Master MOSiG - M1

Report

Signal Fundamentals

Computer Exercises N. 1

Authors

Jander Botelho do Nascimento

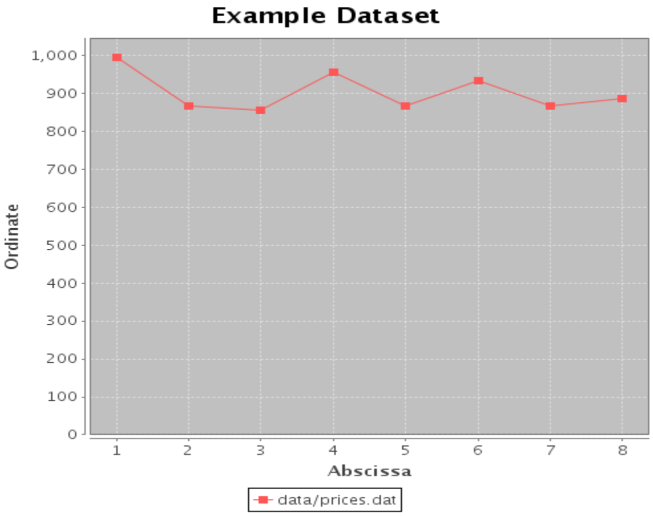
Raquel Araújo de Oliveira

Grenoble, 13 October 2010

Index

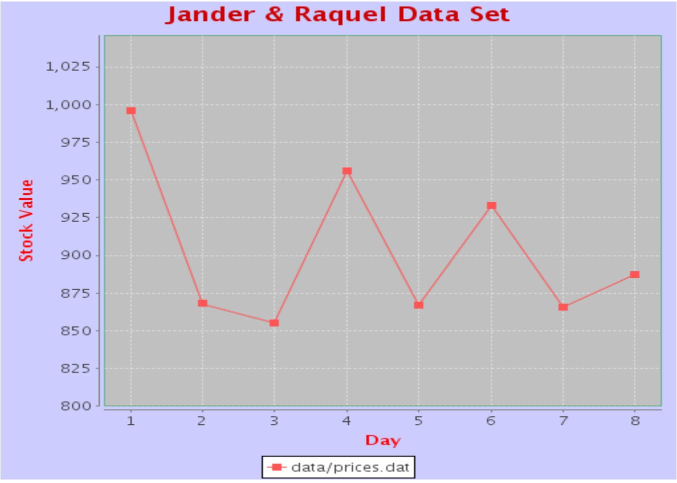
Basic Signal Processing

1.1-In the directory named data, create a file named prices.dat containing the predicted stock market price of the exercise number 6 (Financial Experts) of the exercises sheet.
Display it with the application.



1.2-By right-clicking on the figure and choosing Properties, change:

- the title of the graphic
- the abscissa and ordinate labels
- the scale to display the signal properly
- the color to personalize your figure.



1.3-In the file Signal.java, complete the methods public double getMean() and public double getStandardDev().

```
public double getMean() {
   int nbSamples = this.getNbSamples();
   double mean = 0.0;

   double totalSum=0.0d;
   //sum all values
   for(int i=0;i<nbSamples;i++){
      totalSum+=this.getValueOfIndex(i);
   }
   //calculates the mean
   mean=totalSum/nbSamples;
   return mean;
 }

public double getStandardDev() {
   int nbSamples = this.getNbSamples();
   double standardDev = 0.0;

   double totalSumVariation=0.0;

   for(int i=0;i<nbSamples;i++){
      double desviation=this.getMean()-this.getValueOfIndex(i);
      double squareOfDesviation=desviation*desviation;
      totalSumVariation+=squareOfDesviation;
   }

   totalSumVariation=totalSumVariation/(nbSamples-1);

   standardDev=Math.sqrt(totalSumVariation);

   return standardDev;
 }
```

1.4-What are the mean and the standard deviation of the expert's predictions?

```
//Mean: 903.25
//Standard deviation: 52.16603736671451
```

1.5-What are the accuracy and the precision of the expert's predictions?

According to the definitions which states that precision is how close the predictions are from each other and the accuracy is how close they are from the real value, we could say that:

σ <StandardDesviation> represents how close the predictions are for each other. So it can be used as our precision value.

Precision=52.166

The difference between the mean (903.25) and the real stock (876) is 27,25. That value represents 3,11% of the real stock (inaccuracy), so the accuracy is(inaccuracy - 100%):

Accuracy=96.88 %

1.6-Prove that for a given i, x_i will be counted in h_j if and only if (see paper-sheet)

From

$$jminVal+j\leq x<minVal+(j+1)*l$$

We can split into two parts and isolate j:

Part 1

$$x\geq l*j+minVal\Rightarrow$$
$$l*j\leq x-minVal\Rightarrow$$
$$j\leq\frac{x-minVal}{l}$$

Part 2

$$x<minVal+(j+l)*l\Rightarrow$$
$$x<minVal+l*j+l\Rightarrow$$
$$l*j>x-minVal-l\Rightarrow$$
$$j>\frac{x-minVal}{l}-1$$

Considering:

$$K=\frac{x-minVal}{l}$$

We may say:

Part 1: $j\leq K$

Part 2: $j>K-1$

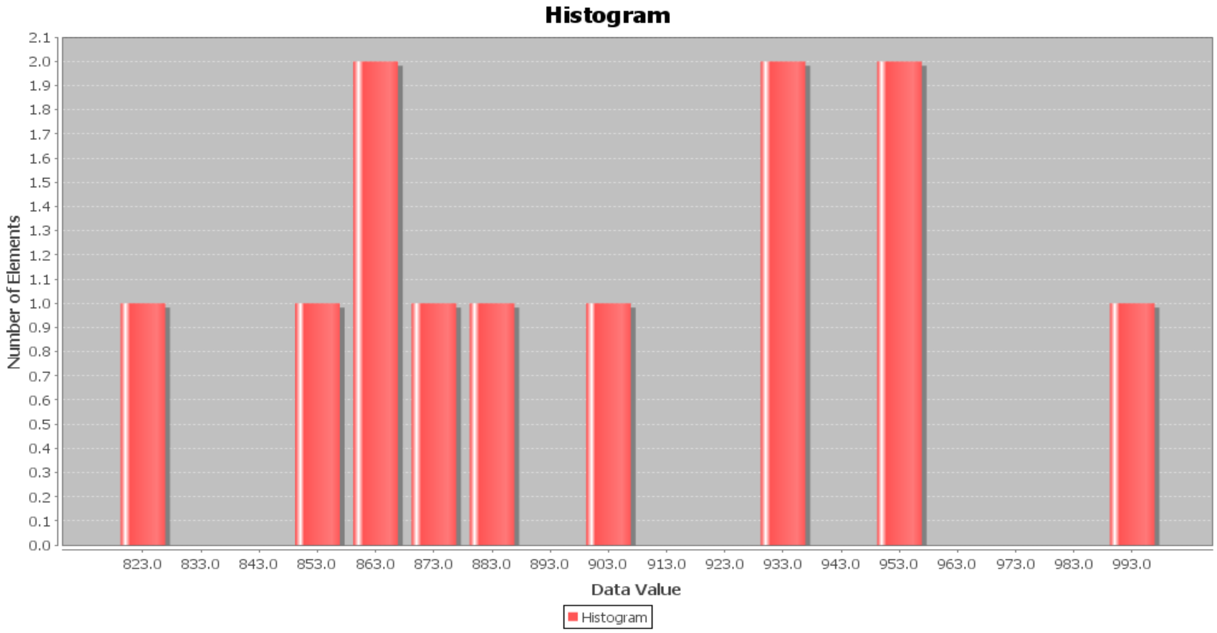
Conclusion:

So, the only way those points(Part 1 and Part 2) intersect is on K, or $K=\frac{x-minVal}{l}$.

1.7-Complete the function public Signal getHitogram(double intervalLength) of the file Signal.java.

```
public Signal getHitogram(double intervalLength) {
    Signal histo = new Signal();
    if (intervalLength <= 0.0) {
      System.err.println("Error in computeHistogram method from Signal:\n The length of an interval must be strictly positive.\n");
      return null;
    }

    int cont;
    for (double valor = this.getMin(); valor<=this.getMax(); valor=valor+intervalLength){
        cont=0;
        for (int j=0; j<=this.getNbSamples()-1;j++){
            if ((this.getValueOfIndex(j)>=valor) && (this.getValueOfIndex(j)<valor+intervalLength)){
                cont++;
            }
        }

        histo.addElement(valor + (intervalLength/2), cont);
    }

    return histo;
  }
```

1.8-Display the histogram the the predicted stock market price with an interval length of 10.



Digital Noise Processing

2.1-Complete the following function returning a signal containing nbElements random numbers between min and max.

```
public static Signal createRandomSeries(double min, double max, int nbElements) {
   Signal resultSignal = new Signal();
   resultSignal.settName("Random Signal");

   for(int i=0;i<=nbElements;i++){
       double randomValue=min+Math.random()*max;
       resultSignal.addElement(i, randomValue);
   }

   return resultSignal;
 }
```

2.2-Compute uniforms random signal and their mean and standard deviation for the following values

- minVal: 0.0

- maxVal: 1.0
- nbElements: 10, 100, 1000 and 10000

and illustrate that the more samples you generate, the closer you should get to the theoretical values.

The theoretical mean formula is:

$$\mu_{theory} = \frac{\max Val + \min Val}{2}$$

When maxVal=1 and minVal=0, the theoretical mean is:

$$\mu_{theory} = \frac{1 + 0}{2}$$

$$\mu_{theory} = 0.5$$

The theoretical standard deviation formula is:

$$\sigma_{theory} = \frac{\max Val - \min Val}{\sqrt{12}}$$

When maxVal=1 and minVal=0, the theoretical standard deviation is:

$$\sigma_{theory} = \frac{1 - 0}{\sqrt{12}}$$

$$\sigma_{theory} = 0.2886$$

Applying the algorithm showed at the question number 2.1, we made 3 cycles of tests(just to be sure), each cycle with 10, 100, 1000 and 10000 randoms elements between 0.0 and 1.0. For each signal was calculated their mean and standard deviation and the results are bellow:

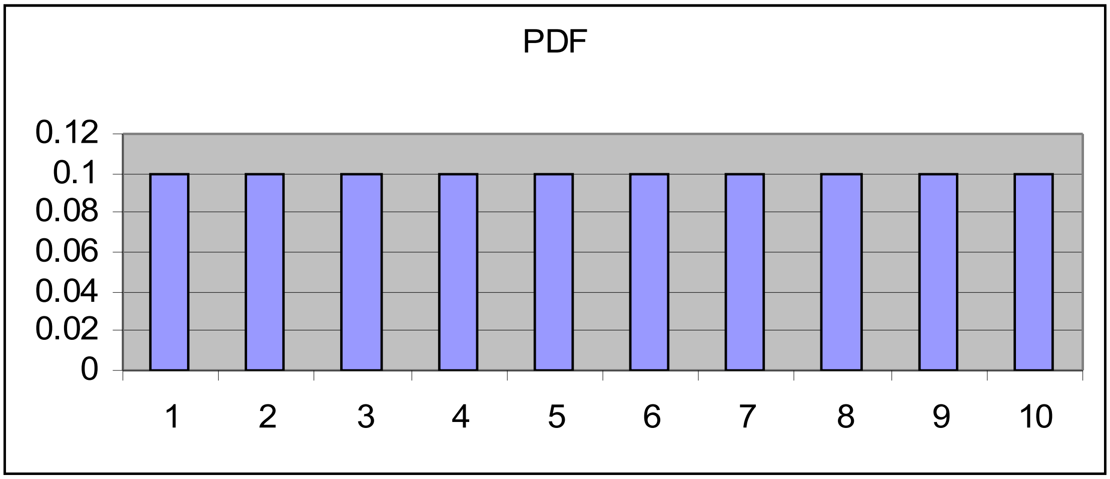
Number of samples	**First Signal**		**Second Signal**		**Third Signal**	
	mean	**standard deviation**	**mean**	**standard deviation**	**mean**	**standard deviation**
10	0.4822	0.2784	0.4454	0.2572	0.3196	0.1845
100	0.4928	0.2845	0.4899	0.2828	0.4911	0.2835
1000	0.4996	0.2884	0.4991	0.2881	0.4984	0.2877
10000	0.4997	0.2885	0.4998	0.2886	0.4998	0.2885

As we can see, the more samples we generate, the closer we get to the theoretical values of the mean and the standard deviation.

2.3-What is the pdf of a uniform random signal? Illustrate that the more samples you generate, the closer the histogram gets to the pdf.

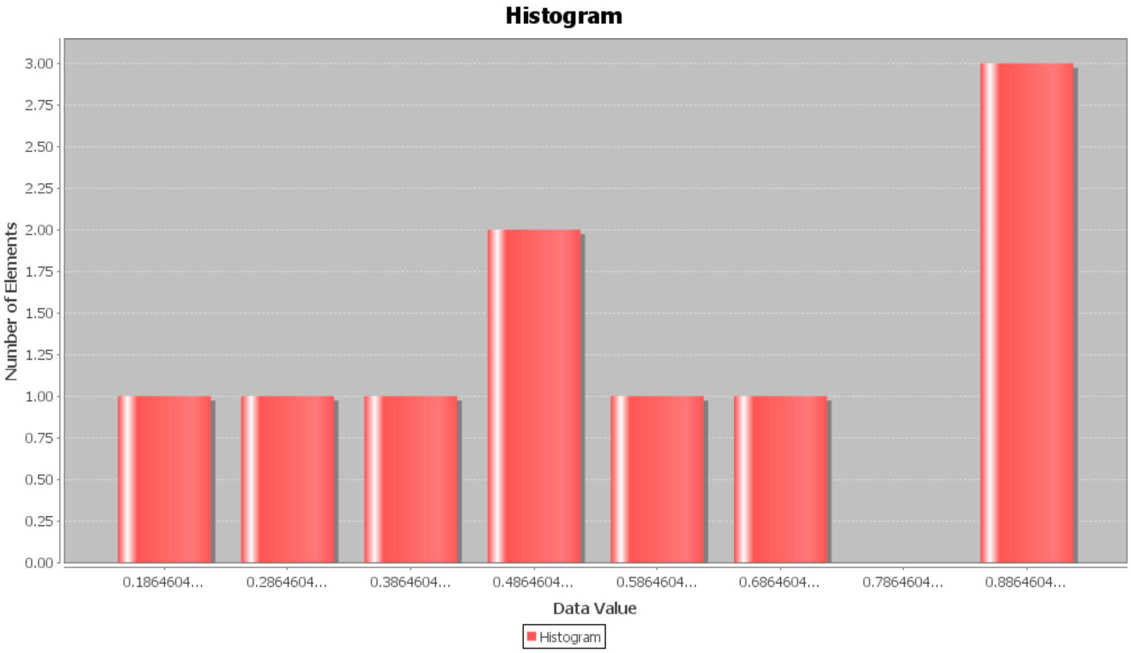
A PDF (Probability Density Function) of a uniform random signal tells us the probability of occur each element of the signal.

Let’s say that a signal has 10 elements (samples). The probability of each element occur at the signal is 1/10. The PDF graphic looks like this one:

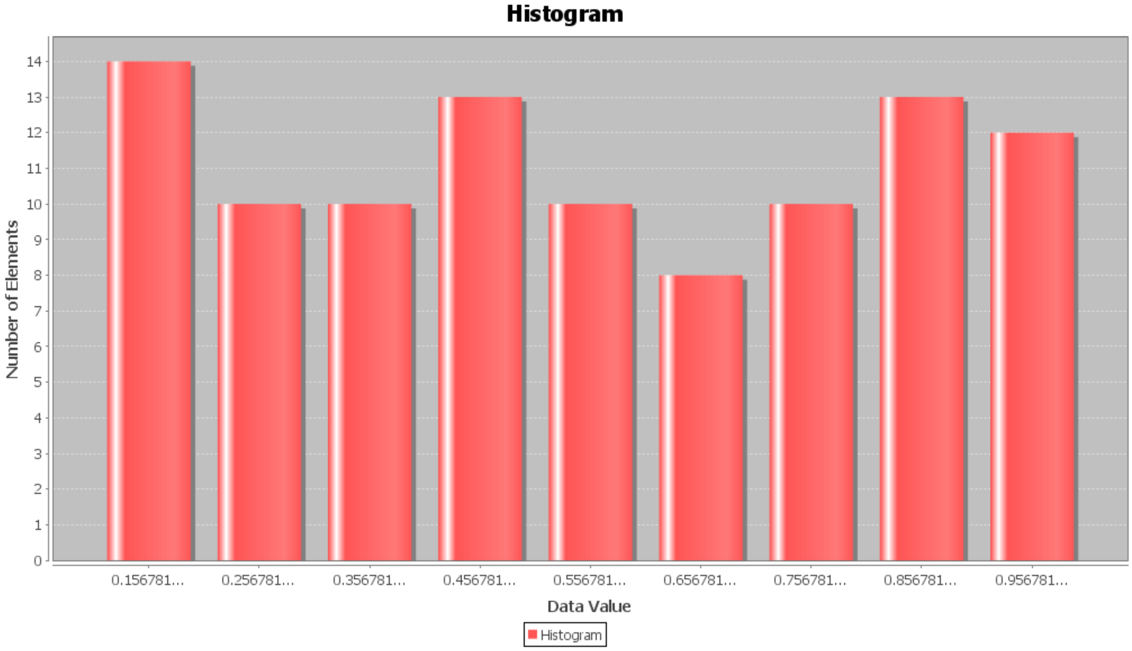


To illustrate that the more samples we generate, the closer the histogram gets to the pdf, lets take some examples.

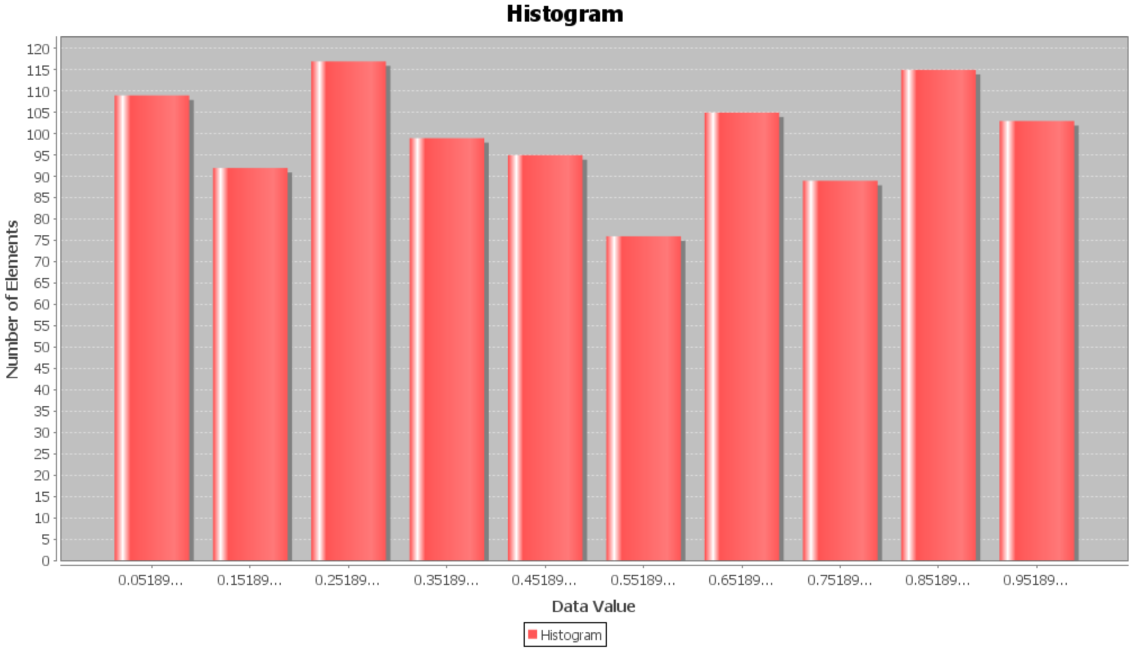
With 10 samples, the histogram is the following:



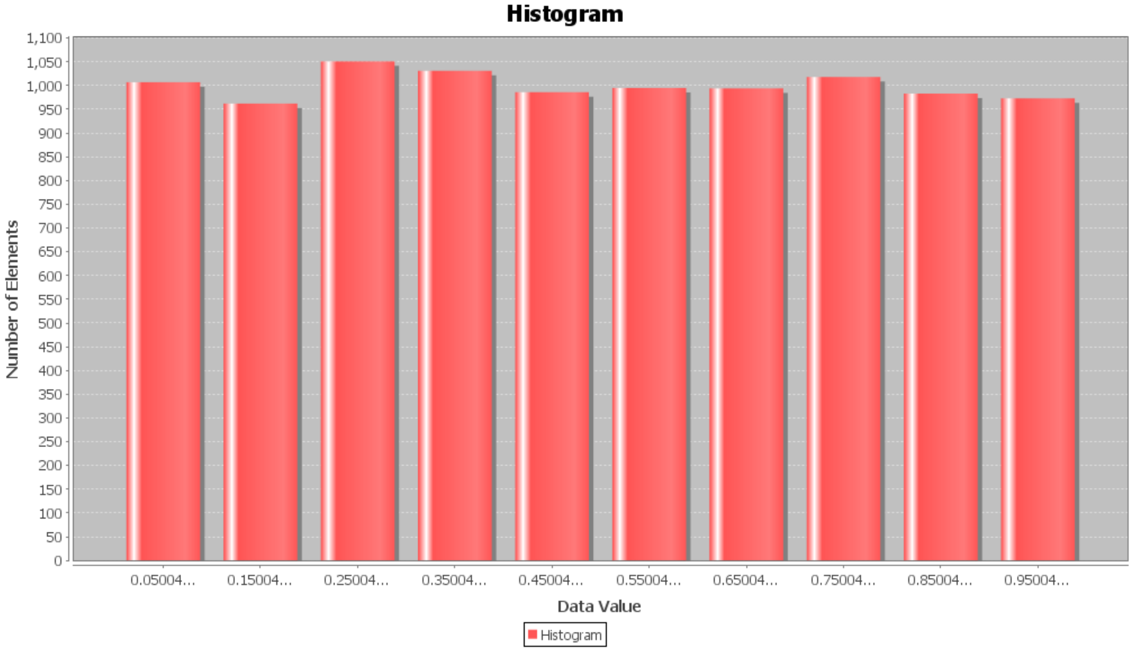
With 100 samples:



With 1.000 samples:



With 10.000 samples:



So, as we can see, the more samples we generate the histogram gets closer to the format of PDF.

2.4-What are the theoretical mean and standard deviation of a signal obtained by added two random numbers generated independently between 0:0 and 1:0 to form each sample?

The theoretical mean of a signal obtained by added 2(two) random numbers generated independently between 0.0 and 1.0 to form each sample is equal the sum of the 2(two) means of the signals. The same for the variance. The variance of a signal obtained by added two random numbers generated independently between 0.0 and 1.0 to form each sample is equal the sum of the two variance of the signals.

2.5-What are the theoretical mean and standard deviation of a signal obtained by adding twelve random numbers generated independently between 0:0 and 1:0 to form each sample?

The theoretical mean of a signal obtained by added 12(twelve) random numbers generated independently between 0.0 and 1.0 to form each sample is equal the sum of the 12(twelve) means of the signals. The same for the variance. The variance of a signal obtained by added two random numbers generated independently between 0.0 and 1.0 to form each sample is equal the sum of the two variance of the signals.

2.6-Deduce from the previous question an algorithm to generate a Gaussian noise signal of mean (sigma)= 0 and standard deviation (sigma)= 1.

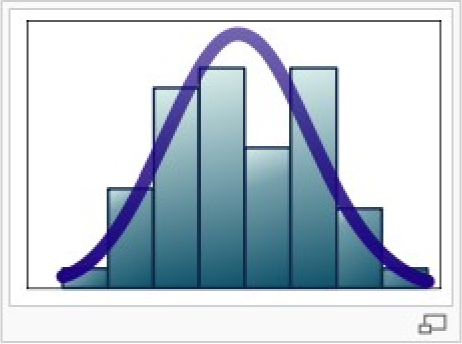
SKIP

2.7-Complete the corresponding code of the function public static Signal createGaussianNoiseSeries(int nbElements) in the file Signal.java.

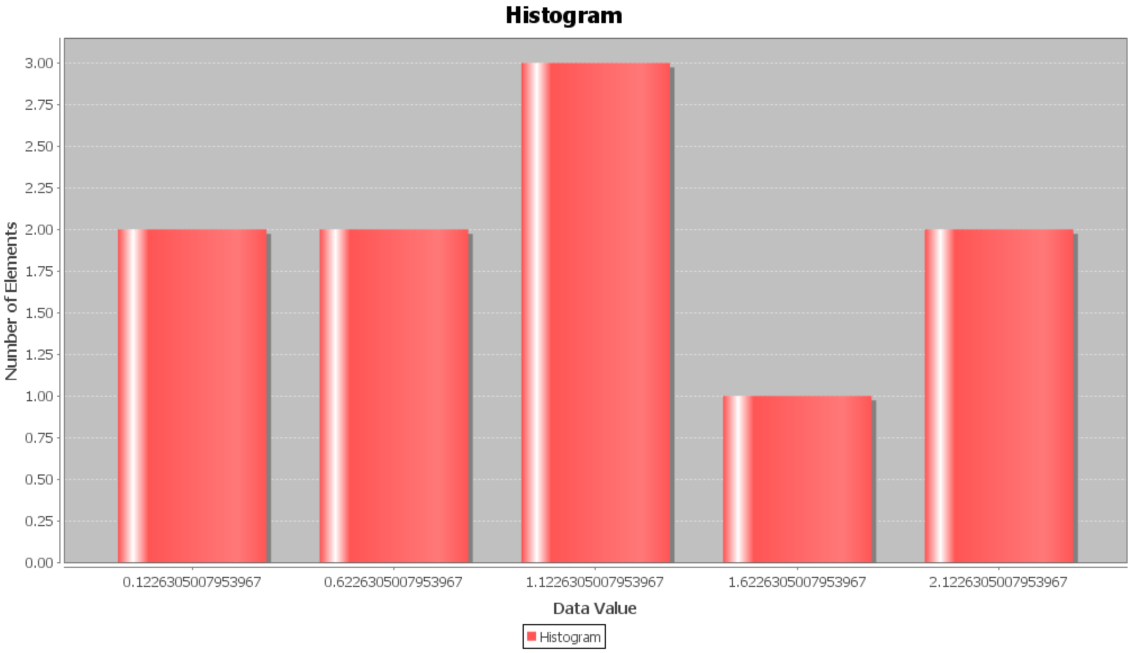
```
public static Signal createGaussianNoiseSeries(int nbElements) {
  Signal resultSignal = new Signal();
  float standardDesviation = 1;
  float mean = 0;

  for (int i = 0; i < nbElements; i++) {
    float rnd1 = (float)Math.random();
    float rnd2 = (float)Math.random();
    float ln=(float)-2d*(float)Math.log(rnd1);
    float sqrtLn=(float)Math.sqrt(ln);
    float cosRad=(float)2 * (float)Math.PI * rnd2;
    //based on http://www.dspguru.com/dsp/howtos/how-to-generate-white-gaussian-noise
    resultSignal.setValueOf(i,sqrtLn*Math.cos(cosRad) * standardDesviation + mean);
  }

  return resultSignal;
}
```

2.8-Illustrate that the more samples you generate and the smaller the length of histogram intervals are, the closer you get to a Gaussian by displaying
your signals histograms and a Gaussian curve.

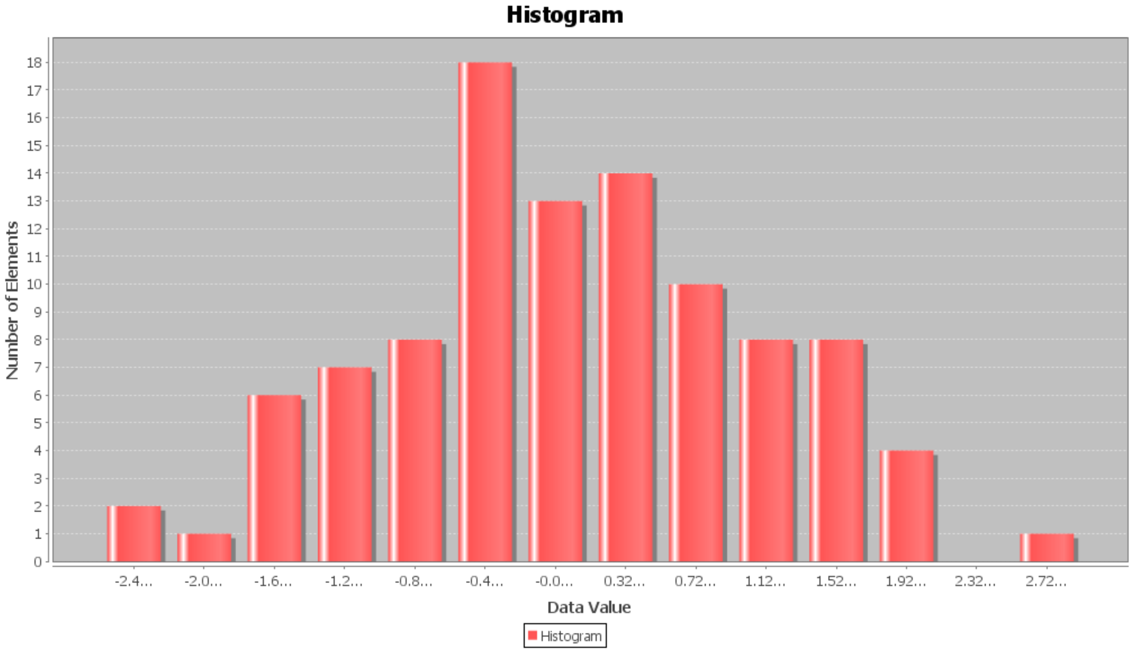
To illustrate that the more samples we generate and the smaller the length of histogram intervals are, the closer we get to a Gaussian, first we’ll display how a Gaussian curve is:



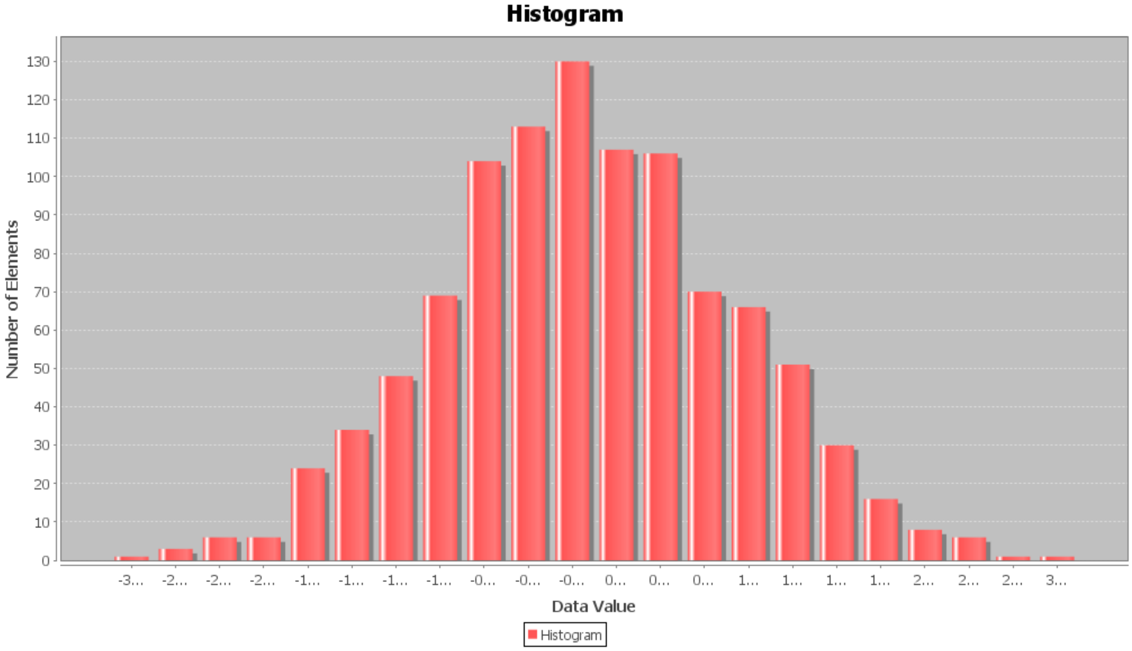
With 10 samples and interval 0.5, the histogram is the following:



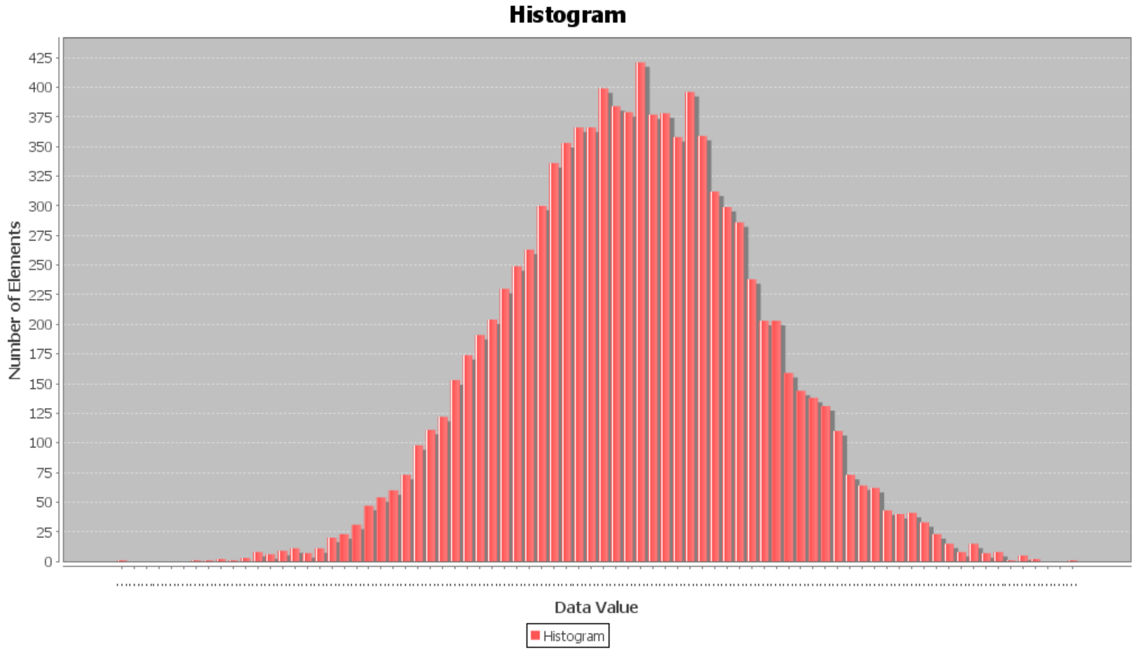
With 100 samples and interval 0.4:



With 1.000 samples and interval 0.3:



With 10.000 samples and interval 0.1:



So, as we can see, the more samples we generate and the smaller the length of histogram intervals are, the closer we get to a Gaussian.

Question To go further

Write an algorithm to generate a Gaussian noise signal with a given mean and a given standard deviation.

SKIP

Signal Decomposition

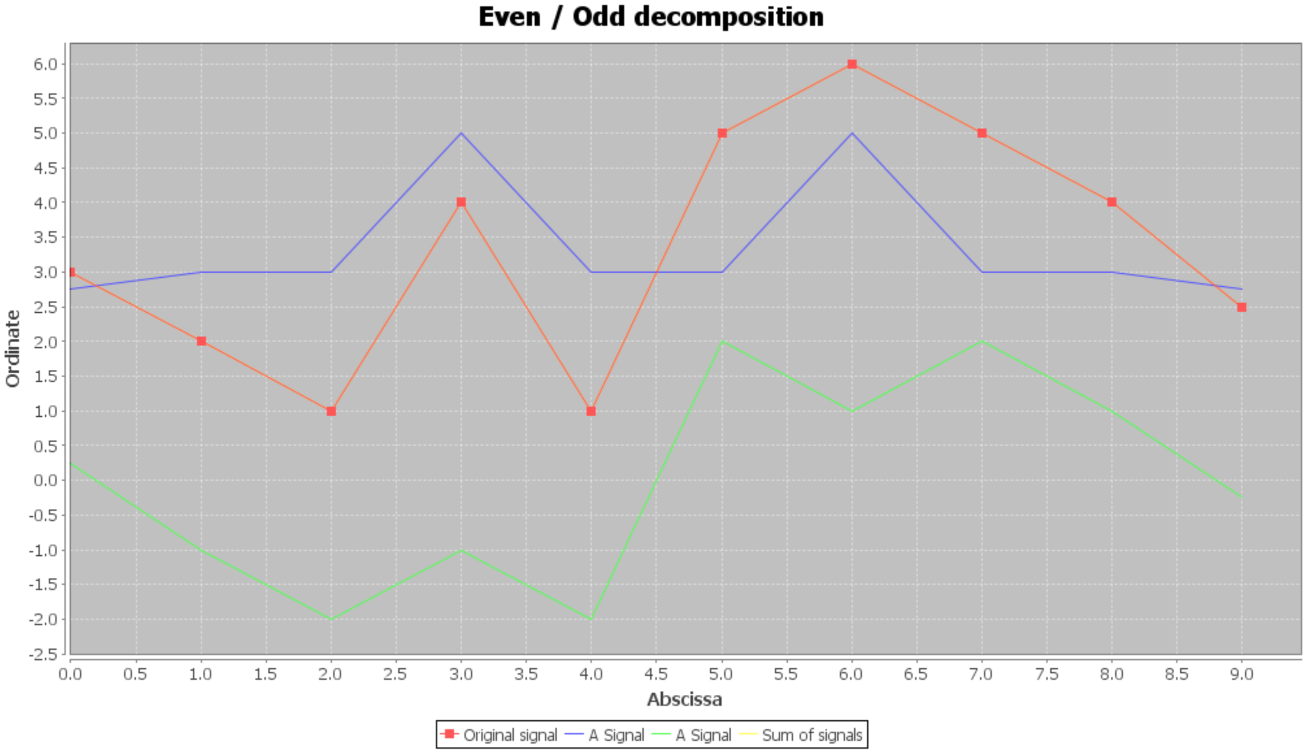
3.1-Complete these methods: public ArrayList<GeneralSignal> evenOddDecomposition ( ), public ArrayList<GeneralSignal> interlacedDecomposition ( ).

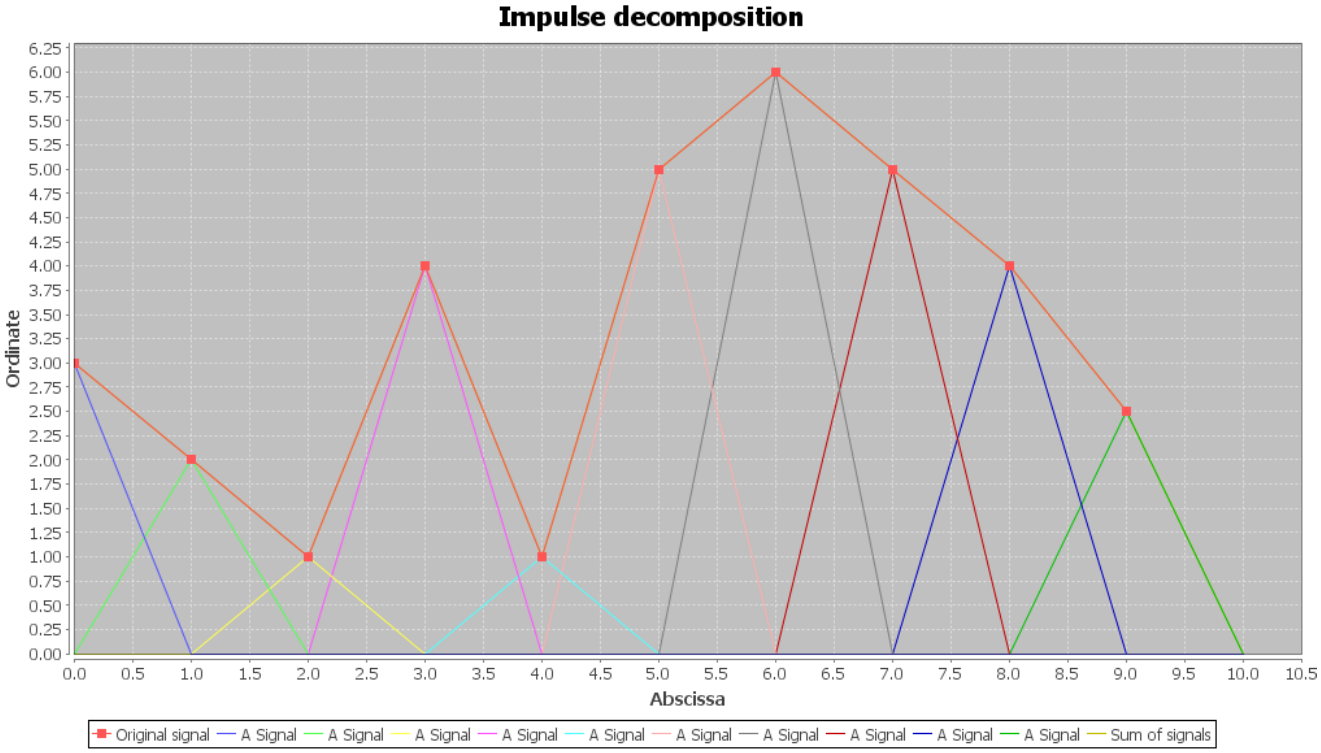
```
public ArrayList<GeneralSignal> evenOddDecomposition() {
    ArrayList<GeneralSignal> list = new ArrayList<GeneralSignal>();

    double x,y;
    int n_ele = this.getNbSamples();

    Signal even_signal = new Signal();
    Signal odd_signal = new Signal();
```

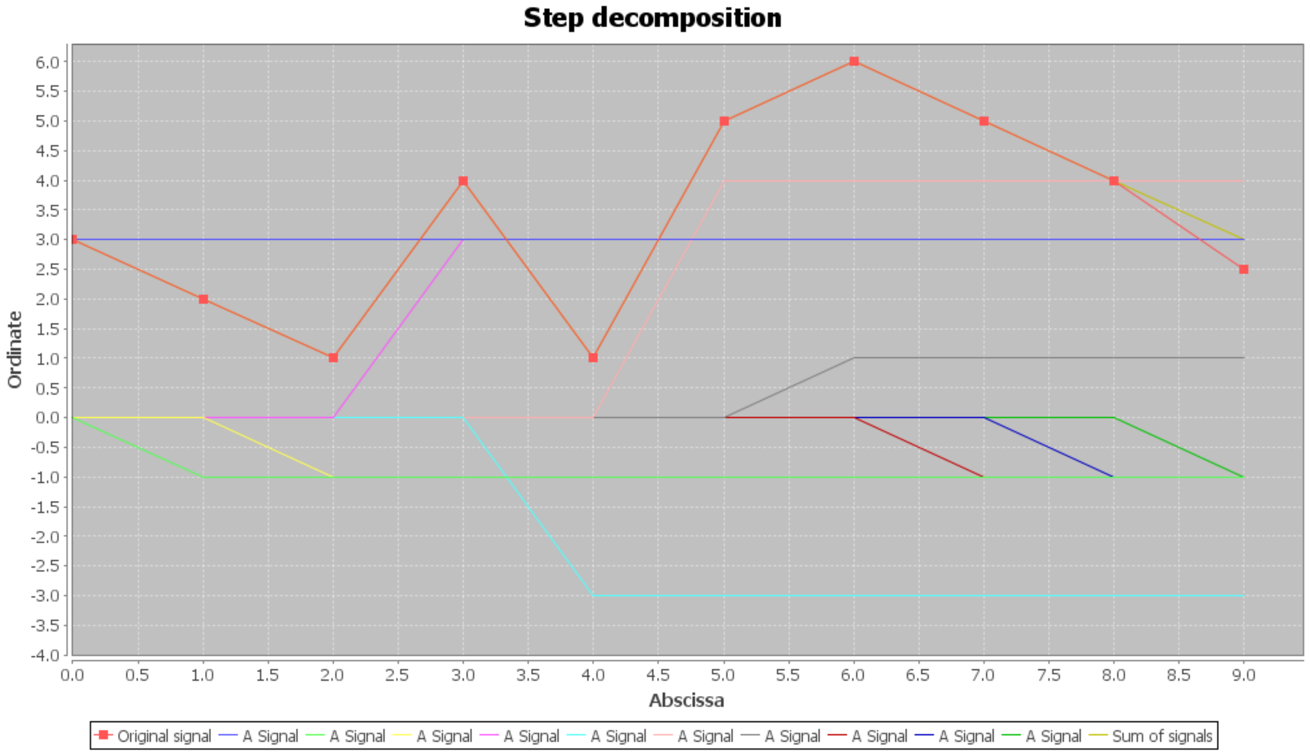
```
  for(int i=0;i<=this.getNbSamples()-1;i++){
    x=this.getAbscissaOfIndex(i);
    y=this.getValueOfIndex(i); //returns the y value of the index

    //System.out.println("n "+i+"=> x:"+x+" x(n):"+y+" N:"+n_ele+" y[N-n]:"+this.getValueOfIndex(n_ele-1-i));
    even_signal.addElement(x, (y + this.getValueOfIndex(n_ele-1-i))/2);
    odd_signal.addElement(x, (y - this.getValueOfIndex(n_ele-1-i))/2);
  }
  list.add(even_signal);
  list.add(odd_signal);

  return list;
 }

public ArrayList<GeneralSignal> stepDecomposition() {
  ArrayList<GeneralSignal> list = new ArrayList<GeneralSignal>();

  // Write your code here
  double x,y;
  int prior;

  for(int n=0;n<=this.getNbSamples()-1;n++){
    x=this.getAbscissaOfIndex(n);
    y=this.getValueOfIndex(n);
    Signal signal = new Signal();
    for(int k=0;k<=this.getNbSamples()-1;k++){
      if (k<n){
        signal.addElement(k, 0);
      }
      else{
        if (n==0){
          signal.addElement(k, y);
        }
        else{
          prior=(int) (y-this.getValueOfIndex((int) x-1));
          signal.addElement(k, prior);
        }
      }
    }
    list.add(signal);
   }
  return list;
 }

 public ArrayList<GeneralSignal> interlacedDecomposition() {
  ArrayList<GeneralSignal> list = new ArrayList<GeneralSignal>();

  // Write your code here
  double x,y;

  Signal even_signal = new Signal();
  Signal odd_signal = new Signal();

  for(int i=0;i<=this.getNbSamples()-1;i++){
    x=this.getAbscissaOfIndex(i);
    y=this.getValueOfIndex(i); //returns the y value of the index

    if ((x % 2) == 0){ //then it is even
      even_signal.addElement(x, y);
      odd_signal.addElement(x, 0);
    }
    else{
      even_signal.addElement(x, 0);
      odd_signal.addElement(x, y);
    }

  }
  list.add(even_signal);
  list.add(odd_signal);

  return list;
 }

public ArrayList<GeneralSignal> impulseDecomposition() {
  ArrayList<GeneralSignal> list = new ArrayList<GeneralSignal>();

  // Write your code here
  double x,y;

  for(int n=0;n<=this.getNbSamples()-1;n++){
    x=this.getAbscissaOfIndex(n);
    y=this.getValueOfIndex(n); //returns the y value of the index
    Signal signal = new Signal();
    for (int k=0;k<=this.getNbSamples();k++){
      if(k==n){
        signal.addElement(k, y);
      }
      else{
        signal.addElement(k, 0);
      }
    }
    list.add(signal);
  }

  return list;
}
```

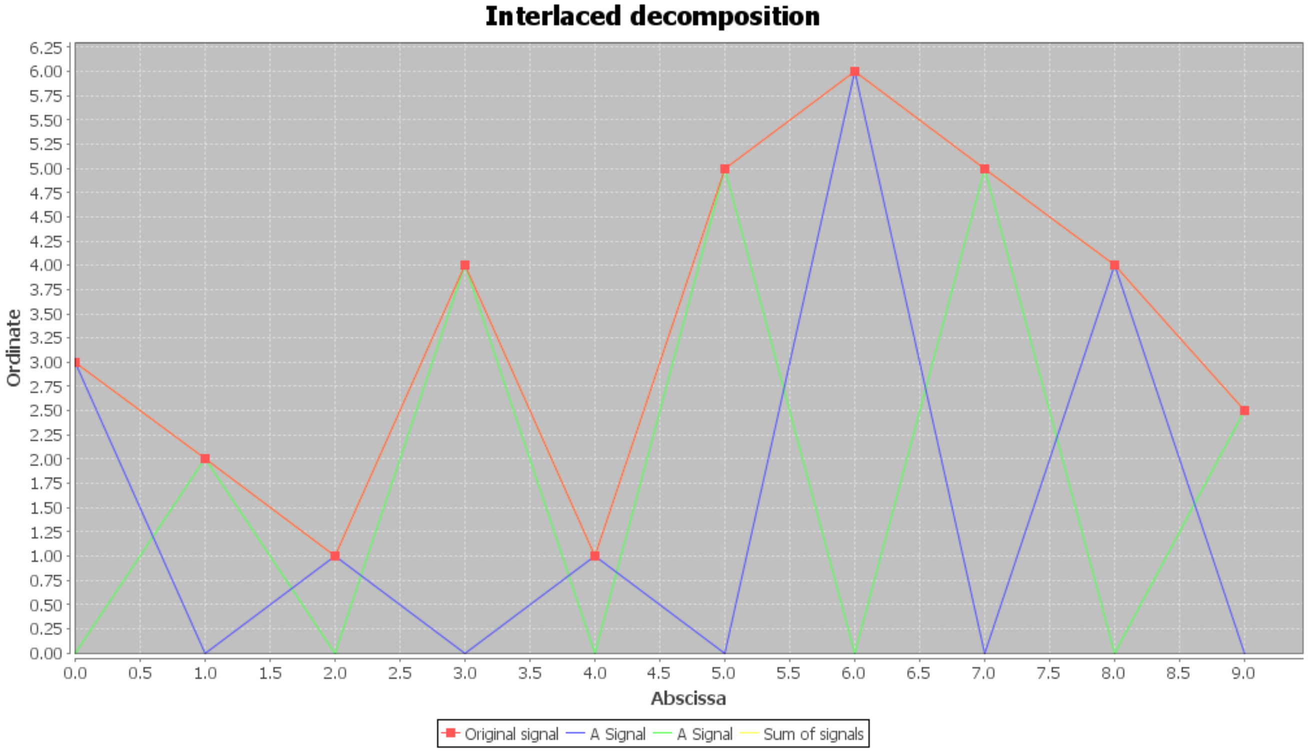
3.2-Create a file containing a small signal (10 samples), either by hand or by generating a random signal (see previous computer exercise).

Test and illustrate the decomposition of this signal.


Even / Odd decomposition
Ordinate
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Abscissa
Original signal
A Signal
A Signal
Sum of signals


Impulse decomposition
Ordinate
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
10.5
Abscissa
Original signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
Sum of signals


Step decomposition
Ordinate
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Abscissa
Original signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
A Signal
Sum of signals


Interlaced decomposition
Ordinate
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Abscissa
Original signal
A Signal
A Signal
Sum of signals