AMS 326 | FALL 2012 | STONY BROOK UNIVERSITY

Homework 4

Dinia Gepte

11/6/2012

Problem 4.3 Use any method of your choice to generate $N = 10^5$ normally distributed random numbers G_i with given mean $\mu = 0$ and variance $\sigma^2 = 9$.

(1): Sort your G_i numbers generated above in 12 groups $(-\infty, -5], (-5, -4], \dots, (5, \infty)$ and count the number of the G_i numbers in each group and record them as $n_1, n_2, \dots, n_{12}$ respectively.

(2): Use the numbers obtained in Part 1 to form a table with 12 entries (as suggested below)

x	$P(x)$
-5.5	n_1
-4.5	n_2
-3.5	n_3
-2.5	n_4
-1.5	n_5
-0.5	n_6
0.5	n_7
1.5	n_8
2.5	n_9
3.5	n_{10}
4.5	n_{11}
5.5	n_{12}

Next, compute $Q(x) = \ln P(x)$ for the 12 values of $P(x)$. Fit the 12 pairs of x vs. $Q(x)$ to a function

$$Q(x) = c_1 + c_2 x^2$$

- A copy of the source code of the program is provided below. To reduce the length, class and method comments are not shown but are in `Problem3.java`.

```
import java.util.HashMap;

import org.apache.commons.math3.linear.Array2DRowRealMatrix;
import org.apache.commons.math3.linear.LUDecomposition;
import org.apache.commons.math3.linear.RealMatrix;

public class Problem3
{
        public static final int N = 100000;   // GIVEN

        public static void main(String[] args)
        {
                // CREATE A NEW DISTRIBUTION WITH mean = 0 AND variance = 9
                Distribution d = new Distribution(0, 9, N);
                double[] num = d.getValues();

                // USE A HASHTABLE TO COUNT THE NUMBER OF THE G_i NUMBERS IN EACH GROUP
                HashMap<Double, Integer> count = new HashMap<Double, Integer>(12);

                // SET THE KEYS OF THE HASHTABLE ALONG WITH A DEFAULT 0 COUNT
                for (double key = -5.5; key <= 5.5; key++)
                        count.put(key, 0);

                // SORT THE G_i NUMBERS INTO 12 GROUPS
                for (int i = 0; i < num.length; i++)
                {
                        if (num[i] <= -5)
                                count.put(-5.5, count.get(-5.5)+1);
```

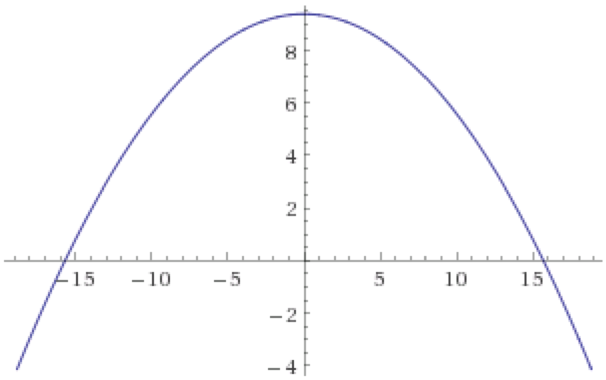
```java
                else if (num[i] <= -4)
                        count.put(-4.5, count.get(-4.5)+1);
                else if (num[i] <= -3)
                        count.put(-3.5, count.get(-3.5)+1);
                else if (num[i] <= -2)
                        count.put(-2.5, count.get(-2.5)+1);
                else if (num[i] <= -1)
                        count.put(-1.5, count.get(-1.5)+1);
                else if (num[i] <= 0)
                        count.put(-0.5, count.get(-0.5)+1);
                else if (num[i] <= 1)
                        count.put(0.5, count.get(0.5)+1);
                else if (num[i] <= 2)
                        count.put(1.5, count.get(1.5)+1);
                else if (num[i] <= 3)
                        count.put(2.5, count.get(2.5)+1);
                else if (num[i] <= 4)
                        count.put(3.5, count.get(3.5)+1);
                else if (num[i] <= 5)
                        count.put(4.5, count.get(4.5)+1);
                else
                        count.put(5.5, count.get(5.5)+1);
        }

        // CALCULATE Q(x) = lnP(x)
        double[] q = new double[12];
        q[0] = functionQ(count.get(-5.5));
        q[1] = functionQ(count.get(-4.5));
        q[2] = functionQ(count.get(-3.5));
        q[3] = functionQ(count.get(-2.5));
        q[4] = functionQ(count.get(-1.5));
        q[5] = functionQ(count.get(-0.5));
        q[6] = functionQ(count.get(0.5));
        q[7] = functionQ(count.get(1.5));
        q[8] = functionQ(count.get(2.5));
        q[9] = functionQ(count.get(3.5));
        q[10] = functionQ(count.get(4.5));
        q[11] = functionQ(count.get(5.5));

        // DISPLAY THE TABLE OF VALUES
        display(count, q);

        // INITIALIZE AND DECLARE THE VALUES OF x TO BE USED IN CURVE FITTING
        double[] x = {-5.5, -4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5};

        // FIT THE GRAPH INTO Q(x) = c1 + c2x^2
        fit(x, q);
}


public static void fit(double[] x, double[] y)
{
        // CALCULATE THE 3x3 MATRIX ON THE LHS
        double[][] a = new double[3][3];
        for (int i = 0; i < x.length; i++)
        {
                a[0][0] = x.length;
                a[0][1] += x[i];
                a[0][2] += x[i]*x[i];
                a[1][0] += x[i];
                a[1][1] += x[i]*x[i];
                a[1][2] += x[i]*x[i]*x[i];
                a[2][0] += x[i]*x[i];
                a[2][1] += x[i]*x[i]*x[i];
                a[2][2] += x[i]*x[i]*x[i]*x[i];
        }

        // CALCULATE THE RHS
        double[] b = new double[3];
        for (int i = 0; i < x.length; i++)
        {
```

```java
                    b[0] += y[i];
                    b[1] += x[i]*y[i];
                    b[2] += x[i]*x[i]*y[i];
            }

            // TRANSFORM THE SQUARE MATRIX INTO A RealMatrix OBJECT
            RealMatrix m = new Array2DRowRealMatrix(a);
            // INVERT m USING LU DECOMPOSITION
            RealMatrix mInverse = new LUDecomposition(m).getSolver().getInverse();
            // MULTIPLY THIS INVERSE WITH b
            double[] coef = mInverse.preMultiply(b);

            System.out.printf("The equation is Q(x) = %.4f + %.4fx^2.", coef[0], coef[2]);
    }

    public static double functionQ(double px)
    {
            return Math.log(px);
    }

    public static void display(HashMap<Double, Integer> map, double[] q)
    {
            System.out.println("  x  | P(x)     |\tQ(x)");
            System.out.println("-5.5 | " + map.get(-5.5) + "\t| " + q[0]);
            System.out.println("-4.5 | " + map.get(-4.5) + "\t| " + q[1]);
            System.out.println("-3.5 | " + map.get(-3.5) + "\t| " + q[2]);
            System.out.println("-2.5 | " + map.get(-2.5) + "\t| " + q[3]);
            System.out.println("-1.5 | " + map.get(-1.5) + "\t| " + q[4]);
            System.out.println("-0.5 | " + map.get(-0.5) + "\t| " + q[5]);
            System.out.println(" 0.5 | " + map.get(0.5) + "\t| " + q[6]);
            System.out.println(" 1.5 | " + map.get(1.5) + "\t| " + q[7]);
            System.out.println(" 2.5 | " + map.get(2.5) + "\t| " + q[8]);
            System.out.println(" 3.5 | " + map.get(3.5) + "\t| " + q[9]);
            System.out.println(" 4.5 | " + map.get(4.5) + "\t| " + q[10]);
            System.out.println(" 5.5 | " + map.get(5.5) + "\t| " + q[11]);
    }
}
```

- When run, the program will produce the following result:

```
  x  | P(x) |     Q(x)
-5.5 | 4670 | 8.448914350662942
-4.5 | 4232 | 8.350429973538136
-3.5 | 6774 | 8.820847033377717
-2.5 | 9426 | 9.151227107483676
-1.5 | 11803| 9.37610901542907
-0.5 | 13162| 9.485089169016064
 0.5 | 12911| 9.46583494017472
 1.5 | 11792| 9.375176614429117
 2.5 | 9447 | 9.153452509769135
 3.5 | 6711 | 8.811503250158239
 4.5 | 4361 | 8.380456667842767
 5.5 | 4711 | 8.45765547870004
The equation is Q(x) = 9.3935 + -0.0381x^2.
```

- The graph of the equation is shown below:



- This program uses Apache's math library with linear algebra functions found in http://commons.apache.org/math/download_math.cgi. It was used to fit the points in the equation Q(x). The program also uses the `Distribution` class which I've created to represent normally distributed n values with the given mean and variance. (So, to create a standard normal distribution, the mean should be 0 and variance 1). I used a hash table to bucket sort the numbers generated by my Distribution class. Fitting the curve into the equation required the use of matrices to implement the least square method.

Problem 4.4 Mr. R. N. Stupid bought $20,000 worth of LousyCompany.com's stocks precisely one year ago. Assume he has gone through 52 weeks of trading with exactly five trading days each. Therefore, he has gone through 260 trading days. The change rate of Mr. Stupid's stock from the previous day for the entire year follows a Normal Distribution with values with mean $\mu = -1\%$ and variance $\sigma^2 = 1.04\%$. The Normal distribution formula is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(x-\mu)^2}{2o^2}}$$

Please do the following:

(1). Compute Mr. Stupid's stock value at the end of each of the 260 trading days. Show the results in a table and a graph;
(2). Do the same as in (1) if the stock varies from the previous day with the same variance as in (1) and μ = -2%;
(3). Do the same as in (1) if the stock changes from the previous day with the same variance as in (1) and $\mu = 1\%$;
(4). Show the results in a table and a graph. The table may look like the following and putting three curves in one graph is a neat idea:

Day	Stock Value for Scheme 1	Stock Value for Scheme 2	Stock Value for Scheme 3
1			
2			
…			
260			

(5) Repeat (1)-(4) above if the variance is $\sigma^2 = 0.114\%$.

- The source code of the driver class is below and can be found in `Problem4.java`.

```java
/**
 * This program prints 260 stock-day values in different schemes. Each scheme
 * varies in mean and variance. The stock values are determined using a
 * normally distributed stock exchange rate.
 *
 * @author Dinia Gepte
 * Homework 4, AMS 326, Fall 2012
 */
public class Problem4
{
	public static final int N = 260;	// NUMBER OF DAYS
	public static final int initStock = 20000;	// INITIAL STOCK

	public static void main(String[] args)
	{
		double[] scheme1 = scheme(-0.01, 0.0104);
		double[] scheme2 = scheme(-0.02, 0.0104);
		double[] scheme3 = scheme(0.01, 0.0104);
		double[] scheme4 = scheme(-0.01, 0.0114);
		double[] scheme5 = scheme(-0.02, 0.0114);
		double[] scheme6 = scheme(0.01, 0.0114);
		System.out.println("Day | Scheme 1 | Scheme 2 | Scheme 3 | Scheme 4 | Scheme 5 | 
Scheme 6");
		for (int i = 1; i < scheme1.length; i++)
			System.out.printf("%d   | %.2f | %.2f | %.2f | %.2f | %.2f | %.2f\n", i,
					scheme1[i], scheme2[i], scheme3[i],
					scheme4[i], scheme5[i], scheme6[i]);
	}

	/**
	 * Returns a listing of N stock values following a normal distribution
	 * with the given mean and variance.
	 * @param mean - mean of the normal distribution
	 * @param var - variance of the normal distribution
	 * @return the listing of N stock values
	 */
	public static double[] scheme(double mean, double var)
	{
		// CREATE A Distribution CLASS WHICH CREATES 260 NORMALLY DISTRIBUTED VALUES
		Distribution d = new Distribution(mean, var, N);

		// RATE PER DAY
		double[] rate = new double[N+1];
		rate[0] = 1;	// INITIAL RATE
		for (int i = 1; i < rate.length; i++)
			rate[i] = rate[i-1] * (1 + d.getGaussRanVar());

		// STOCK VALUE PER DAY
		double[] stockValue = new double[N+1];
		stockValue[0] = initStock;	// INITIAL STOCK VALUE
		for (int i = 1; i < stockValue.length; i++)
			stockValue[i] = stockValue[i-1]*Math.pow(1+rate[i], 1.0/N);

		return stockValue;
	}
}
```

- A sample output of the program is displayed below where each scheme is based on when they appeared in the problem.

```
Day | Scheme 1 | Scheme 2 | Scheme 3 | Scheme 4 | Scheme 5 | Scheme 6
1   | 20050.26 | 20052.47 | 20056.38 | 20060.49 | 20050.97 | 20054.27
2   | 20097.82 | 20101.07 | 20108.05 | 20127.94 | 20103.68 | 20106.25
3   | 20148.15 | 20148.19 | 20158.44 | 20191.30 | 20160.97 | 20162.52
```

4	20200.45	20192.76	20206.37	20248.49	20220.38	20215.79
5	20256.16	20240.79	20260.39	20305.59	20280.10	20265.62
6	20316.13	20293.39	20317.33	20360.56	20343.49	20315.77
7	20384.92	20346.89	20382.08	20406.39	20414.58	20369.23
8	20451.42	20403.78	20443.25	20461.50	20475.04	20418.02
9	20518.95	20467.90	20495.54	20516.03	20538.45	20470.61
10	20586.23	20527.01	20549.21	20574.53	20604.34	20522.03
11	20653.03	20586.15	20602.40	20631.94	20660.06	20574.51
12	20717.04	20640.44	20657.77	20696.91	20710.62	20630.83
13	20781.42	20696.64	20717.03	20756.45	20764.68	20689.99
14	20846.02	20755.07	20779.34	20821.57	20812.30	20751.98
15	20912.15	20811.81	20847.07	20881.08	20860.99	20820.35
16	20977.03	20872.92	20912.27	20935.42	20902.93	20888.82
17	21033.44	20932.58	20984.96	20983.68	20946.37	20961.45
18	21088.88	20990.81	21057.58	21030.85	20984.75	21030.37
19	21144.35	21048.32	21136.68	21079.25	21018.00	21099.91
20	21200.70	21102.08	21214.60	21123.96	21047.46	21167.55
21	21247.44	21156.82	21299.44	21171.87	21079.38	21227.75
22	21293.57	21209.77	21385.98	21219.49	21109.36	21295.14
23	21340.36	21267.21	21459.42	21271.78	21138.18	21361.17
24	21378.87	21325.14	21531.38	21332.04	21168.22	21429.35
25	21412.04	21375.64	21601.02	21392.27	21198.28	21495.15
26	21446.19	21418.83	21670.73	21445.47	21229.48	21552.06
27	21483.73	21456.89	21731.31	21493.76	21262.58	21609.82
28	21518.70	21491.66	21797.47	21540.08	21297.84	21671.11
29	21554.19	21530.52	21868.48	21588.68	21330.84	21733.34
30	21588.98	21567.08	21933.94	21634.73	21365.21	21803.90
31	21623.39	21606.84	21996.16	21685.98	21404.48	21885.67
32	21654.85	21653.31	22063.99	21749.15	21445.37	21978.47
33	21684.69	21700.23	22130.64	21815.95	21481.32	22075.98
34	21715.35	21745.61	22198.69	21887.28	21510.75	22178.81
35	21742.17	21786.54	22270.90	21963.95	21536.31	22279.32
36	21773.00	21830.76	22349.81	22042.27	21563.44	22372.35
37	21808.63	21870.19	22430.95	22110.59	21584.48	22463.92
38	21846.70	21915.13	22511.15	22175.19	21600.62	22551.35
39	21884.07	21958.61	22578.59	22239.70	21617.19	22643.59
40	21916.57	21995.15	22649.68	22296.94	21634.01	22725.75
41	21947.12	22032.34	22721.27	22352.78	21651.26	22810.57
42	21982.38	22067.10	22796.88	22410.42	21670.21	22891.99
43	22019.39	22102.48	22876.62	22471.12	21689.91	22974.14
44	22059.05	22136.57	22954.53	22523.12	21711.12	23059.75
45	22095.88	22170.07	23025.17	22565.17	21731.95	23141.68
46	22133.31	22202.94	23091.84	22609.19	21749.17	23229.04
47	22166.71	22238.38	23161.64	22653.01	21765.37	23319.89
48	22201.09	22272.69	23231.71	22695.68	21784.76	23417.95
49	22237.30	22304.81	23296.91	22735.33	21805.54	23521.51
50	22273.48	22337.94	23367.37	22772.03	21826.31	23626.29
51	22308.91	22369.80	23434.78	22801.81	21845.83	23736.31
52	22344.42	22400.84	23503.18	22832.18	21863.37	23845.66
53	22377.77	22427.66	23568.17	22860.96	21882.04	23962.14
54	22410.49	22454.69	23631.05	22885.01	21901.22	24071.26
55	22446.51	22479.05	23688.64	22905.63	21922.11	24181.83
56	22483.60	22504.40	23749.28	22928.31	21942.24	24299.00
57	22522.42	22528.38	23803.23	22954.48	21962.65	24417.09
58	22566.37	22551.46	23857.77	22985.07	21980.12	24521.72
59	22604.76	22576.03	23913.25	23015.22	21994.26	24623.95
60	22645.74	22599.02	23967.32	23044.17	22010.74	24727.65

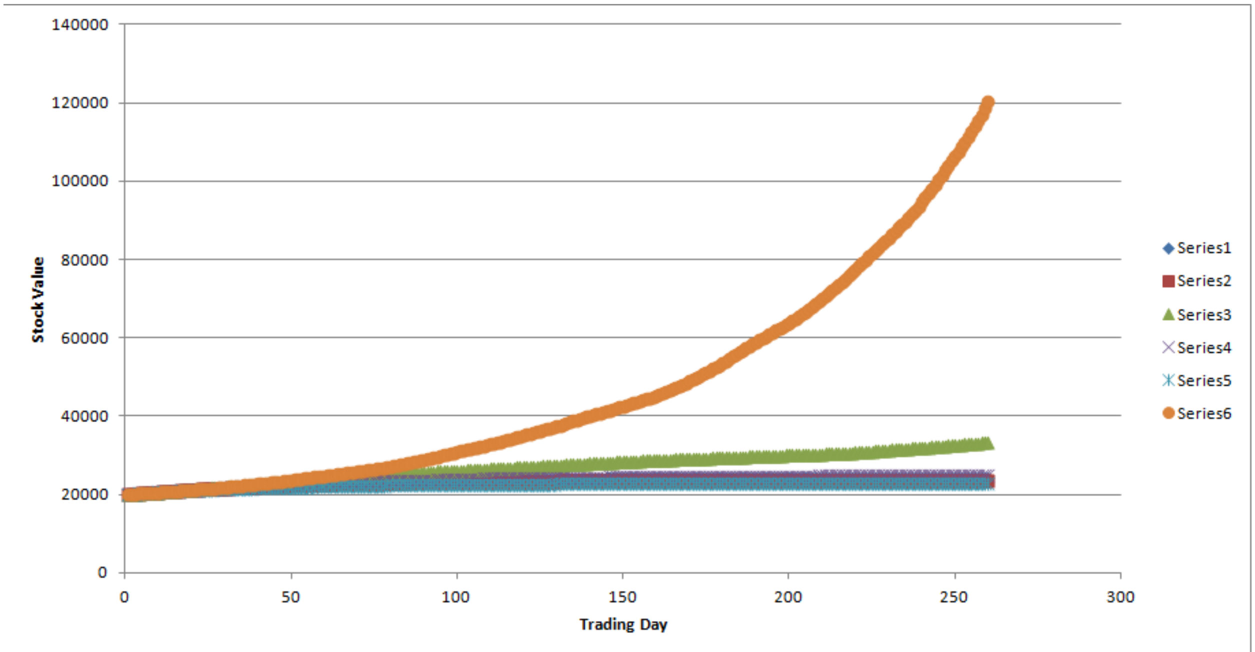
61	22684.34	22620.48	24028.07	23072.28	22026.61	24836.61
62	22716.03	22644.67	24091.30	23097.99	22043.85	24955.98
63	22748.27	22669.40	24157.58	23122.84	22060.34	25064.38
64	22779.79	22697.74	24218.30	23148.60	22078.36	25169.46
65	22808.24	22725.41	24280.13	23170.22	22098.03	25280.70
66	22839.19	22753.38	24339.34	23192.03	22117.24	25378.38
67	22870.01	22781.62	24402.32	23215.39	22135.31	25478.65
68	22903.20	22805.58	24466.97	23234.10	22155.08	25579.15
69	22932.91	22825.44	24524.36	23252.84	22172.41	25688.54
70	22961.48	22843.84	24582.09	23273.00	22188.32	25811.84
71	22990.93	22859.31	24648.24	23293.87	22204.42	25929.10
72	23023.16	22877.24	24713.19	23309.86	22219.29	26052.46
73	23054.73	22892.25	24781.65	23326.70	22233.69	26183.60
74	23089.12	22905.67	24848.07	23343.27	22246.42	26314.39
75	23123.67	22916.65	24915.36	23360.27	22258.78	26442.48
76	23156.07	22927.23	24984.51	23373.79	22270.35	26578.85
77	23190.54	22937.73	25051.12	23388.42	22283.44	26715.64
78	23220.85	22948.13	25113.99	23406.17	22297.55	26852.11
79	23254.64	22956.96	25175.37	23422.76	22311.72	26994.30
80	23289.10	22966.07	25233.19	23437.70	22325.26	27137.49
81	23321.36	22975.27	25288.00	23452.21	22337.47	27291.06
82	23347.40	22983.87	25343.92	23466.62	22350.18	27463.28
83	23371.40	22991.83	25386.39	23481.08	22362.79	27647.50
84	23387.95	22999.37	25432.18	23498.04	22374.22	27829.90
85	23403.34	23005.46	25475.18	23517.95	22384.45	28012.12
86	23416.93	23012.56	25514.96	23536.40	22394.01	28189.70
87	23430.53	23018.88	25556.51	23560.22	22401.82	28361.10
88	23444.56	23025.27	25597.53	23583.16	22408.90	28536.49
89	23459.27	23032.23	25630.53	23605.43	22417.45	28711.34
90	23471.67	23038.57	25665.05	23628.04	22425.96	28882.28
91	23483.65	23045.47	25700.25	23648.13	22434.21	29053.64
92	23496.35	23052.61	25737.54	23670.89	22443.24	29233.47
93	23509.55	23058.77	25775.70	23696.52	22450.40	29415.21
94	23522.77	23064.50	25815.25	23717.42	22456.57	29612.71
95	23535.72	23070.61	25855.11	23738.84	22463.54	29804.47
96	23550.14	23077.45	25894.02	23762.02	22470.91	30009.42
97	23564.57	23083.21	25932.04	23785.78	22478.72	30225.47
98	23575.98	23089.71	25969.60	23812.16	22486.50	30436.07
99	23587.36	23097.01	26009.58	23837.75	22495.02	30651.13
100	23600.25	23104.04	26050.78	23860.91	22503.88	30860.74
101	23613.86	23111.65	26096.58	23889.65	22511.70	31066.33
102	23628.51	23118.42	26144.98	23918.12	22518.97	31259.98
103	23642.99	23124.58	26192.03	23949.15	22526.37	31449.96
104	23658.96	23131.31	26236.55	23979.28	22532.98	31639.65
105	23674.44	23138.22	26286.73	24004.97	22538.35	31818.34
106	23688.78	23145.77	26340.37	24025.91	22543.87	31994.78
107	23703.54	23153.78	26391.02	24046.71	22549.46	32189.32
108	23717.83	23162.76	26443.41	24067.61	22554.24	32389.29
109	23731.09	23171.11	26482.92	24087.85	22559.49	32596.39
110	23745.58	23179.73	26520.99	24105.94	22564.50	32810.74
111	23760.33	23188.44	26563.70	24122.07	22569.72	33016.29
112	23773.94	23198.23	26608.01	24138.66	22574.76	33225.45
113	23788.51	23208.10	26650.27	24150.49	22579.90	33433.94
114	23801.99	23217.07	26694.96	24162.69	22584.47	33667.75
115	23815.48	23224.75	26747.65	24173.93	22589.66	33922.88
116	23825.77	23232.59	26800.68	24181.87	22594.90	34169.92
117	23834.87	23240.44	26849.87	24189.00	22600.90	34412.31

118	23842.50	23249.11	26893.54	24197.57	22606.10	34654.36
119	23849.92	23258.37	26935.34	24207.63	22610.91	34872.66
120	23857.09	23268.68	26978.78	24218.42	22616.43	35073.48
121	23864.62	23279.00	27022.49	24228.85	22621.30	35291.32
122	23872.85	23291.09	27062.51	24239.54	22626.25	35514.75
123	23881.19	23302.35	27104.34	24248.67	22631.32	35759.63
124	23888.92	23311.51	27147.15	24258.07	22635.94	35992.07
125	23896.38	23320.00	27192.95	24267.31	22640.75	36239.90
126	23904.10	23327.88	27238.16	24276.33	22645.11	36482.86
127	23912.77	23336.35	27279.05	24286.76	22648.74	36732.76
128	23921.82	23344.27	27328.88	24296.58	22652.51	36977.60
129	23930.93	23350.70	27385.95	24304.32	22655.90	37207.10
130	23940.47	23356.20	27441.68	24312.79	22659.40	37437.02
131	23949.71	23360.94	27495.65	24322.87	22663.04	37689.29
132	23957.98	23365.02	27539.23	24331.78	22666.66	37958.15
133	23965.56	23369.13	27589.80	24338.76	22670.07	38219.84
134	23972.84	23373.40	27630.50	24345.13	22673.44	38492.70
135	23980.65	23378.12	27672.42	24351.10	22677.08	38762.97
136	23988.79	23382.66	27719.95	24356.74	22680.89	39030.92
137	23996.70	23387.25	27764.46	24362.30	22684.36	39301.18
138	24005.59	23391.72	27814.00	24368.40	22688.10	39575.40
139	24013.10	23396.14	27859.86	24374.23	22692.15	39843.51
140	24021.25	23400.13	27908.89	24379.55	22696.41	40089.70
141	24030.39	23403.80	27961.56	24384.63	22699.85	40341.83
142	24038.52	23407.13	28010.44	24389.25	22702.92	40588.57
143	24046.96	23410.21	28060.78	24394.07	22705.66	40810.14
144	24054.68	23413.55	28109.90	24399.11	22708.26	41043.96
145	24063.13	23416.52	28159.31	24403.55	22711.07	41284.41
146	24071.32	23419.22	28203.96	24407.76	22713.69	41528.93
147	24080.34	23422.11	28249.41	24411.86	22716.44	41776.95
148	24088.72	23425.34	28295.91	24415.97	22719.22	42036.39
149	24097.41	23428.63	28343.24	24420.55	22722.05	42293.23
150	24105.17	23432.49	28388.19	24424.28	22724.75	42555.66
151	24113.19	23436.92	28432.66	24428.68	22727.21	42793.74
152	24121.62	23440.85	28472.74	24433.39	22729.56	43043.87
153	24130.19	23444.85	28511.80	24438.61	22731.83	43304.09
154	24137.68	23448.40	28550.45	24443.85	22733.97	43569.91
155	24144.08	23451.88	28588.99	24449.33	22735.81	43829.58
156	24150.73	23455.73	28624.95	24454.01	22737.82	44099.53
157	24157.66	23459.87	28665.40	24458.61	22739.85	44380.05
158	24164.38	23463.66	28706.58	24463.39	22741.37	44680.03
159	24169.83	23467.52	28746.41	24468.10	22742.73	44993.95
160	24175.80	23471.16	28786.20	24473.44	22744.22	45315.72
161	24182.22	23474.60	28820.58	24479.19	22745.86	45647.27
162	24187.37	23478.04	28851.42	24486.11	22747.16	45988.45
163	24191.39	23481.89	28880.19	24492.97	22748.40	46330.11
164	24195.49	23485.57	28909.19	24498.78	22749.60	46664.67
165	24199.49	23489.32	28938.02	24504.30	22750.90	47016.13
166	24203.38	23493.12	28967.80	24509.81	22752.15	47365.59
167	24206.56	23497.33	29001.86	24515.16	22753.79	47733.35
168	24209.39	23501.22	29031.89	24520.62	22755.02	48110.15
169	24212.27	23504.54	29067.00	24526.51	22756.33	48503.61
170	24215.69	23507.36	29096.72	24532.30	22757.55	48904.33
171	24218.89	23509.88	29127.68	24538.47	22758.75	49330.90
172	24222.22	23511.92	29160.87	24544.81	22759.92	49766.39
173	24225.14	23514.06	29192.88	24552.27	22761.11	50219.23
174	24228.00	23516.25	29221.30	24560.77	22762.42	50666.79

175	24230.68	23518.32	29250.41	24568.77	22763.76	51132.75
176	24233.68	23520.03	29278.74	24576.50	22765.10	51608.47
177	24236.48	23521.79	29306.59	24583.75	22766.58	52056.69
178	24238.99	23523.46	29335.35	24590.75	22767.85	52547.86
179	24241.40	23525.33	29365.94	24598.75	22769.46	53052.56
180	24243.50	23527.19	29395.63	24606.53	22771.03	53570.16
181	24245.54	23528.96	29426.88	24613.75	22772.55	54109.44
182	24247.54	23530.92	29457.79	24621.19	22774.11	54638.60
183	24249.63	23532.56	29487.85	24628.40	22775.72	55169.15
184	24251.84	23534.34	29521.42	24635.69	22777.50	55723.44
185	24253.63	23535.90	29556.07	24641.62	22779.18	56279.79
186	24255.38	23537.31	29589.47	24647.71	22781.03	56798.75
187	24256.80	23538.75	29622.27	24653.07	22782.79	57323.13
188	24258.24	23540.09	29657.22	24657.58	22784.09	57849.31
189	24259.52	23541.57	29694.24	24661.64	22785.19	58343.90
190	24260.70	23543.01	29730.74	24665.86	22786.20	58867.10
191	24261.90	23544.19	29763.85	24669.59	22787.25	59421.56
192	24263.14	23545.57	29799.80	24673.12	22788.33	59933.22
193	24264.60	23546.75	29832.13	24676.75	22789.38	60408.96
194	24266.07	23547.91	29860.24	24680.18	22790.33	60899.25
195	24267.62	23548.91	29888.00	24684.72	22791.35	61355.35
196	24269.01	23550.01	29916.43	24689.03	22792.17	61822.99
197	24270.36	23551.19	29945.93	24693.18	22792.91	62306.16
198	24271.87	23552.24	29975.13	24697.92	22793.62	62767.13
199	24273.18	23553.19	30002.58	24702.99	22794.26	63252.12
200	24274.28	23554.06	30027.10	24707.23	22794.96	63756.87
201	24275.44	23554.93	30052.04	24712.27	22795.65	64282.02
202	24276.68	23555.74	30077.70	24718.10	22796.25	64855.20
203	24277.83	23556.38	30109.27	24723.84	22796.97	65440.14
204	24279.03	23557.15	30140.42	24728.97	22797.77	66026.34
205	24280.22	23557.92	30166.41	24734.16	22798.60	66633.69
206	24281.36	23558.72	30196.96	24739.19	22799.36	67224.50
207	24282.50	23559.49	30227.66	24744.86	22800.10	67844.52
208	24283.64	23560.29	30261.50	24751.32	22800.77	68513.46
209	24284.88	23561.19	30294.39	24757.58	22801.39	69192.96
210	24286.03	23562.10	30326.87	24763.76	22801.91	69879.97
211	24286.89	23563.02	30360.62	24769.41	22802.42	70583.50
212	24287.53	23563.86	30393.66	24775.32	22802.90	71317.66
213	24288.18	23564.79	30425.58	24780.97	22803.31	72041.46
214	24288.81	23565.80	30459.73	24787.12	22803.70	72789.28
215	24289.39	23566.69	30495.48	24792.53	22804.07	73555.84
216	24289.87	23567.55	30534.80	24798.43	22804.49	74298.76
217	24290.28	23568.52	30578.44	24804.07	22804.87	75058.71
218	24290.61	23569.42	30620.11	24809.30	22805.22	75830.22
219	24290.89	23570.25	30666.29	24814.89	22805.68	76618.93
220	24291.14	23571.12	30713.91	24821.46	22806.09	77437.42
221	24291.39	23571.97	30768.22	24828.23	22806.43	78280.98
222	24291.62	23572.78	30821.51	24835.14	22806.73	79130.72
223	24291.86	23573.56	30876.39	24841.74	22807.04	79969.40
224	24292.06	23574.31	30929.73	24848.02	22807.36	80774.55
225	24292.27	23575.32	30989.36	24854.33	22807.69	81588.24
226	24292.47	23576.20	31056.72	24859.54	22808.06	82360.15
227	24292.67	23577.03	31115.38	24864.57	22808.39	83090.99
228	24292.88	23577.89	31185.75	24869.60	22808.74	83853.53
229	24293.10	23578.58	31262.82	24874.28	22809.05	84648.98
230	24293.29	23579.19	31328.99	24878.32	22809.38	85447.46
231	24293.47	23579.80	31399.44	24883.36	22809.68	86301.42

232	24293.65	23580.47	31469.29	24889.06	22809.94	87173.33
233	24293.82	23581.09	31526.35	24894.41	22810.17	88029.67
234	24294.00	23581.75	31585.09	24900.61	22810.40	88854.39
235	24294.16	23582.48	31643.99	24906.50	22810.61	89706.58
236	24294.32	23583.17	31703.89	24913.13	22810.82	90584.92
237	24294.50	23583.75	31765.33	24918.65	22811.01	91523.10
238	24294.67	23584.32	31833.46	24923.28	22811.23	92538.48
239	24294.81	23584.86	31897.62	24927.69	22811.47	93574.89
240	24294.95	23585.41	31954.04	24932.46	22811.73	94701.19
241	24295.08	23585.94	32013.67	24936.41	22812.00	95820.52
242	24295.23	23586.47	32070.24	24941.33	22812.23	96919.60
243	24295.35	23587.00	32127.13	24946.47	22812.45	98041.39
244	24295.47	23587.56	32181.05	24951.40	22812.64	99206.52
245	24295.56	23588.04	32242.77	24955.78	22812.85	100327.81
246	24295.66	23588.47	32309.52	24959.94	22813.07	101510.32
247	24295.76	23588.99	32377.62	24963.85	22813.27	102762.84
248	24295.85	23589.46	32455.79	24967.22	22813.45	104000.12
249	24295.95	23589.92	32534.22	24970.82	22813.64	105182.80
250	24296.04	23590.36	32611.90	24974.44	22813.82	106334.92
251	24296.12	23590.82	32685.19	24978.05	22813.98	107517.40
252	24296.21	23591.21	32754.27	24981.04	22814.11	108707.37
253	24296.31	23591.59	32823.91	24984.19	22814.24	109933.58
254	24296.42	23591.97	32896.51	24987.19	22814.37	111190.85
255	24296.54	23592.35	32973.57	24990.11	22814.50	112510.57
256	24296.67	23592.69	33041.24	24992.85	22814.63	113937.88
257	24296.78	23592.99	33114.43	24995.68	22814.80	115422.07
258	24296.90	23593.26	33180.75	24998.16	22814.95	116968.48
259	24297.03	23593.54	33245.73	25000.52	22815.09	118543.17
260	24297.17	23593.84	33305.64	25002.62	22815.23	120184.40

- The plot of the data points above is shown below. MS Excel was used to graph the points. The legend corresponds to the schemes (e.g. Series1 = Scheme1). Due to the relatively big number of points, the points themselves display the trend line of the graph. Series 1 to 5 may seem like straight lines but in a closer inspection, they follow a logarithmic pattern because of the distribution function.



- Multiple runs of the program generate different numerical results because the rate in each day depends on the randomly generated z-value. Also, this program uses the `Distribution` class I created specifically for this homework.
- The `Distribution` class generates N normally distributed values with the given mean and variance that are accepted during initialization. I found this to be the simplest way to be able to have distribution values with different means and variances. The class also has a method `getGaussRanVar()` which selects a random value in the distribution. This was used for the next day stock rate exchange.