Estimation of standard deviation of error using Monte Carlo method

Amritam Sarcar

September 25, 2008

1 Program

```
import java.util.Random;
public class MonteCarloMethod {

                public static void main(String args[]){
                /* SigmaCompute <sigma> <x_bar(s)> <iter>*/
                MonteCarloMethod sC = new MonteCarloMethod();
                int iter;
                double y_bar, y, delta_y = 0.0, sigma_actual,sigma_bar;
                double[] sigma_given, x_bar, delta_x, x_actual;
                // get Sigma
                sigma_given = sC.getSigma(args[0]);
                // get x_bar
                x_bar = sC.getX(args[1]);
                // get iter
                iter = sC.getIter(args[2]);
                // compute value
                y_bar = sC.f(x_bar);
                // compute sigma using Monte carlo
                for(int i=0; i<iter;i++){
                // compute delta_Xi
                delta_x = sC.computeDeltaX(x_bar, sigma_given);
                // compute X_actual
                x_actual = sC.computeXActual(delta_x, x_bar);
                // compute y
                y = sC.f(x_actual);
                // compute delta_y
                delta_y += Math.pow(y_bar-y, 2.0);
                }
                // compute sigma
                delta_y = (delta_y/iter);
                sigma_actual = Math.sqrt(delta_y);
                // print the results
                System.out.println("Sigma Computed :: " + sigma_actual);
        }
```

```
/**
 * get the given sigma values
 **/
public double[] getSigma(String args){
        // assuming valid input --> no error checking code
        String[] tempString;
        double[] sigma;
        // fetch Sigma
        tempString = args.split(",");
        sigma = new double[tempString.length];
        for(int i=0;i<tempString.length;i++)
                sigma[i] = Double.parseDouble(tempString[i]);
        return sigma;
}

/**
 * get the given values for X
 **/
public double[] getX(String args){
        String[] tempString;
        double[] x_bar;
        // fetch X_bar
        tempString = args.split(",");
        x_bar = new double[tempString.length];
        for(int i=0;i<tempString.length;i++)
                x_bar[i] = Double.parseDouble(tempString[i]);
        return x_bar;
}

/**
 * get the no. of iterations
 **/
public int getIter(String args){
        return Integer.parseInt(args);
}

/**
 * Definition of function f
 **/
private double f(double[] x){
        // sum --> function given
        double sum = 0.0;
        for(int i = 0; i< x.length; i++)
                sum += x[i];
        return sum;
}

/**
 * Computing delta_x
 **/
private double[] computeDeltaX(double[] x_bar, double[] sigma){
        double[] delta_x = new double[x_bar.length];
        for(int i=0;i<x_bar.length;i++)
                delta_x[i] = sigma[i]*gauss();
        return delta_x;
}
```

```
        /**
         * computing x_actual
         **/
        private double[] computeXActual(double[] delta_x, double[] x_bar){
                double[] x_actual = new double[delta_x.length];
                for(int i=0;i<x_bar.length;i++)
                        x_actual[i] = x_bar[i] - delta_x[i];
                return x_actual;
        }

        /**
         * finding gauss
         **/
        private double gauss(){
                double gauss = 0.0;
                Random gen = new Random();
                for(int i=0;i<12;i++)
                        gauss += gen.nextDouble() -0.5;
                return gauss;
        }
}
```

2 Output

The above program was run several times by varying the parameters. The sample outputs are as follows -

```
amritam-sarcars-macbook:Desktop Amritam$ java MonteCarloMethod 10,20,30
2,7,18 1000
Sigma Computed ::  38.03793631154864
amritam-sarcars-macbook:Desktop Amritam$ java MonteCarloMethod 10,10,10
2,7,18 1000
Sigma Computed ::  18.386093338376533
amritam-sarcars-macbook:Desktop Amritam$ java MonteCarloMethod 10,10,10
0,0,0 1000
Sigma Computed ::  17.78168250383329
```