

Wavelet Analysis Tool

1. Class Wavelet

Wavelet is the abstract base class for other WAT classes

```
typedef double wavereal;
```

```
class Wavelet
```

Public methods:

Constructors:

- Wavelet Wavelet()
- Wavelet Wavelet(Wavelet&)

Destructor:

- virtual void ~Wavelet()

2. Class WaveData

WaveData is the time-domain data container with corresponding methods.

```
class WaveData : public Wavelet
```

Public data members:

- int N - number of elements in data array,
- wavereal* data - time domain data array,
- wavereal Rate - data sampling rate,

Public methods:

Constructors:

- WaveData WaveData(int n)
- WaveData WaveData()
- WaveData WaveData(WaveData&)

Destructor:

- virtual void ~WaveData()

Accessors:

- void getStatistics(wavereal&, wavereal&)

Operators:

- WaveData& operator*=(const wavereal c)
- WaveData& operator+=(const wavereal c)
- WaveData& operator+=(const WaveData& a)
- WaveData& operator+=(const WaveData* a)

- WaveData& operator==(const wavereal c)
- WaveData& operator==(const WaveData* a)
- WaveData& operator==(const WaveData& a)
- WaveData& operator=(const WaveData& a)
- WaveData& operator=(const wavereal c)
- WaveData& operator=(const TSeries* ts)
- WaveData& operator=(const TSeries& ts)
- WaveData& operator=(const WaveData* a)

Mutators:

- void add(const WaveData&, int = 0, int = 0, int = 0)
- void sub(const WaveData& s, int = 0, int = 0, int = 0)
- void cpf(const WaveData&, int = 0, int = 0, int = 0)
- void FFT(int = 1)
- void Resample(const WaveData&, wavereal, int = 6)
- wavereal Stack(const WaveData&, int)
- wavereal Stack(const WaveData&, int, int)
- wavereal Stack(const WaveData&, wavereal)

Interfaces:

- virtual void Streamer(TBuffer& b)
- virtual void Dump(const char*, int = 0)
- void DumpBinary(const char*, int = 0)
- void DumpShort(const char*, int = 0)
- void DumpSpectrum(const char* fname, int option = 2)
- void ReadBinary(const char*)
- void ReadShort(const char*)

3. Class WaveRDC

WaveRDC is the class for compressed data structures and methods.

```
class WaveRDC : public WaveData
```

Public data members:

- int NL number of layers in compressed array dataz
- int NZ number of elements in compressed array dataz
- int optz current layer compression options
- unsigned int* dataz compressed data array

Private data members:

- int Nlsw number of 32-bits layer service words (lsw)
- int freebits free bits in the last word of current block
- int kl current layer encoding bit length for large integers

- int ks current layer encoding bit length for small integers
- int kbsw length of the Block Service Word
- short Shift current layer shift subtracted from original data
- short zero number that encodes 0;

Public methods:

Constructors:

- WaveRDC WaveRDC()
- WaveRDC WaveRDC(WaveRDC&)

Destructor:

- virtual void ~WaveRDC()

Accessors:

- void Dir(int v = 1)

Operators:

- WaveRDC& operator+=(const WaveRDC* a)
- WaveRDC& operator+=(const WaveRDC& a)
- WaveRDC& operator=(const WaveRDC* a)
- WaveRDC& operator=(const WaveRDC& a)

Mutators:

- void ResizeZ(int n)
- int Compress(const WaveData&, bool fast = false)
- int unCompress(WaveData&, int level = 1)

Interfaces:

- virtual int DumpZ(const char*, int = 0)

Algorithms:

Lossless Random Data Compression (RDC) - see reference.

References:

<http://www.phys.ufl.edu/LIGO/wavelet/compress.html>

4. Class WSeries

WSeries provides generic methods for data manipulation in wavelet domain.

```
class WSeries : public WaveData
```

Public methods:

Constructors:

- WSeries WSeries(int n)
- WSeries WSeries()
- WSeries WSeries(WSeries&)

Destructor:

- virtual void ~WSeries()

Operators:

- WSeries& operator*=(const wavereal)
- WSeries& operator+=(const WSeries&)
- WSeries& operator+=(const WSeries*)
- WSeries& operator+=(const wavereal)
- WSeries& operator-=(const WSeries&)
- WSeries& operator-=(const wavereal)
- WSeries& operator-=(const WSeries*)
- WSeries& operator=(const WSeries*)
- WSeries& operator=(const WSeries&)
- WSeries& operator=(const wavereal)

Mutators

- void Merge2(WSeries& wd2)

5. Class WaveDWT

WaveDWT is the base class for all Discrete Wavelet Transform wavlets.

class WaveDWT : public Wavelet

Public data members:

- WSeries* pWDC pointer to WSeries class containing data
- int nh number of filter coefficients
- int MaxLevel maximal level of decomposition
- int Level current level of decomposition
- int IsTree 0-diadic, 1-binary tree
- int border see constants definitions above

Public methods:

Constructors:

- WaveDWT(int m=1,int ist=0, int brd=1);

Destructor:

- virtual void ~WaveDWT()

Accessors:

- int getIdA(int, int)
- int getIdF(int, int)
- int getIdL(int, int)
- void getLayer(WaveData&, int)
- void getFreqLayer(WaveData&, int)
- virtual int getMaxLevel()

Mutator:

- virtual void decompose(int, int)

- virtual void reconstruct(int, int)
- virtual void t2w(const WaveData&, int = -1)
- virtual void t2w(int = 1)
- virtual void t2w(const TSeries&, int = -1)
- virtual void w2t(int = 1)
- virtual void w2t(TSeries&, int = -1)
- virtual void w2t(WaveData&, int = -1)
- virtual void Denoise(wavereal)
- void putFreqLayer(WaveData&, int)
- void putLayer(WaveData&, int)

Interfaces:

- int DumpWTAsBitmap(const char*, unsigned int = 0)

6. Class WaveletL

Class WaveletL implements lifting wavelets.

```
class WaveletL : public WaveDWT
```

Private data members:

- wavereal* hp pointer to array of predict coefficients.
- wavereal* hu pointer to array of update coefficients.

Public methods:

Constructors:

- WaveletL WaveletL(int m, int n, int ist, int brd = 1)
- WaveletL WaveletL(int n = 4, int ist = 0)
- WaveletL WaveletL(WaveletL&)

Destructor:

- virtual void ~WaveletL()

Accessors:

- virtual int getMaxLevel()

Mutators:

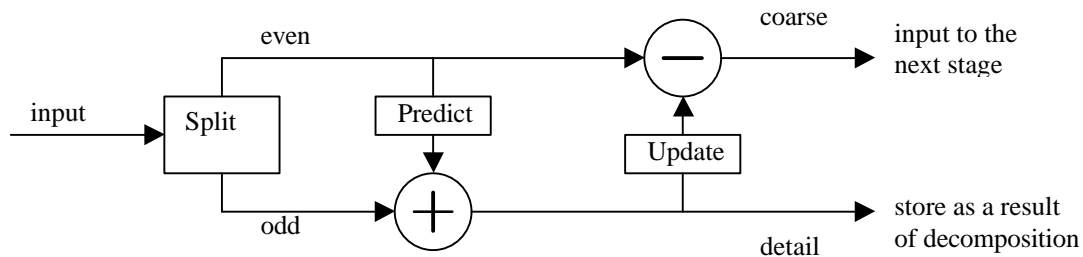
- virtual void decompose(int level, int layer)
- virtual void reconstruct(int level, int layer)

Protected methods:

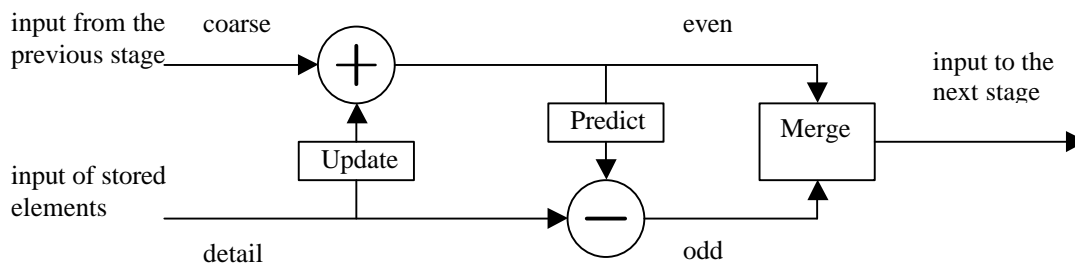
```
wavereal icoeff(const int n, const int i, const wavereal x)
```

Algorithms:

Each decomposition step may be illustrated by the following scheme:



The inverse transformation is implemented as reversed algorithm for decomposition and may be illustrated by the following scheme:



References:

Wim Sweldens, Peter Schroder “Building Your Own Wavelets at Home”

7. Class WaveletD

Class WaveletD implements Daubechies (order = 2..30) and Symlet (order=-2..-30) families of wavelets.

class WaveletD : public WaveDWT

Private data members:

- wavereal* h pointer to array of Daubechies filter coefficients

Public methods:

Constructors:

- WaveletD WaveletD(int order = 2, int tree = 0)
- WaveletD WaveletD(int n, int ord, int tree)
- WaveletD WaveletD(int n, int ord, int tree, int brd)
- WaveletD WaveletD(WaveletD&)

Destructor:

- virtual void ~WaveletD()

Mutators:

- virtual void decompose(int, int)
- virtual void reconstruct(int, int)

8. Class WaveletM

Class WaveletM implements Meyer wavelets (filter length 32).

```
class WaveletM : public WaveDWT
```

Public data members:

Constructors:

- WaveletM WaveletM(int = 0)
- WaveletM WaveletM(int, int)
- WaveletM WaveletM(WaveletM&)

Destructor:

- virtual void ~WaveletM()

Mutators:

- virtual void decompose(int, int)
- virtual void reconstruct(int, int)

9. Class WaveletI

Class WaveletI implements integer wavelets based on lifting transform. Input and output data for decompose() and reconstruction() are integer numbers.

```
class WaveletI : public WaveDWT
```

Private data members:

- wavereal* hp pointer to array of predict coefficients.
- wavereal* hu pointer to array of update coefficients.
- int boundaries

Public methods:

Constructors:

- WaveletI WaveletI(int m, int n, int ist, int bounds)
- WaveletI WaveletI(int n = 4, int ist = 0, int bounds = 0)
- WaveletI WaveletI(WaveletI&)

Destructor:

- virtual void ~WaveletI()

Accessors:

- virtual int getMaxLevel()

Mutators:

- virtual void decompose(int, int)
- virtual void reconstruct(int, int)

Private methods:

- void decompose2(int, int)
- void decompose4(int, int)
- void decompose6(int, int)
- void decompose8(int, int)
- void decompose_gc(int, int)
- wavereal icoeff(const int n, const int i, const wavereal x)
- void reconstruct2(int, int)
- void reconstruct4(int, int)
- void reconstruct6(int, int)
- void reconstruct8(int, int)
- void reconstruct_gc(int, int)
- waveint WINT(wavereal wrv)

10. Class WaveletG

Class WaveletG implements continuous wavelet transform that uses the family of Gaussian wavelets.

```
class WaveletG : public Wavelet
```

Public data members:

- WSeries* pWDC pointer to WSeries object with data.

Private data members:

- int scaleN
- wavereal scale0
- wavereal scale1
- TR_INFO histTrInfo table for source.
- TR_INFO shiftTrInfo table for shift.
- TR_INFO scaleTrInfo table for scale.
- func _TrFunc wavelet function.
- wavereal* _TrHist temporary pointer to data.
- TR_TABLE _TrTable transformation table information.

- `_wave wavenum` number of wavelet.
- `wavereal RelSquare` used relative square of analyzing wavelet.
- `wavereal bxdelta`
- `int halfwindow` half-width of the window

Public methods:

Constructors:

- `WaveletG WaveletG(int = 2)`
- `WaveletG WaveletG(WaveletG&)`

Destructor:

- `virtual void ~WaveletG()`

Mutators:

- `void t2w(WaveData& t, int _scaleN, wavereal _scale0, wavereal _scale1, int method = 0, int norm = 1, wavereal unused = 0.0001)`

Interfaces:

- `void DumpWTAsBitmap(const char* fn, int palette = 0)`

Private methods:

- `TR_INFO& buildHistTrInfo()`
- `TR_INFO& buildShiftTrInfo()`
- `int buildTrTableFast(wavereal scale, wavereal rate)`
- `TR_INFO& clearTrInfo(TR_INFO&)`
- `wavereal convTr(wavereal shift, wavereal scale)`
- `void convTr(wavereal* result, wavereal scale)`
- `void destroyTrTable()`
- `wavereal fastTr(int shiftj, wavereal scale)`
- `void fastTr(wavereal* result, wavereal scale)`
- `wavereal nativeTr(wavereal shift, wavereal scale)`
- `void nativeTr(wavereal* result, wavereal scale)`
- `void setTrHist(wavereal* h)`
- `void ultrafastTr(wavereal* result, wavereal scale)`
- `wavereal ultrafastTr(int shiftj, wavereal scale)`

11. Class LineFilter

The LineFilter class contains methods to track and remove quasi-monochromatic lines.

References:

<http://www.phys.ufl.edu/LIGO/LINE/index.html>

12. Lossy compression module

This module implements lossy data compression with the use of wavelet methods.

- `int Compress(WaveData &, int* &, int=0, int=0, const double=1., const double=1., int=10, int=10);`
- `int Compress(double [], int, int* &, int=0, int=0, const double=1., const double=1., int=10, int=10);`
- `int Compress(float [], int, int* &, int=0, int=0, const double=1., const double=1., int=10, int=10);`
- `int Compress(int [], int, int* &, int=0, int=0, const double=1., const double=1., int=10, int=10);`
- `int Compress(short [], int, int* &, int=0, int=0, const double=1., const double=1., int=10, int=10);`
- `int unCompress(int*, WaveData &);`
- `int unCompress(int*, double* &);`
- `int unCompress(int*, float* &);`
- `int unCompress(int*, int* &);`
- `int unCompress(int*, short* &);`

References:

<http://www.phys.ufl.edu/LIGO/wavelet/compress.html>

13. ReadFrFile module

This module allows access to data compressed by WAT lossy compression and stored in frame format files.

- `WaveData* ReadFrFile(double tlen, double tskip, char *cname, char *fname, bool seek=true);`

References:

<http://www.phys.ufl.edu/LIGO/wavelet/compress.html>