CONFIDENTIAL DOCUMENT

Report about Pwelch

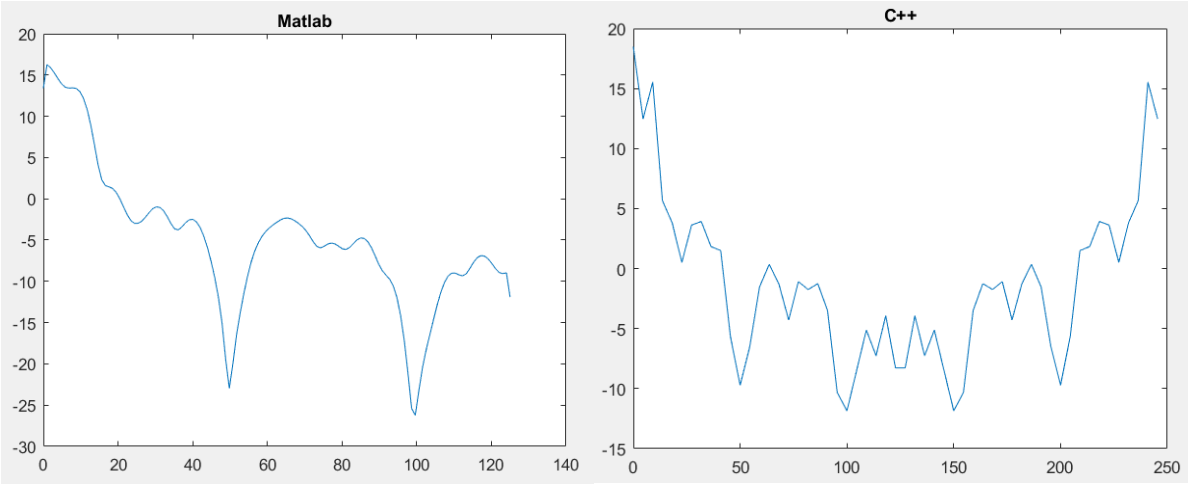
Report Names: Fanny Grosselin	Verification Name: Yohan Attal Date: Signature:
History of modifications: - *Creation 20/01/2017* : Fanny Grosselin Description of the changes in MBT_PWelchComputer and MBT_Fourier in order to have the same results than with the pwelch of Matlab.	

Introduction :

This document lists and describes the changes induces in MBT_PWelchComputer and MBT_Fourier in order to converge to the same result than with pwelch of Matlab.

1. Comparison of the psd with a signal with 250 datapoints.

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
initial MBT_PWelchComputer and initial MBT_Fourier



Size of frequency vector : 1x129 (from 0 to 125Hz) 1x55 (from 0 to 245,45Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
First change of MBT_PWelchComputer and initial MBT_Fourier

First change of MBT_PWelchComputer.cpp:

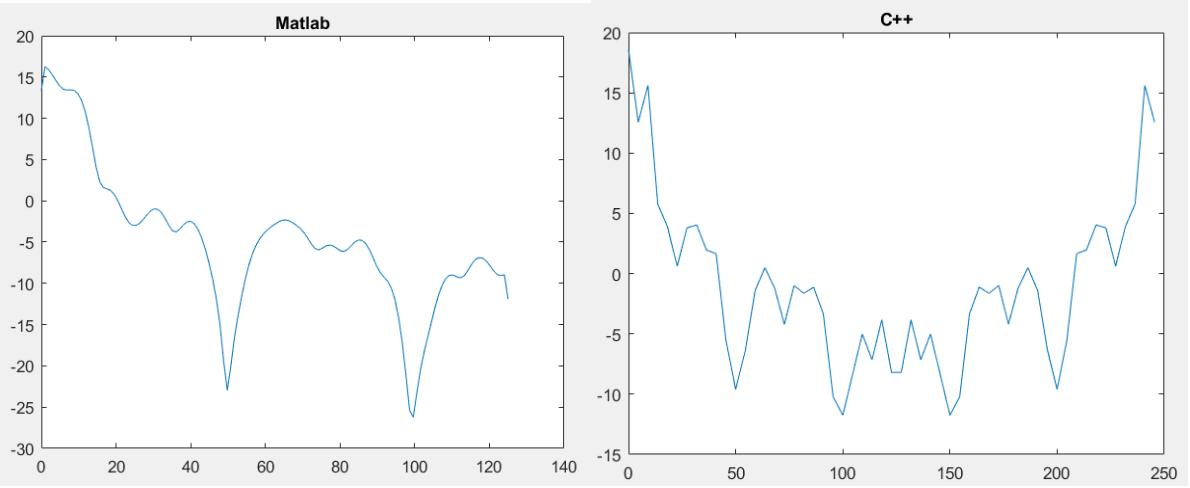
```
const float  PI_F=3.14159265358979f;
    switch (m_windowType) {
        case RECT:
            return 1;
            break;

        case HANN:
           //return (1.0 - cos(2.0 * PI_F * n / windowLength)) / 2.0;
            return (1.0 - cos(2.0 * PI_F * n / (windowLength-1))) / 2.0; // Fanny Grosselin
2017/01/10 (windowLength-1) instead of windowLength
            break;

        case HAMMING:
           //return 0.54 - 0.46 * cos(2.0 * PI_F * n / windowLength);
            return 0.54 - 0.46 * cos(2.0 * PI_F * n / (windowLength-1));   // Fanny Grosselin
2017/01/10 (windowLength-1) instead of windowLength
            break;

        default:
            break;
    }
```

Explainations:

w = hamming(L) returns an L-point symmetric Hamming window in the column vector w. L should be a positive integer. The coefficients of a Hamming window are computed from the following equation.

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), \quad 0 \le n \le N$$

The window length is L=N+1.

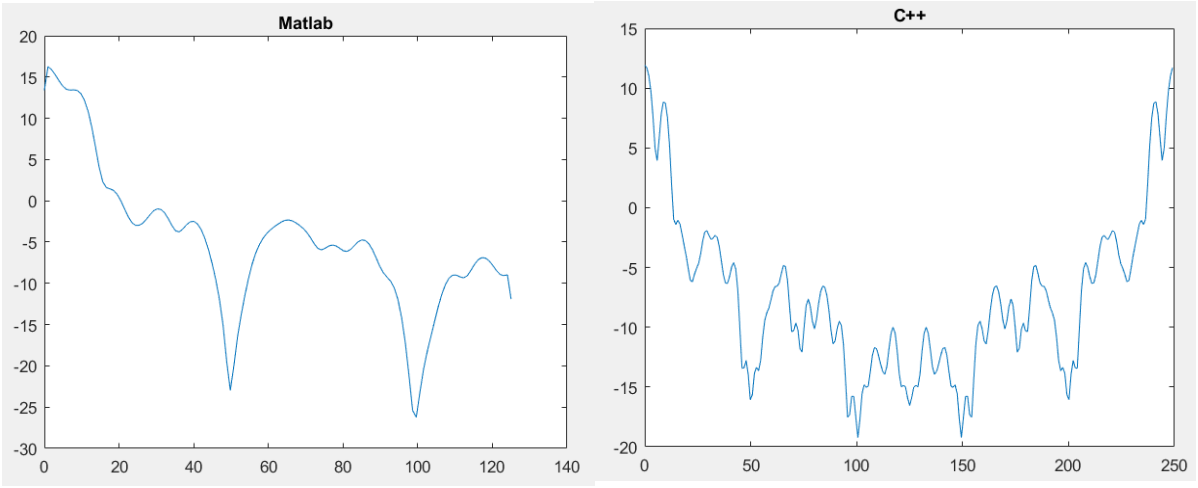


Size of frequency vector : 1x129 (from 0 to 125Hz) 1x55 (from 0 to 245,45Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Second change of MBT_PWelchComputer and initial MBT_Fourier

Second change of MBT_PWelchComputer.cpp:

```
Zero-padding of each segment.
```



Size of frequency vector : 1x129 (from 0 to 125Hz) 1x256 (from 0 to 245,45Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Second change of MBT_PWelchComputer and First change of MBT_Fourier

First change of MBT_Fourier.cpp:

```
Change ceil to floor in the computation of powerOfTwoLength in
MBT_FourierBluestein::bluesteinConvolutionParallel.
```

No change concerning the plot.

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Second change of MBT_PWelchComputer and Second change of MBT_Fourier

Second change of MBT_Fourier.cpp:

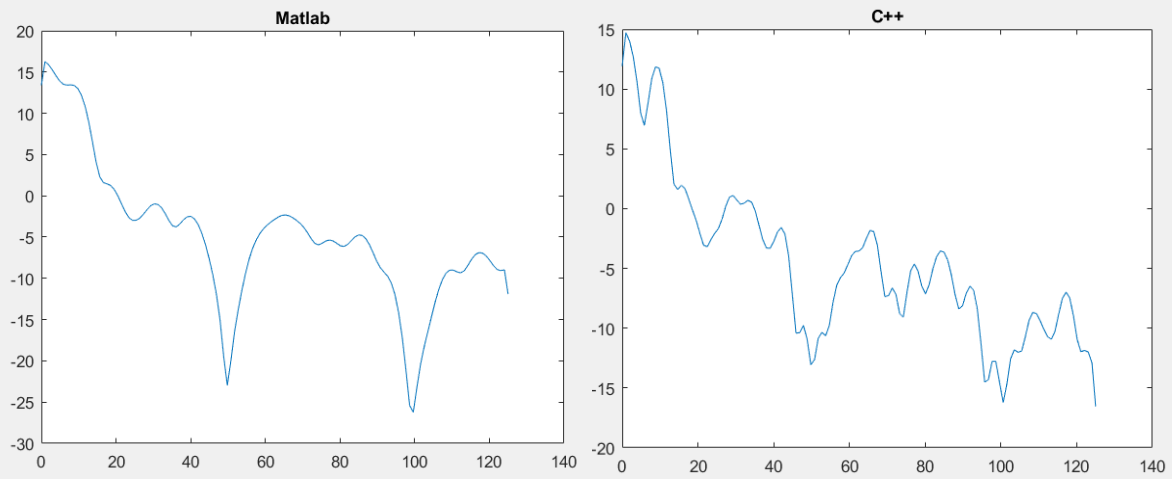
```
Zero-padding instead of padding by the mirror of the signal in
MBT_FourierBluestein::workerBluesteinConvolutionParallelB and
MBT_FourierBluestein::bluesteinConvolutionParallel
```

No change concerning the plot.

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Third change of MBT_PWelchComputer and Second change of MBT_Fourier

Third change of MBT_PWelchComputer.cpp:

```
Compute the one-sided spectrum.
```

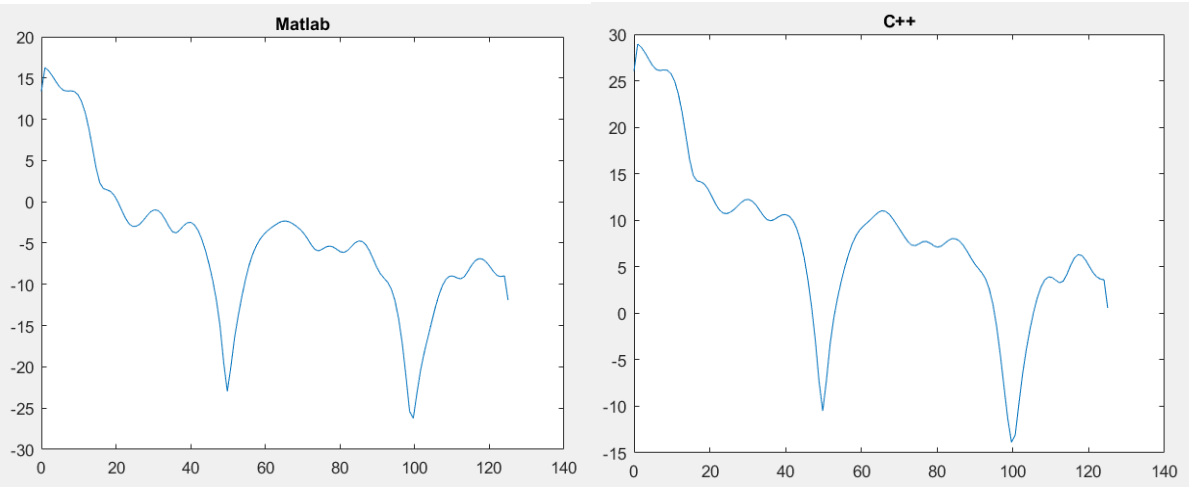


Same frequency vector : 1x129 (from 0 to 125Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Fourth change of MBT_PWelchComputer and Second change of MBT_Fourier

Fourth change of MBT_PWelchComputer.cpp:

```
Correct the complexSignal generation: computeWindow should be called with i and not window
because "n" of computeWindow should be in [0:windowLength-1] and not in [0:windowNumber].
```

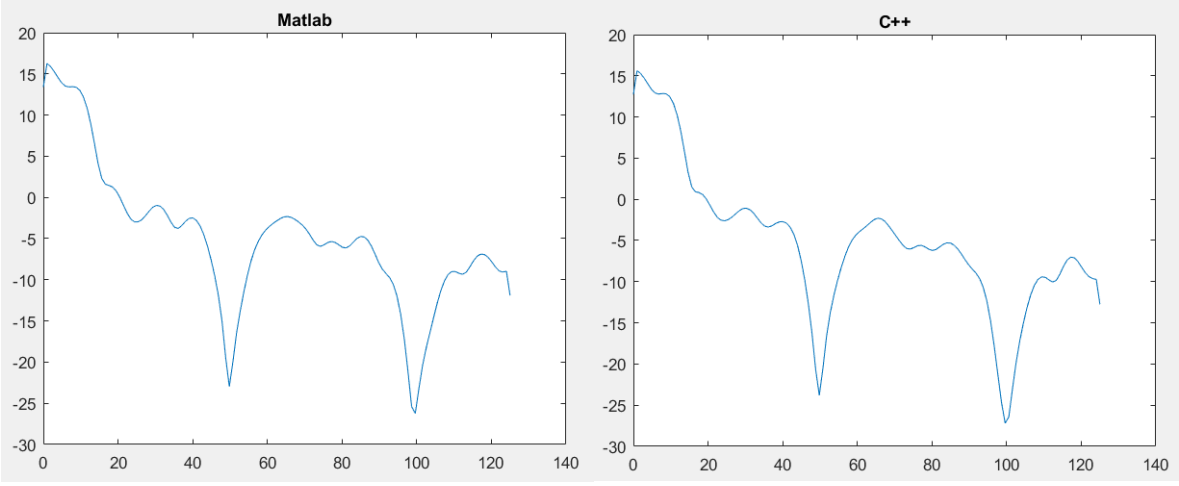


Same frequency vector : 1x129 (from 0 to 125Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Fifth change of MBT_PWelchComputer and Second change of MBT_Fourier

Fifth change of MBT_PWelchComputer.cpp:

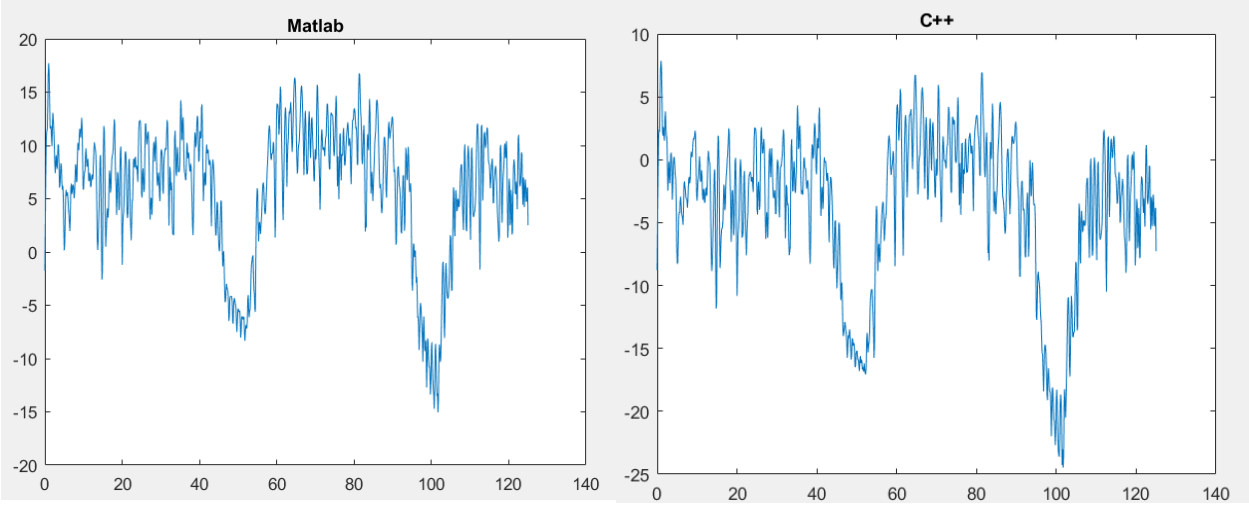
```
Compensate the power of the Hamming window by dividing the power by the power of the window.
```



Same frequency vector : 1x129 (from 0 to 125Hz)

2. Comparison of the psd with a signal with 5000 datapoints.

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Fifth change of MBT_PWelchComputer and Second change of MBT_Fourier

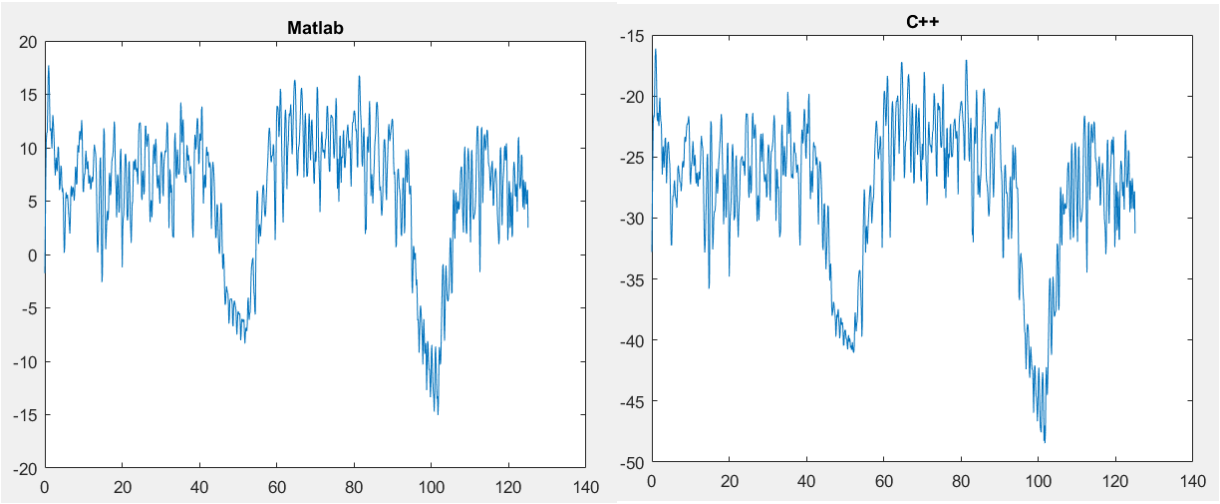


Same frequency vector : 1x1025 (from 0 to 125Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Sixth change of MBT_PWelchComputer and Second change of MBT_Fourier

Sixth change of MBT_PWelchComputer.cpp:

```
Divide the power by samplingRate like in Matlab.
```

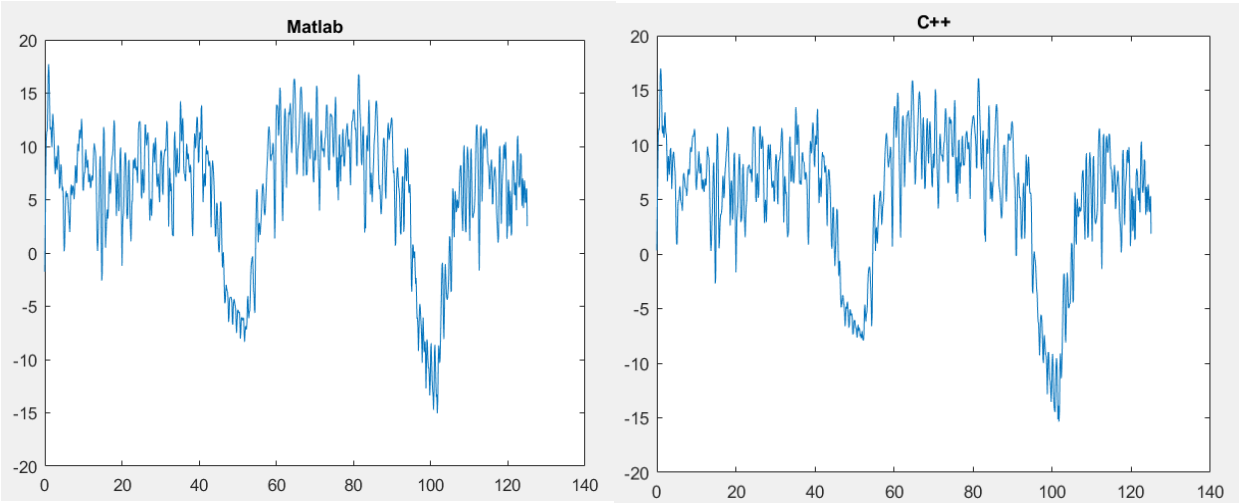


Same frequency vector : 1x1025 (from 0 to 125Hz)

[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Seventh change of MBT_PWelchComputer and Second change of MBT_Fourier

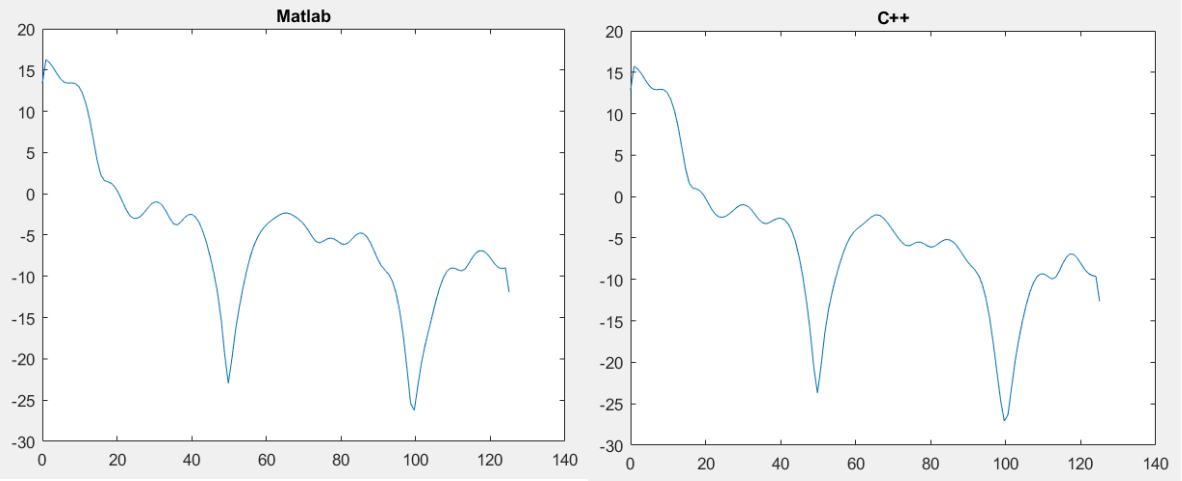
Seventh change of MBT_PWelchComputer.cpp:

Add a new method to compute fft and use it instead of MBT_Fourier::forwardBluesteinFFT.
This new nethod was found here: http://stackoverflow.com/questions/10121574/safe-and-fast-fft



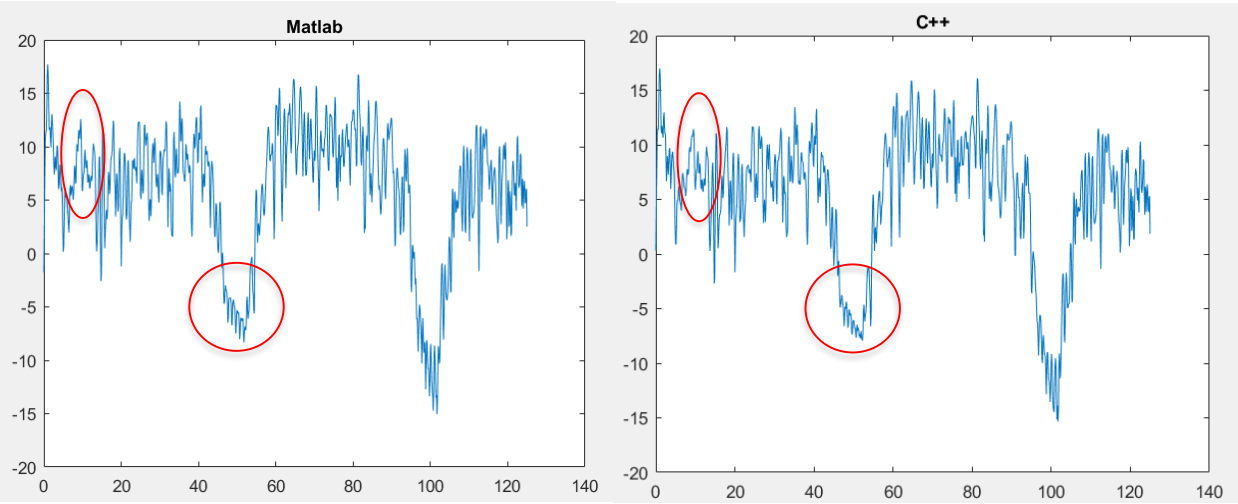
Same frequency vector : 1x1025 (from 0 to 125Hz)

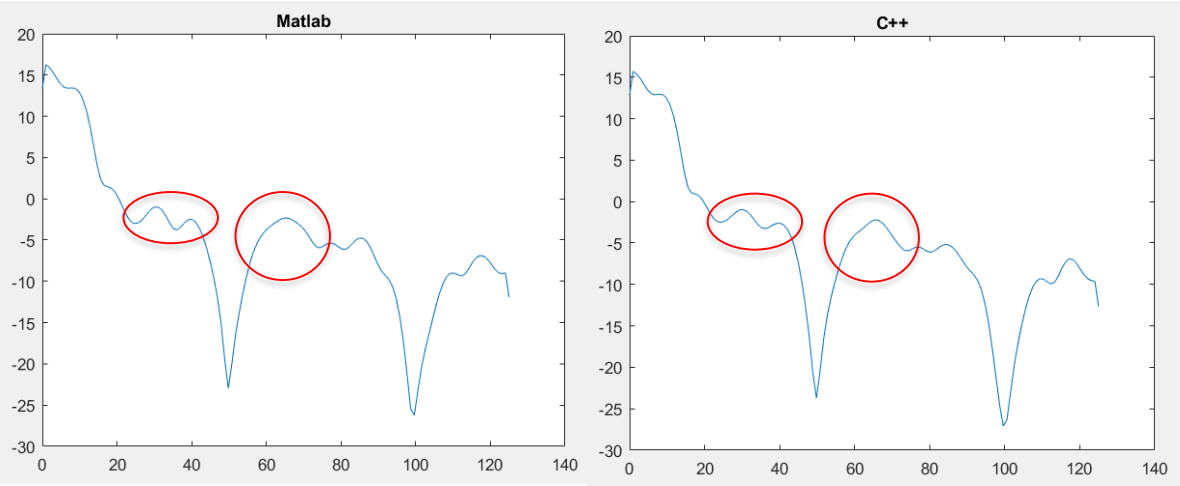
For a signal of 250 datapoints:



Conclusion: almost the same trend at the same level but not exactly exactly the same values.

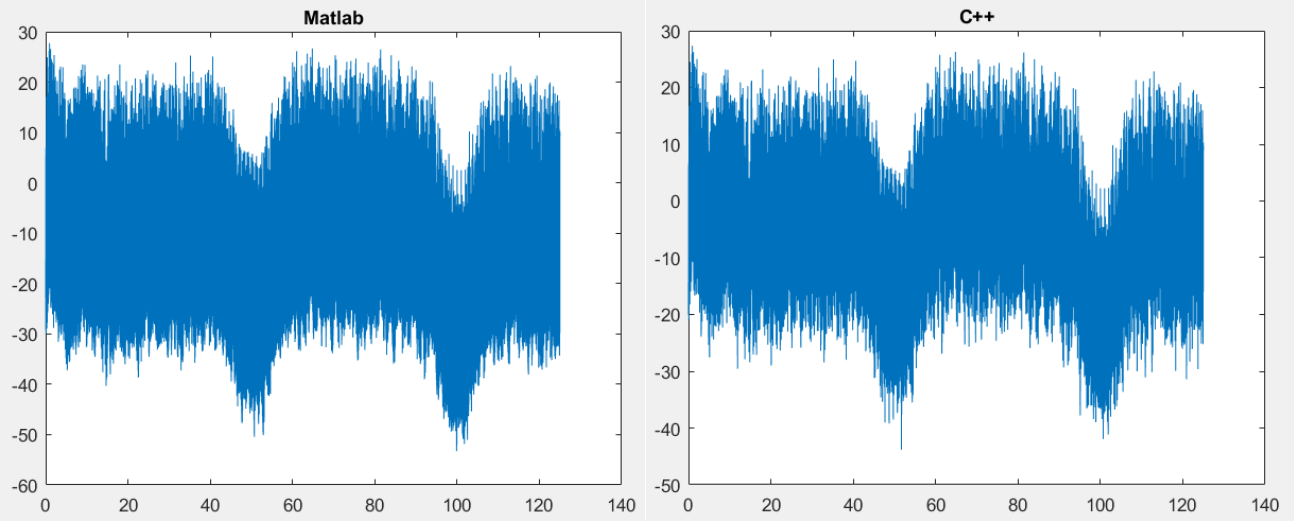
See below the little differences:





3. Comparison of the psd with a signal with 160000 datapoints.

160000 datapoints ~ 10.667min



Same frequency vector : 1x32769(from 0 to 125Hz)

In C++, a such signal need 213,856s = 3.56min to compute fft…

4. Adjustement of MBT_Fourier to have the same result than with fft

.

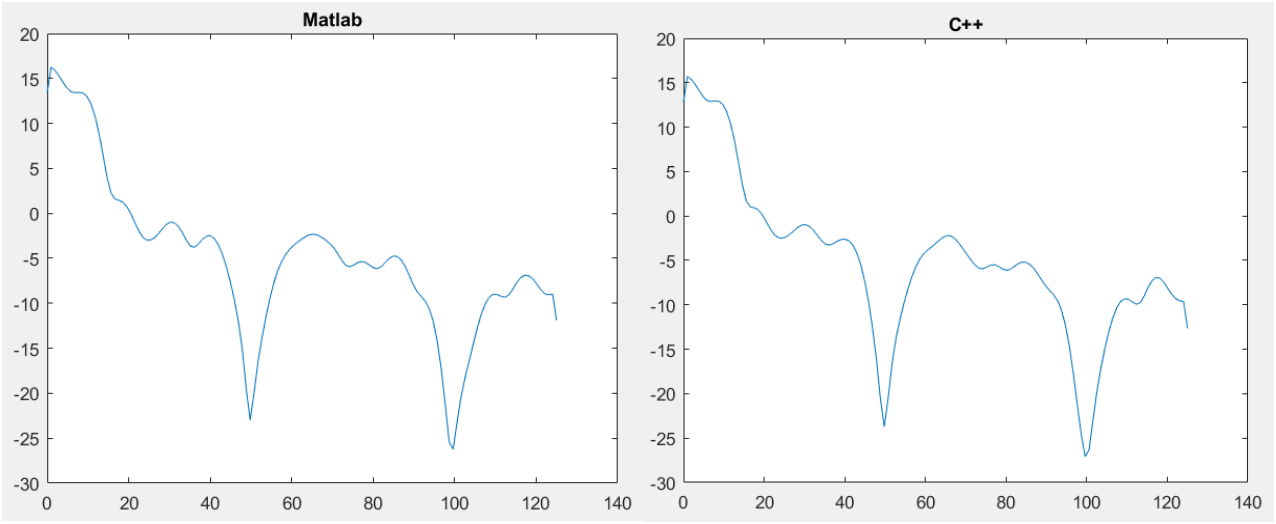
[psdM,fM] = pwelch(inputData, [],[],[],250)
VS
Seventh change of MBT_PWelchComputer and Third change of MBT_Fourier

Third change of MBT_Fourier.cpp:

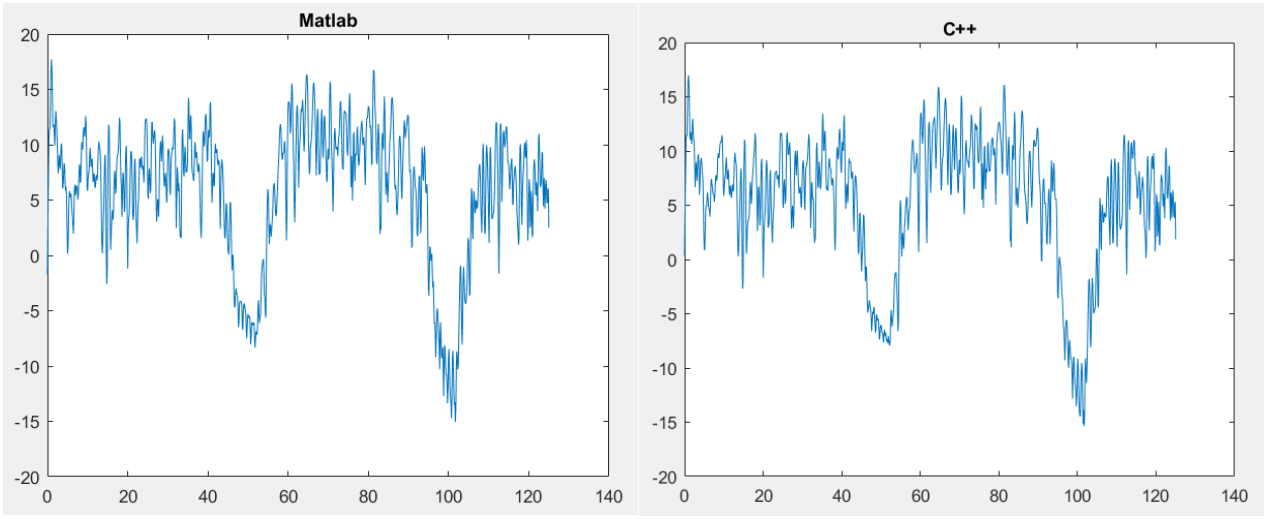
```
void   MBT_Fourier::forwardBluesteinFFT(std::vector<std::complex<float>   >&   inputData,   const
MBT_FourierOptions option)
{
    MBT_FourierBluestein::bluestein(inputData, getExponentSign(option));
    //forwardScaling(inputData, option); //No done in fft of
http://stackoverflow.com/questions/10121574/safe-and-fast-fft

}

template<typename T>
void MBT_Fourier::radix2Reorder(std::vector<T>& inputData)
{
    int j = 0;
    for (int i = 0; i < inputData.size() - 1; i++)
    {
        //Swap values
        if (i < j)
        {
            T temp = inputData[i];
            inputData[i] = inputData[j];
            inputData[j] = temp;
        }

        //int len = (int)inputData.size();
```

```
        int len = (int)inputData.size()/2; // like in
http://stackoverflow.com/questions/10121574/safe-and-fast-fft


        //do // Comment by Fanny Grosselin on 2017/01/20
        //{ // Comment by Fanny Grosselin on 2017/01/20
            //len >>= 1; // Comment by Fanny Grosselin on 2017/01/20
            j ^= len;
        //} // Comment by Fanny Grosselin on 2017/01/20
        while ((j & len) == 0)
        {// add these lines like in http://stackoverflow.com/questions/10121574/safe-and-fast-
fft

            len/=2;
            j^=len;
        };
    }
}


void    MBT_Fourier::radix2Step(std::vector<std::complex<float>    >&    inputData,    const    int
exponentSign, const int levelSize, const int indexInLevel)
{
    // Twiddle Factor
    const float  PI_F=3.14159265358979f;
    float exponent = (exponentSign * indexInLevel) * PI_F / levelSize;
    std::complex<float> w (cos(exponent), sin(exponent));

    //int step = levelSize << 1; // Commented by Fanny Grosselin 2017/01/20
    //for (int i = indexInLevel; i < inputData.size(); i += step)
    for (int i = indexInLevel; i < inputData.size(); i += 2*levelSize) // like on
http://stackoverflow.com/questions/10121574/safe-and-fast-fft

    {
        std::complex<float> value = inputData[i];
        std::complex<float> modifier = w * inputData[i + levelSize];
        inputData[i] = value + modifier;
        inputData[i + levelSize] = value - modifier;
    }
}
```

Signal of 250 datapoints:



Signal of 5000 datapoints:



Signal of 160000 datapoints: