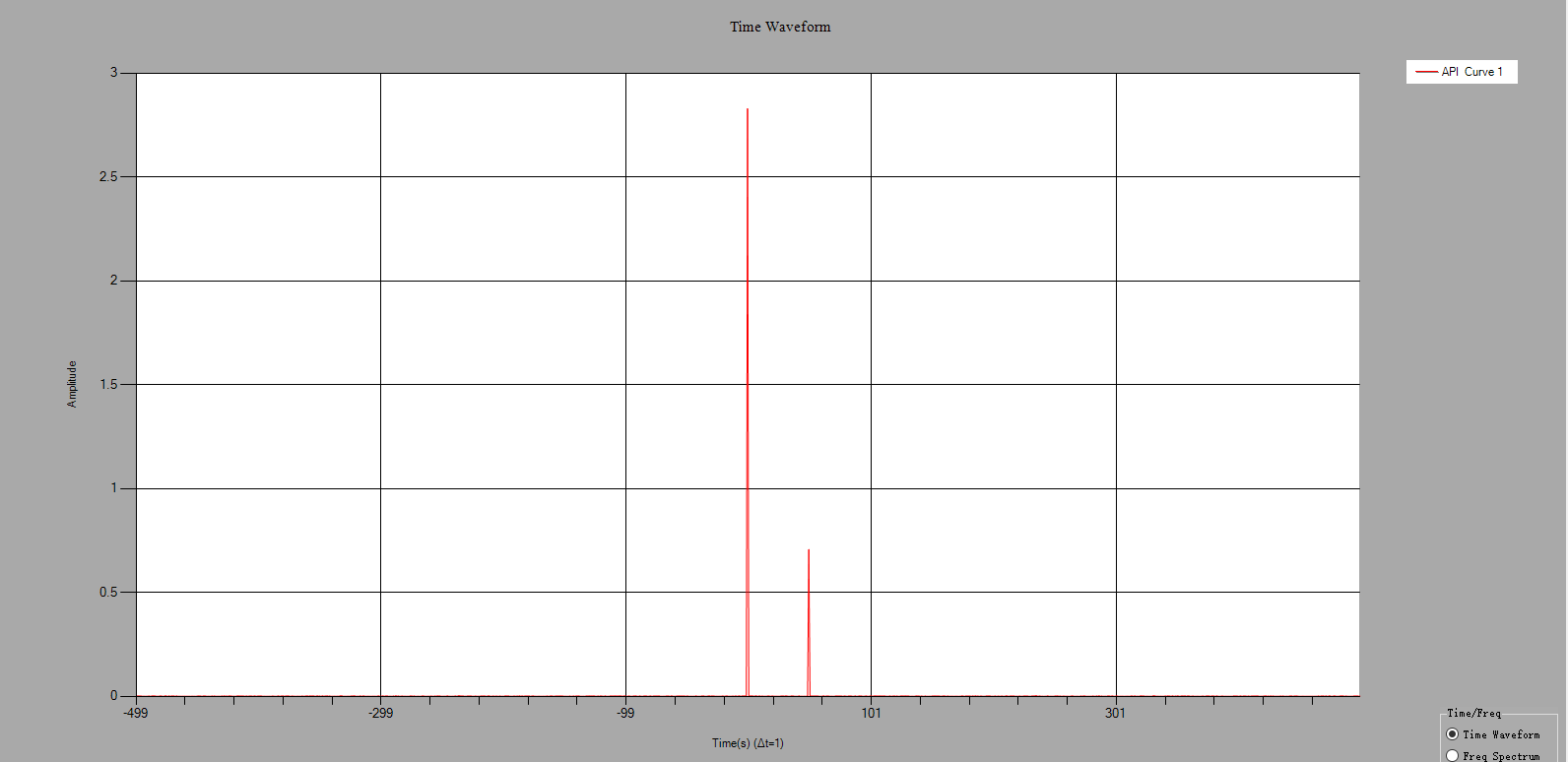
a = \_task.FindPeak(spec, 0.4, -15, 150);

Assert.IsTrue(Math.Abs((f0Value - a[0].PeakValue) / f0Value) < 0.01);//直流分量的幅值

Assert.IsTrue(Math.Abs((f10Value - a[1].PeakValue) / f10Value) < 0.01); //正弦信号的有效值

Assert.IsTrue(Math.Abs((10 - a[1].PeakFrequency) / 10) < 0.01);//正弦信号的频率

**Results:**



Pic 5.2 Plural FFT

# LicenseBase

When the engineer uses the DSP toolkit to complete the secondary development and does not want to activate it again on the client's machine, use the following code:

LicenseBase.Validate("ComputerID", "ActiveCode", "ActiveTime");

computerID:

activeCode:

activeTime: Fully activated，activeTime=””, Temporarily activate activeTime as seconds