Program Structures and Algorithms
Spring 2023(SEC – 8)

NAME: Daiming Yang
NUID: 002771605

Task: Assignment 1 (Random Walk)

Imagine a drunken man who, starting out leaning against a lamp post in the middle of an open space, takes a series of steps of the same length: 1 meter. The direction of these steps is randomly chosen from North, South, East or West. **After m steps, how far (*d*), generally speaking, is the man from the lamp post?** Note that *d* is the Euclidean distance of the man from the lamp-post.

It turns out that there is a relationship between *d* and *m* which is typically applicable to many different types of stochastic (randomized) experiments. Your task is to implement the code for the experiment and, most importantly, to **deduce the relationship**.

Relationship Conclusion:

$$\boldsymbol{d = 0.87 * n^{0.5}}$$

Evidence to support that conclusion:
After 1000 experiments separately for from 1 to 999 steps, the distances for each time is stored by testplot.java. Then those data are plotted and analysed by using Matlab.
As shown in the Matlab curve fitter tool, the result of random walk perfectly fits the power regression model, which is $a * x^b$. With 95% confidence bounds, the Coefficients are $a \approx 0.0.8723, b \approx 0.5025$.

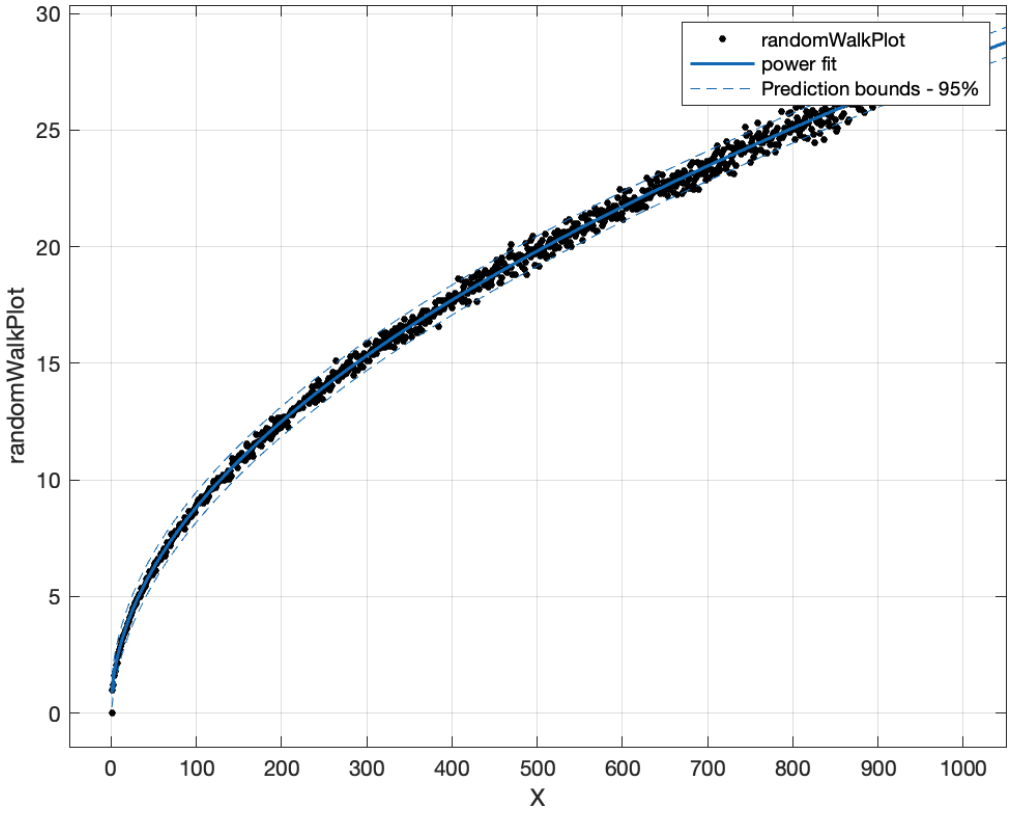


Figure 1 power fit

Graphical Representation:

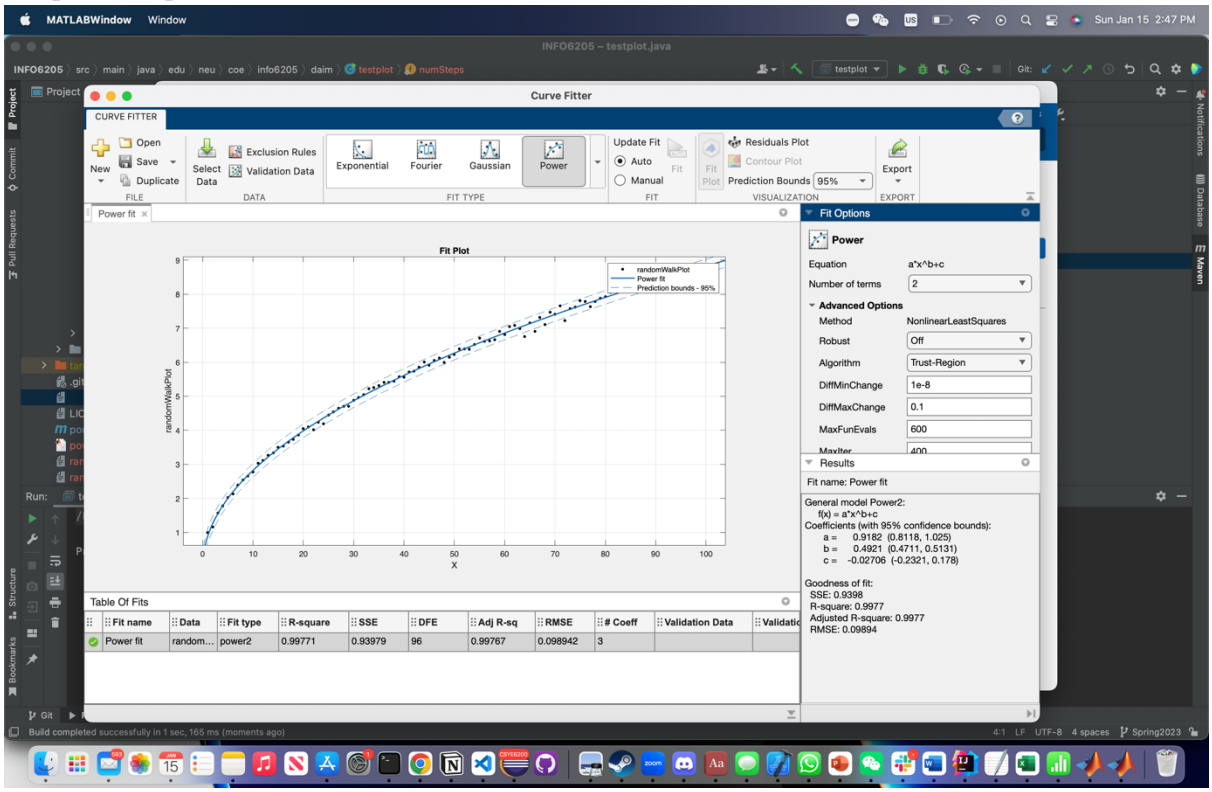


Fig.2.Curve fitter for 99 steps

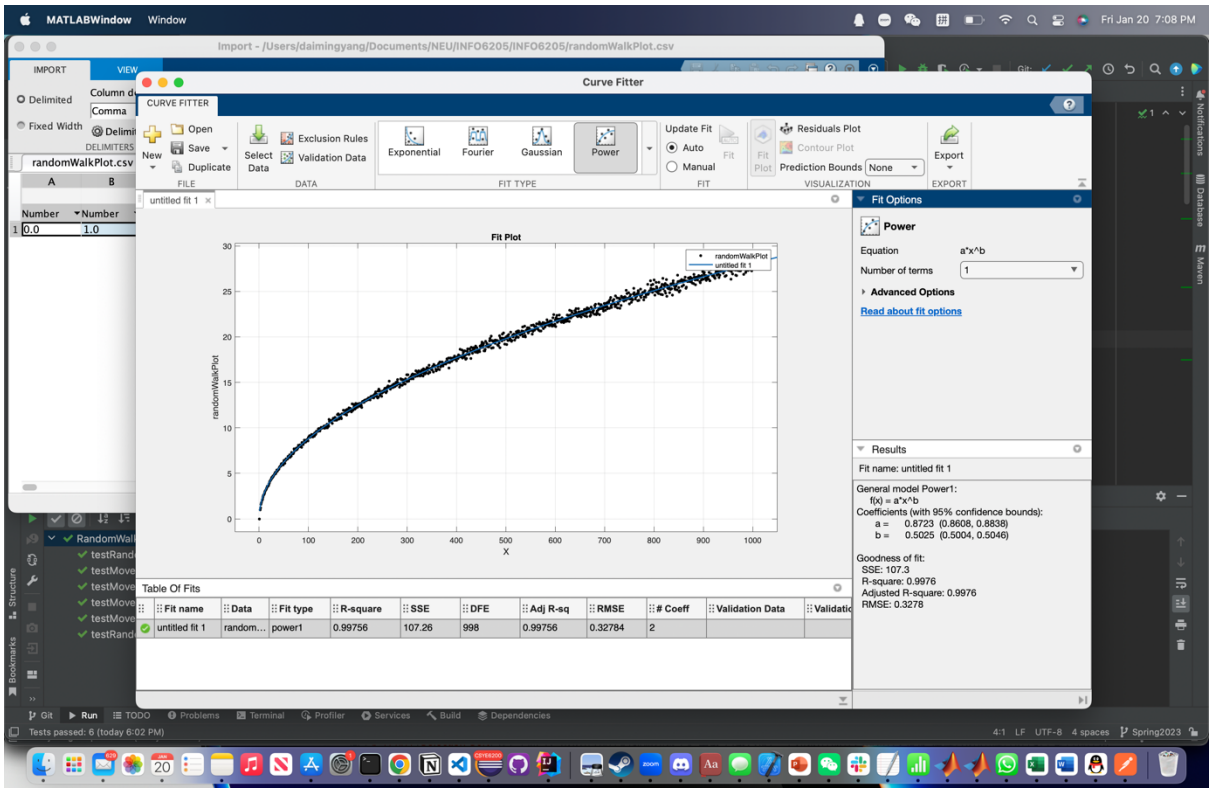
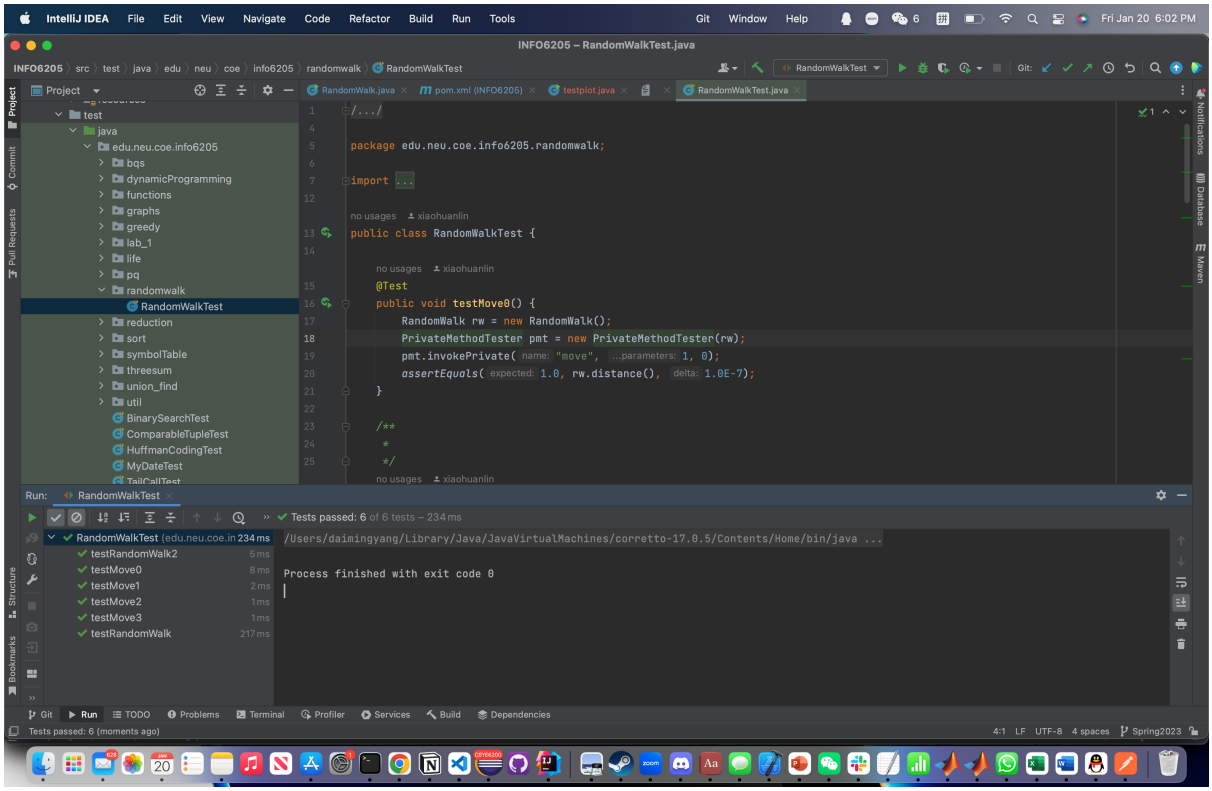


Fig.3.Curve fitter for 999 steps

Unit Test Screenshots:



Code:

RandomWalk

```
/*
 * Copyright (c) 2017. Phasmid Software
 */

package edu.neu.coe.info6205.randomwalk;

import java.util.Random;

public class RandomWalk {

    private int x = 0;
    private int y = 0;

    private final Random random = new Random();

    /**
     * Private method to move the current position, that's to say the
drunkard moves
     *
     * @param dx the distance he moves in the x direction
     * @param dy the distance he moves in the y direction
     */
    private void move(int dx, int dy) {
        // FIXME do move by replacing the following code
//          throw new RuntimeException("Not implemented");
        x += dx;
        y += dy;
        // END
    }

    /**
```

```
     * Perform a random walk of m steps
     *
     * @param m the number of steps the drunkard takes
     */
    private void randomWalk(int m) {
        // FIXME
        for (int i = 0; i < m; i++) {
            randomMove();
        }
        // END
    }

    /**
     * Private method to generate a random move according to the rules of
the situation.
     * That's to say, moves can be (+-1, 0) or (0, +-1).
     */
    private void randomMove() {
        boolean ns = random.nextBoolean();
        int step = random.nextBoolean() ? 1 : -1;
        move(ns ? step : 0, ns ? 0 : step);
    }

    /**
     * Method to compute the distance from the origin (the lamp-post where
the drunkard starts) to his current position.
     *
     * @return the (Euclidean) distance from the origin to the current
position.
     */
    public double distance() {
        // FIXME by replacing the following code
         return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
        // END
    }

    /**
     * Perform multiple random walk experiments, returning the mean
distance.
     *
     * @param m the number of steps for each experiment
     * @param n the number of experiments to run
     * @return the mean distance
     */
    public static double randomWalkMulti(int m, int n) {
        double totalDistance = 0;
        for (int i = 0; i < n; i++) {
            RandomWalk walk = new RandomWalk();
            walk.randomWalk(m);
            totalDistance = totalDistance + walk.distance();
        }
        return totalDistance / n;
    }

    public static void main(String[] args) {
        if (args.length == 0)
            throw new RuntimeException("Syntax: RandomWalk steps
[experiments]");
        int m = Integer.parseInt(args[0]);
        int n = 30;
        if (args.length > 1) n = Integer.parseInt(args[1]);
```

```
        double meanDistance = randomWalkMulti(m, n);
        System.out.println(m + " steps: " + meanDistance + " over " + n + "
experiments");
    }

}
```

TestPlot

```
package edu.neu.coe.info6205.daim;

import com.opencsv.CSVWriter;
import edu.neu.coe.info6205.randomwalk.RandomWalk;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;

public class testplot {
    static String filePath = "randomWalkPlot.csv";
    static final int totalSteps = 1000;
    static int numSteps;
    static final int numExperiments = 1000;
    static String[] distance = new String[totalSteps];

    public static void main(String[] args) {
        for (int i = 0; i < totalSteps; i++) {
            numSteps = i;
            distance[i] =
String.valueOf(RandomWalk.randomWalkMulti(numSteps, numExperiments));
        }

        writeDataLineByLine(filePath, distance);
    }
    public static void writeDataLineByLine(String filePath, String[]
distance)
    {
        File file = new File(filePath);
        try {
            // create FileWriter object with file as parameter
            FileWriter outputfile = new FileWriter(file);

            // create CSVWriter object filewriter object as parameter

            CSVWriter writer = new CSVWriter(outputfile);

            writer.writeNext(distance);

            // closing writer connection
            writer.close();
        }
        catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```