Signal Generator

Generated by Doxygen 1.8.0

Sat Mar 24 2012 01:17:35

Contents

1 Data Structure Documentation

1.1 SignalGenerator Class Reference

Public Member Functions

- std::vector< double > * GetSignal ()
- std::vector< double > * GetSignalX ()
- void GenerateSignal (void)
- void SetAmplitude (const double &amplitude)
- void SetDecayConstant (const double &decay)
- void SetDelay (const double &delay)
- void SetFileName (const std::string &fileName)
- void SetFlattop (const double &flattop)
- void SetNoise (const bool &noise)
- void SetNoiseAmplitude (const double &amplitude)
- void SetPeriod (const double &period)
- void SetRisetime (const double &risetime)
- void SetSigma (const double &sigma)
- void SetSignalLength (const unsigned int &sigLength)
- void SetSignalResolution (const double &sigRes)
- void SetSignalType (const std::string &sigType)
- void SetSignalType (const std::string &sigTypeA, const std::string &sigTypeB)

1.1.1 Member Function Documentation

1.1.1.1 void SignalGenerator::GenerateSignal (void)

Generate a signal

```
                                        {
if(type_ == "custom") {
   CustomFunction();
}else if(typeA_ != "") {
```

```
      FillVector(signal_, type_);
     FillVector(signalA_, typeA_);
      CompositeFunction();
   }else {
      FillVector(signal_, type_);
   }

   if(hasNoise_)
      for(unsigned int i = 0; i < signal_.size(); i++)
         signal_[i] += noiseAmp_*Noise();
}
```

1.1.1.2 std::vector<double>* SignalGenerator::GetSignal ( ) [inline]

Returns the values of the generated signal

If a custom function is specified this will return the y values

Remember: The size of the vector will be length * resolution

```
{return &signal_;};
```

1.1.1.3 std::vector<double>* SignalGenerator::GetSignalX ( ) [inline]

Returns the x values of a custom function.

```
{return &signalA_;};
```

1.1.1.4 void SignalGenerator::SetAmplitude (const double & *amplitude*) [inline]

Set the amplitude of the generated signal

```
{amp_ = amplitude;};
```

1.1.1.5 void SignalGenerator::SetDecayConstant (const double & *decay*) [inline]

Set the decay constant for an exponential decay. (not implemented)

```
{decay_ = decay;};
```

1.1.1.6 void SignalGenerator::SetDelay (const double & *delay*) [inline]

Set the delay for the signal

```
{mu_ = delay;};
```

1.1.1.7 void SignalGenerator::SetFileName (const std::string & *fileName*) [inline]

Set the file name to read the custom function.

The file should contain a series of xy pairs

```
{fileName_ = fileName;};
```

1.1.1.8 void SignalGenerator::SetFlattop (const double & *flattop*) [inline]

Set the flattop for the trapezoidal signal (not implemented)

```
{flattop_ = flattop;};
```

1.1.1.9 void SignalGenerator::SetNoise (const bool & *noise*) [inline]

Set to true to give the signal some white noise

```
{hasNoise_ = noise;}
```

1.1.1.10 void SignalGenerator::SetNoiseAmplitude (const double & *amplitude*) [inline]

Set the amplitude of the noise

```
{noiseAmp_ = amplitude;};
```

1.1.1.11 void SignalGenerator::SetPeriod (const double & *period*) [inline]

Set the periodicity of the signal (if periodic).

```
{period_ = period;};
```

1.1.1.12 void SignalGenerator::SetRisetime (const double & *risetime*) [inline]

Set the risetime for a trapezoidal signal (not implemented)

```
{risetime_ = risetime;};
```

1.1.1.13 void SignalGenerator::SetSigma (const double & *sigma*) [inline]

Set the sigma for the gaussian signal

```
{sigma_ = sigma;};
```

1.1.1.14 void SignalGenerator::SetSignalLength (const unsigned int & *sigLength*) [inline]

Set the length of the generated signal

```
{length_ = sigLength;}
```

1.1.1.15 void SignalGenerator::SetSignalResolution (const double & *sigRes*) [inline]

Set the resolution for the generated signal

```
{res_ = sigRes;}
```

1.1.1.16 void SignalGenerator::SetSignalType (const std::string & *sigType*) [inline]

Set the type of signal to generate
Available types: sine, cosine, gaussian, sawtooth, square, triangle, custom, composite.

```
{type_ = sigType;}
```

1.1.1.17 void SignalGenerator::SetSignalType (const std::string & *sigTypeA*, const std::string & *sigTypeB*) [inline]

Set the types of signals to generate for a composite signal

```
    {type_ = sigTypeA; typeA_ = sigTypeB;};
```

The documentation for this class was generated from the following files:

- include/SignalGenerator.hpp
- src/SignalGenerator.cpp

2 File Documentation

2.1 include/SignalGenerator.hpp File Reference

```
#include <string>
#include <vector>
#include <cstdlib>
#include "time.h"
```

Data Structures

- class SignalGenerator

2.1.1 Detailed Description

Class for generating different kinds of signals; also allows for one to add random noise.

Author

S.V. Paulauskas

Date

15 March 2012

3 Example Documentation

3.1 pictures.dox

Example pictures

Red function - Signal without noise

Green Function - Signal with noise

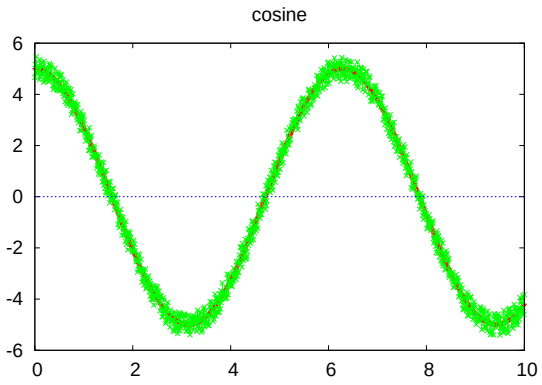

Figure 1: Cosine Function

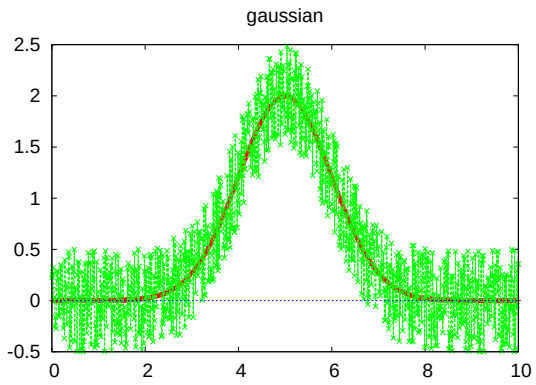

Figure 2: Gaussian Function

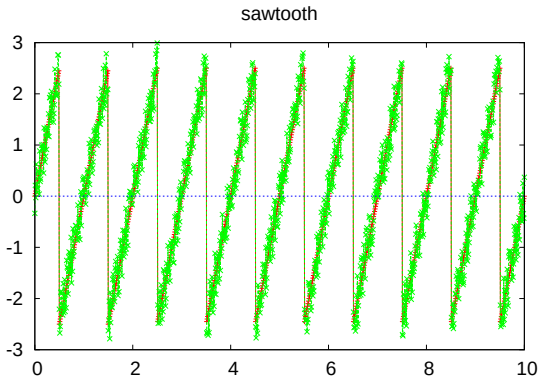

Figure 3: Sawtooth Function

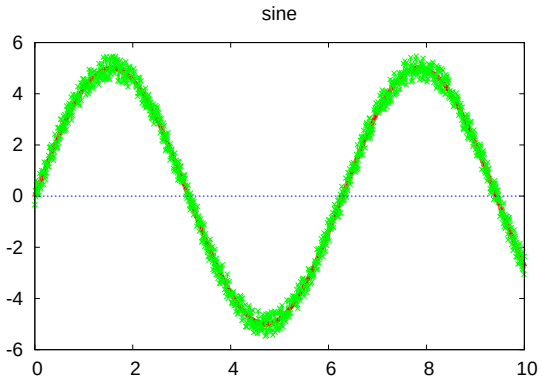

Figure 4: Sine Function

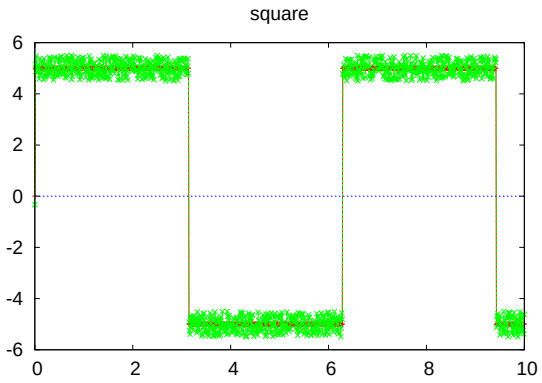

Figure 5: Square Function

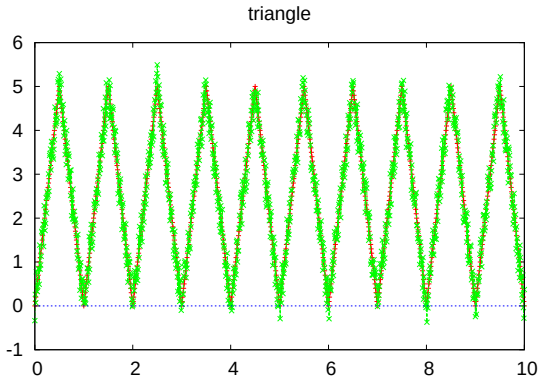

Figure 6: Triangle Function

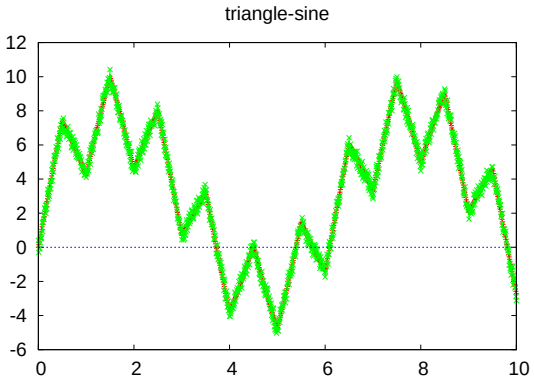

Figure 7: Triangle+Sine Function

3.2 sig.cpp

A simple example. The example generates a simple function, a compound function, and finally a custom function. It will then output these functions into separate data files.