1.1

1.1 Random Binary Expansions

Question 1

We would like to determine $\hat{F}(x)$ to a certain accuracy with a certain confidence. Let's say that we want it to be correct to 2 d.p. (i.e. error of ± 0.005) with confidence 95%.

The random variable $1[X^n \leq x]$ is Bernoulli$\left(\hat{F}(x)\right)$, so it has mean $\hat{F}(x)$ and variance $\hat{F}(x)\left(1-\hat{F}(x)\right)$. By Central Limit Theorem, for $Z \sim N(0,1)$ we have $\frac{\frac{1}{N}\sum_{j=1}^{N} 1\left[X_j^n \leq x\right] - \hat{F}(x)}{\sqrt{\frac{\hat{F}(x)\left(1-\hat{F}(x)\right)}{N}}} \to Z$ in distribution for large N.

For the specified accuracy and confidence, we want $P\left(\left|\frac{1}{N}\sum_{j=1}^{N} 1\left[X_j^n \leq x\right] - \hat{F}(x)\right| \leq 0.0005\right) \geq 0.95$, so by property of Normal Distribution we have $\frac{0.005}{\sqrt{\frac{\hat{F}(x)\left(1-\hat{F}(x)\right)}{N}}} \geq 2\Phi^{-1}(0.975) - 1 \approx 1.96$. As we want this to be true $\forall\, x$, we have $\frac{0.005}{\sqrt{\frac{1}{4N}}} = \frac{0.005}{\sup\sqrt{\frac{\hat{F}(x)\left(1-\hat{F}(x)\right)}{N}}} \geq 1.96$, which gives $N \geq 38416$.

The programming class Q1 estimates $\hat{F}(x)$ at $x = \frac{k}{2048}$ for each $k \in \{0, 1, \ldots, 2048\}$ by counting the number of sample elements $\leq \frac{k}{2048}$ and then dividing by N. The graph of $\hat{F}$ is plotted as below (using the libraries JCommon and JFreeChart).

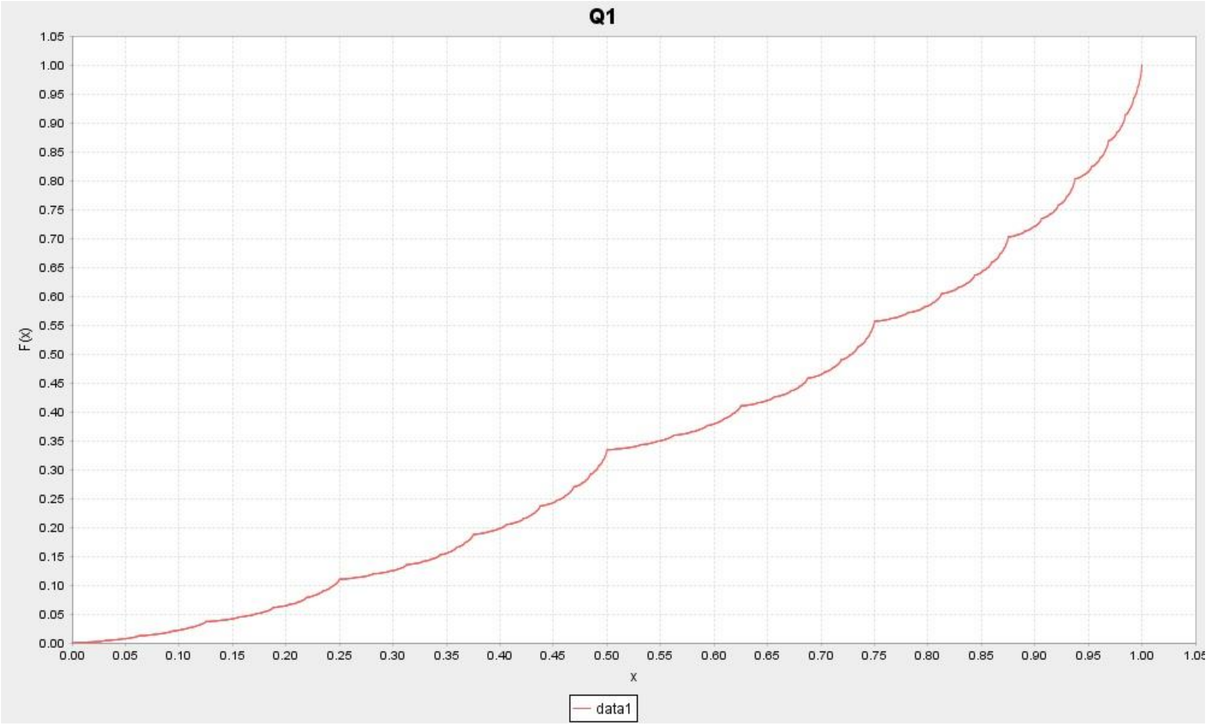

Figure 1.1: Graph of $\hat{F}$ for $p = \frac{2}{3}$ and $n = 30$

Question 2

For $X = \sum_{i=1}^{\infty} \frac{U_i}{2^i}$, we have $X \leq x$ iff either (1) $\exists\, j$ s.t. $U_j = 0$, $x_j = 1$ and $U_i = x_i \; \forall\, i < j$, or (2) $U_i = x_i \; \forall\, i$.

Let $f_x(j)$ be the number of $i < j$ s.t. $x_i = 1$.
Then case (1) has probability $\sum_{j:\, x_j = 1} p^{f_x(j)} (1-p)^{j - f_x(j)}$, and case (2) has probability $\leq \lim_{N \to \infty} (1-p)^{N-n} = 0$. Therefore $F(x) = \sum_{j:\, x_j = 1} p^{f_x(j)} (1-p)^{j - f_x(j)}$.

Question 3

The class Q3 calculates $F(x)$ at each $x = \frac{k}{2048}$ using the formula from Question 2. The graph is plotted as below.

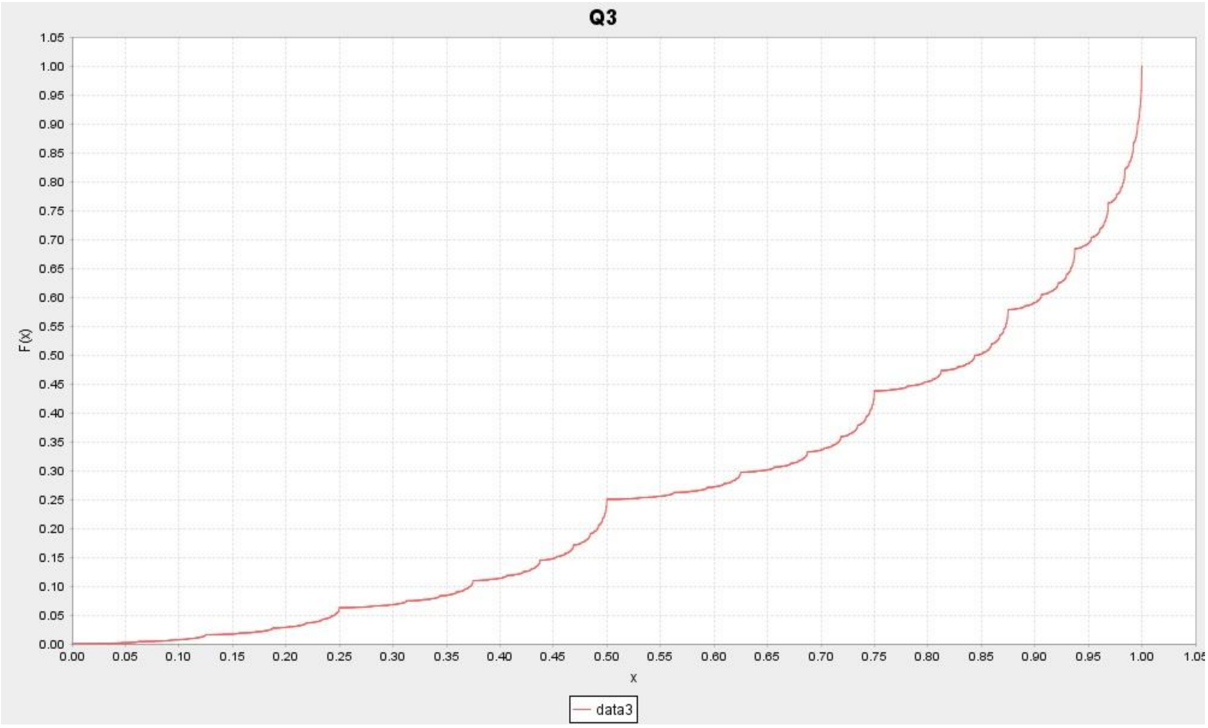

Figure 3.1: Graph of $F(x)$ for $p = \frac{3}{4}$ and $n = 11$

And the corresponding graph from Question 1 is as below.

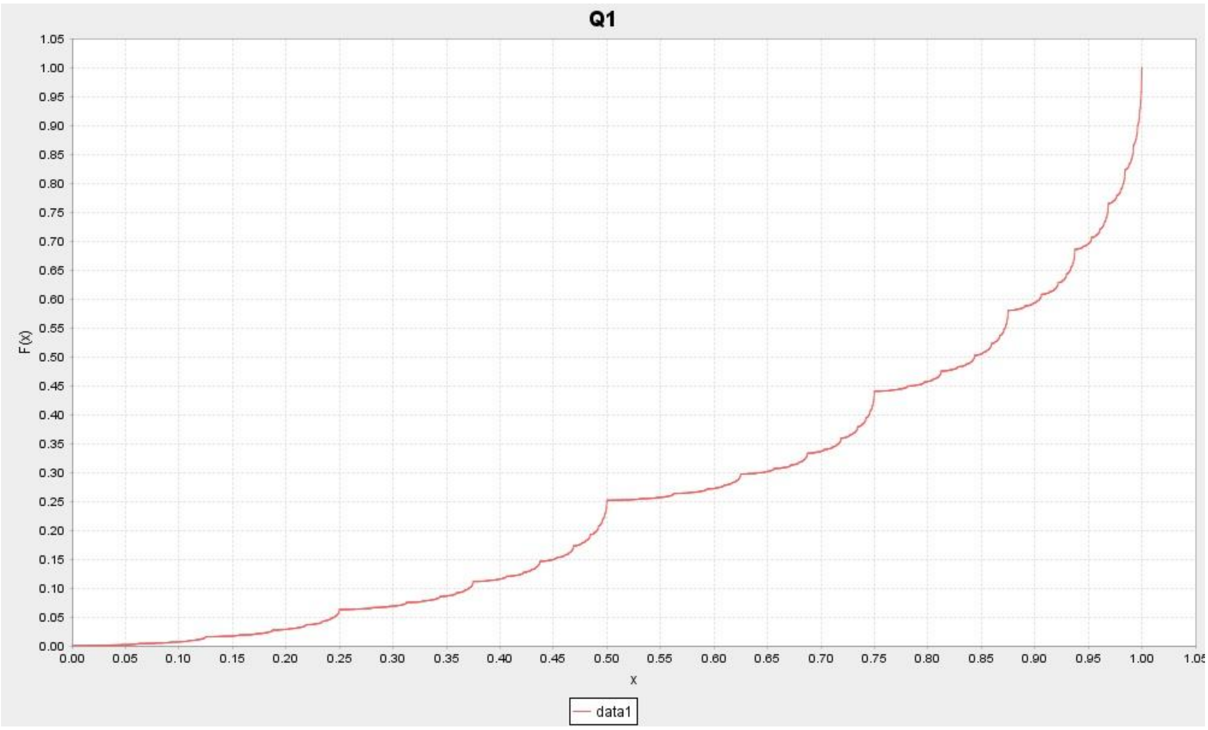

Figure 3.2: Graph of $\hat{F}(x)$ for $p = \frac{3}{4}$ and $n = 30$

The shape of the graph looks identical to that generated by Q1.

Q3 also finds the error of $\hat{F}$ generated from Question 1, for which comparison is possible as both Q1 and Q3 plot at the same x. The table below shows the maximal error, the average error, and the number of times error exceeds 0.005.

Trial	Maximal error	Average error	# Bad estimations
1	0.003958923625032473	0.0011499718735567843	0
2	0.0021961452264877	6.944365162155503E-4	0
3	0.0029702144878598347	6.839723064895164E-4	0
4	0.0034977211052554247	7.836971878788185E-4	0
5	0.005408436295788577	0.0020820801348143576	16

Table 3.1: Comparison of plots generated by Q1 and Q3, repeated 4 times

This is consistent with the tolerated error (0.005 with 95%) we established in Question 1.

We compare the complexities of Q1 and Q3. Suppose we plot at $x = \frac{k}{2^n}$ in both cases.

Q1 requires flipping a coin $n \times N$ times, computing X, sorting the results, and then counting the required number of elements by going through the list again. The complexity is $O(nN) + O(N) + O(N \log N) + O(N + 2^n) = O(nN + N \log N + 2^n)$. Assuming we fix our required accuracy (thus fixing N), the complexity is $O(2^n)$.

Q3 requires computing $p^{f(j)}(1-p)^{j-f(j)}$ for each digit 1 appearing in $x = \frac{k}{2^n}$. The total number of 1's appearing in x across all $k < 2^n$ is $\frac{n \times 2^n}{2} = n \times 2^{n-1}$, so Q3 has complexity of $O(n \times 2^n)$.

Q3 is more time consuming than Q1 in exchange for accuracy.

Question 4

Let $c = \sum_{i=1}^{n} \frac{c_i}{2^i}$ where $c_i \in \{0,1\}$ and $c_n = 1$. Consider $c \pm \frac{1}{2^s}$ where $s > n$.
Using the function f as in Question 2,

$$F\left(c + \frac{1}{2^s}\right) - F(c) = p^{f_{c+\frac{1}{2^s}}(s)}(1-p)^{s-f_{c+\frac{1}{2^s}}(s)} \leq \max\{p, 1-p\}^s \to 0, \text{ and}$$

$$\begin{aligned} F(c) - F\left(c - \frac{1}{2^s}\right) &= p^{f_c(n)}(1-p)^{n-f_c(n)} - \sum_{j=n+1}^{s} p^{f_{c-\frac{1}{2^s}}(j)}(1-p)^{j-f_{c-\frac{1}{2^s}}(j)} \\ &= p^{f_c(n)}(1-p)^{n-f_c(n)} - \sum_{j=n+1}^{s} p^{f_c(n)+j-n-1}(1-p)^{j-f_c(n)-j+n+1} \\ &= p^{f_c(n)}(1-p)^{n-f_c(n)}\left(1 - \sum_{j=0}^{s-n-1} p^j(1-p)\right) \\ &= p^{f_c(n)}(1-p)^{n-f_c(n)}p^{s-n} \to 0 \end{aligned}$$

As F is increasing and $\frac{1}{2^s} \to 0$, the above ensures that $x \to c \implies F(x) \to F(c)$, so F is continuous at c.

Now for any x_0 without a finite binary expansion, we can find integer $0 \le a_s < 2^s$ s.t. $\frac{a_s}{2^s} < x_0 < \frac{a_s}{2^s} + \frac{1}{2^s}$. Then $\lim_{s\to\infty} \frac{a_s}{2^s} = \lim_{s\to\infty} \left(\frac{a_s}{2^s} + \frac{1}{2^s}\right) = x$.
As $\frac{a_s}{2^s}$ has a finite binary expansion, by above $F\left(\frac{a_s}{2^s} + \frac{1}{2^s}\right) - F\left(\frac{a_s}{2^s}\right) \to 0$. Hence F increasing ensures $x \to x_0 \implies F(x) \to F(x_0)$, , so F is continuous at x_0.

Question 5

For simplicity we use $|\delta| < \frac{1}{16}$. The class Q5 plot $\frac{F(c+\delta)-F(c)}{\delta}$ against δ for $\delta = \pm\frac{k}{2^{20}}$ where $1 \le k \le 2^{16}$. The choice ensures δ sufficiently close to 0 and has a good range of numbers of digit 1s.

The graph for $\delta > 0$ is as below.

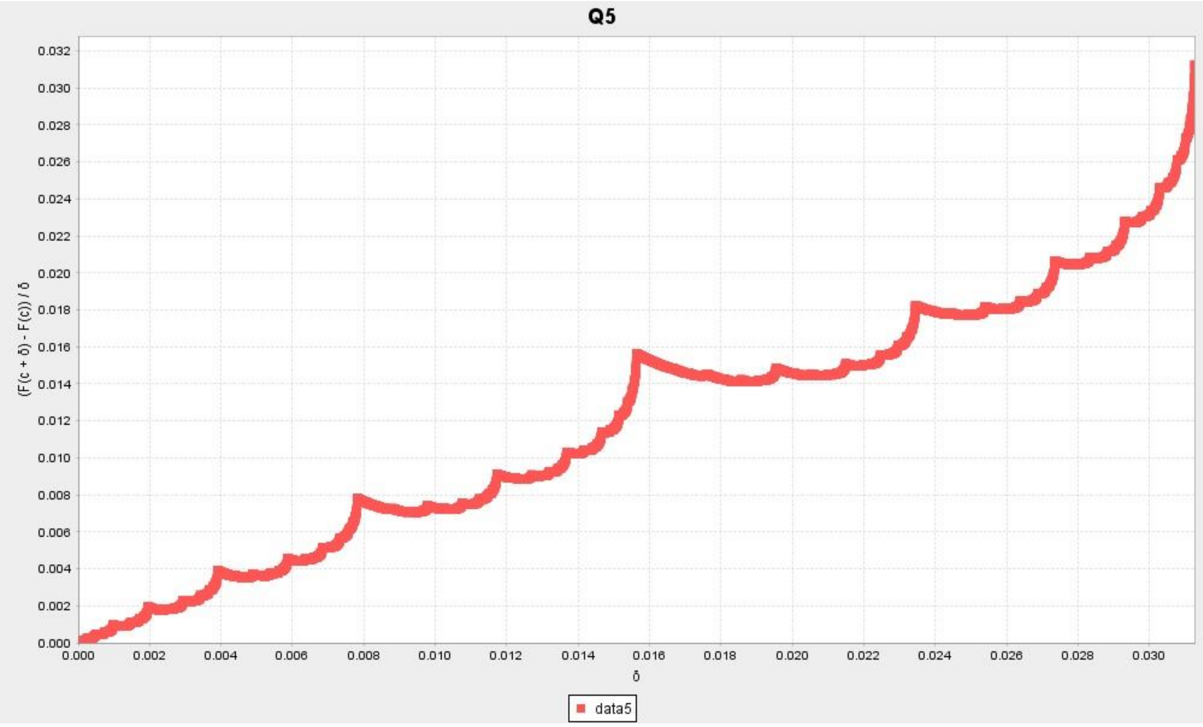

Figure 5.1: Graph of $\frac{\mathrm{F(c+\delta)-F(c)}}{\delta}$ against δ for $\delta > 0$

It appears that F is right-differentiable, as the right limit appears to be 0.

The graph for $\delta < 0$ is shown below.

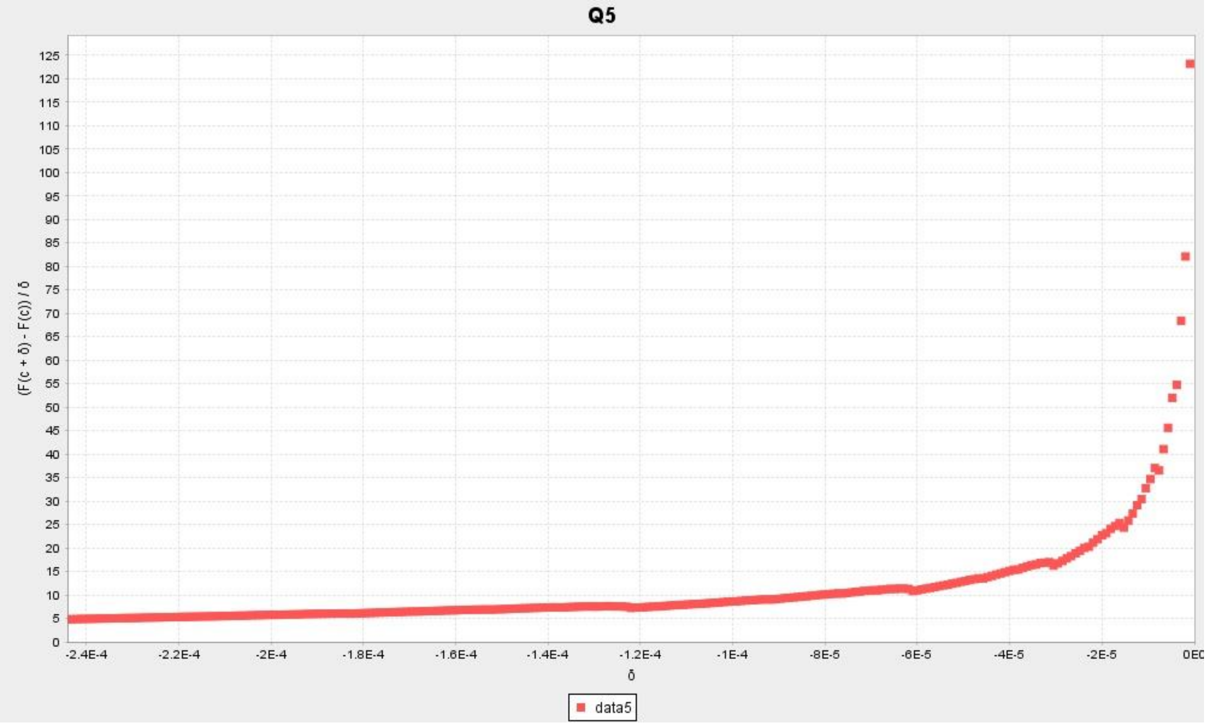

Figure 5.2: Graph of $\frac{F(c+\delta)-F(c)}{\delta}$ against δ for $\delta > 0$

It appears that F is not left-differentiable, as the left limit appears to be ∞.

Question 6

Case 1: $p > \frac{1}{2}$

With reference to the plots in Question 5, we conjecture that F is right- but not left-differentiable.

<u>Right-differentiability</u>

Let $c = \sum_{i=1}^{n} \frac{c_i}{2^i}$ and $\delta = \sum_{i=r}^{s} \frac{c_i}{2^i}$ where $n + 1 \leq r < s$, $c_r = 1$ and $c_i \in \{0, 1\}$. Then

$$\begin{aligned} F(c+\delta) - F(c) &= \sum_{j \geq n+1:\, x_j = 1} p^{f_{c+\delta}(j)} (1-p)^{j - f_{c+\delta}(j)} \\ &= \sum_{j \geq n+1:\, x_j = 1} p^{f_\delta(j) + f_c(n) + 1} (1-p)^{j - f_\delta(j) - f_c(n) - 1} \\ &= \left(\frac{p}{1-p}\right)^{f_c(n)+1} F(\delta) \end{aligned}$$

Now $\frac{F(\delta)}{\delta} \leq F\left(\frac{1}{2^{r-1}}\right) / \frac{1}{2^r} = (1-p)^{r-1} 2^{-r}$. As $\delta \to 0$ implies $r \to \infty$, and $1 - p < \frac{1}{2}$, we have $\lim_{\delta \searrow 0} (1-p)^{r-1} 2^{-r} = 0$. Then by Sandwich Thm $\lim_{\delta \searrow 0} \frac{F(c+\delta) - F(c)}{\delta} = 0$.

<u>Light-differentiability</u>

For $\delta < 0$ we consider $\delta_i = -\frac{1}{2^i}$ for $i \geq n+1$. Then by a result in Question 4 $F(c+\delta_i) - F(c) = -p^{f_c(n)}(1-p)^{n-f_c(n)}p^{i-n} = -\left(\frac{1-p}{p}\right)^{n-f_c(n)} p^i$, so $p > \frac{1}{2}$ implies $\frac{F(c+\delta_i)-F(c)}{\delta_i} = \left(\frac{1-p}{p}\right)^{n-f_c(n)} p^i 2^i \to \infty$ when $i \to \infty$. F is not left-differentiable.

Case 2: $p < \frac{1}{2}$

We plot $F(x)$ for $p = \frac{1}{4}$ and compare with that for $p = \frac{3}{4}$ (in Figure 3.1). The class Q6 generates the superposed plot and is shown below.

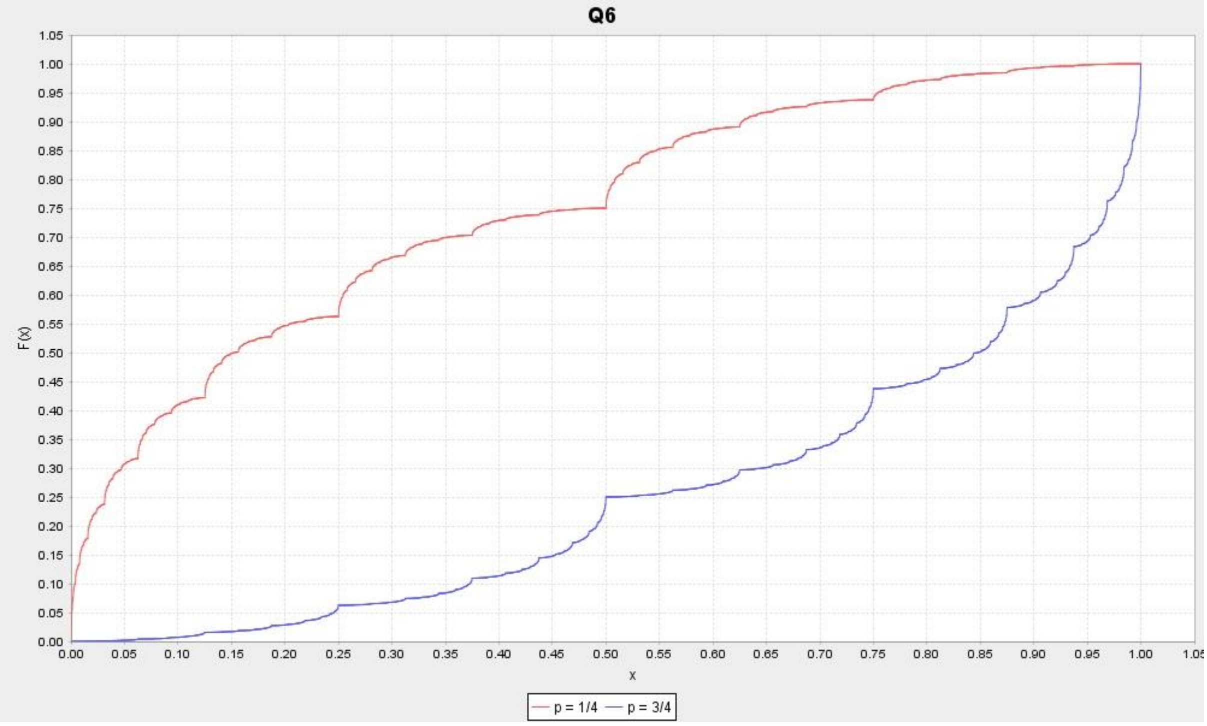

Figure 6.1: Graphs of F for p = $\frac{1}{4}$ and p = $\frac{3}{4}$

The two graphs look like rotation copies of each other. Hence we conjecture that F for $p < \frac{1}{2}$ is left- but not right-differentiable.

For any $x \in (0,1)$, $1-x$ is the number that replaces all digits 0 in x by 1, and all digits 1 by 0. By symmetry $P_{p=a}(X \leq x)$ is the same as $P_{p=1-a}(X \geq 1-x)$. As $P(X=x) = 0$, this implies $F_{p=a}(x) = 1 - F_{p=1-a}(1-x)$.

Let $a < \frac{1}{2}$. By Case 1 we have $F_{p=a}(x)$ right- but not left-differentiable, so $F_{p=a}(x) = 1 - F_{p=1-a}(1-x)$ is left- but not right-differentiable.

Case 3: $p = \frac{1}{2}$

The class Q6 also plots graph of F for $p = \frac{1}{2}$ as below.

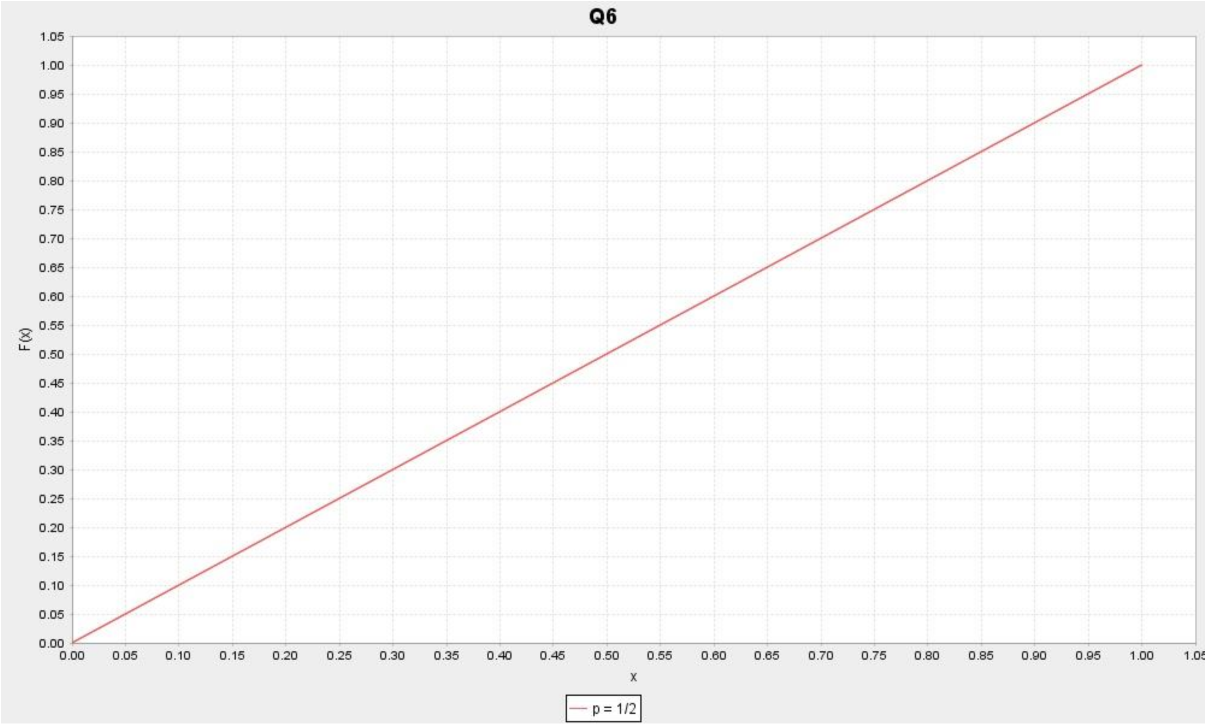

Figure 6.2: Graph of F for $\mathrm{p} = \frac{1}{2}$

We show that $F(x) = x$. By Question 2 for any $c = \sum_{i=1}^{n} \frac{c_i}{2^i}$ (i.e. with a finite binary expansion), $F(c) = \sum_{j:\, x_j=1} p^{f_c(j)}(1-p)^{j-f_c(j)} = \sum_{j:\, x_j=1} \left(\frac{1}{2}\right)^j = c$.

Now for any $x = \sum_{i=1}^{\infty} \frac{x_i}{2^i}$ we can find sequence $\left(\sum_{i=1}^{n} \frac{x_i}{2^i}\right)$ converging to x. By Question 4 F is continuous, so $F(x) = x$. This implies F is differentiable with derivative 1.

Appendix

Main classes

Coin.java

```java
package project;

class Coin {

	double p;

	byte toss() {
		if (Math.random() < p) return 1;
		else return 0;
	}

	Coin(double one) {
		p = one;
	}

}
```

Q1.java

```java
package project;

import java.io.IOException;

public class Q1 {

	static double[] xData = new double[2049];
	static double[] yData = new double[2049];

	static void task(double p, String file) throws IOException {

		Coin c = new Coin(p);
		double[] sample = new double[38416];

		for(int i = 0; i < 38416; i++) {
			double tmp = 0;
			for (int j = 0; j < 30; j++) tmp += (double) c.toss() / Math.pow(2, j+1);
			sample[i] = tmp;
		}

		utilities.Tools.mergeSort(sample, 38416);

		for (int i = 0; i < 2049; i++) xData[i] = (double) i / 2048;

		int count = 0, i = 1;
		yData[0] = 0;
		while (i < 2049 && count < 38416)
			if (sample[count] <= (double) i / 2048) count++;
			else {
				yData[i] = (double) count / 38416;
				i++;
			}
		for (int j = i; j < 2049; j++) yData[j] = 1;

		graphs.XYDataset d = new graphs.XYDataset("Q1", "x", "F(x)");
		d.addSeries(xData, yData, 2049, "data1");
```

```
            d.plotLine(false);
    //      d.display();
            d.save(file);

    }

    public static void main(String[] args) throws IOException {

            task(2/3d, "F1.jpg");

    }

}
```

Q3.java

```java
package project;

import java.io.IOException;

public class Q3 {

	static double[] xData = new double[2049];
	static double[] yData = new double[2049];

	static double F(double p, int i, int n) {

		byte[] b = utilities.Tools.extractBinary(i, n);
		int count = 0;
		double sum = 0;
		for (int j = 1; j < n + 1; j++)
			if (b[n - j] == 1) {
				sum += Math.pow(p, count) * Math.pow(1d - p, j - count);
				count++;
			}
		return sum;

	}

	static void fillData(double p) {

		for (int i = 0; i < 2048; i++) {
			xData[i] = (double) i / 2048;
			yData[i] = F(p, i, 11);
		}
		xData[2048] = 1;
		yData[2048] = 1;

	}

	public static void main(String[] args) throws IOException {

```

```java
		double p = 3/4d;

		fillData(p);

		graphs.XYDataset d = new graphs.XYDataset("Q3", "x", "F(x)");
		d.addSeries(xData, yData, 2049, "data3");
		d.plotLine(false);
	//	d.display();
		d.save("F3.1.png");

		Q1.task(p,"2.png");
		double max = 0d, average = 0d;
		int badEst = 0;
		for (int i = 0; i < 2049; i++) {
			double tmp = Math.abs(Q1.yData[i] - Q3.yData[i]);
			if (tmp > 0.005d) badEst++;
			if (tmp > max) max = tmp;
			average += tmp;
		}
		average /= 2049;
		System.out.println(max + "\n" + average + "\n" + badEst);

	}

}
```

Q5.java

```java
package project;

import java.io.IOException;

public class Q5 {

	public static void main(String[] args) throws IOException {

		double p = 3/4d;

		double xData[] = new double[65536];
		double yData[] = new double[65536];

		for (int k = 1; k < 65537; k++) {
			xData[k - 1] = k / Math.pow(2, 20);
			yData[k - 1] = (Q3.F(p, (9 >> 16) + k, 20) - Q3.F(p, 9 >> 16, 20)) * Math.pow(2, 20) / k;
		}

		graphs.XYDataset d = new graphs.XYDataset("Q5", "\u03b4", "(F(c + \u03b4) - F(c)) / \u03b4");
		d.addSeries(xData, yData, 65536, "data5");
		d.setRange('x', 0, Math.pow(2, -5));
		d.plotScatter();
	//	d.display();
		d.save("F5.1.jpg");

		for (int k = 1; k < 65537; k++) {
			xData[k - 1] = -k / Math.pow(2, 20);
			yData[k - 1] = (Q3.F(p, (9 << 16) - k, 20) - Q3.F(p, 9 << 16, 20)) * Math.pow(2, 20) / -k;
		}

		d = new graphs.XYDataset("Q5", "\u03b4", "(F(c + \u03b4) - F(c)) / \u03b4");
		d.addSeries(xData, yData, 65536, "data5");
		d.setRange('x', -Math.pow(2, -5), 0);
		d.plotScatter();
	//	d.display();
		d.save("F5.2a.jpg");
```

```java

		d = new graphs.XYDataset("Q5", "\u03b4", "(F(c + \u03b4) - F(c)) / \u03b4");
		d.addSeries(xData, yData, 65536, "data5");
		d.setRange('x', -Math.pow(2, -12), 0);
		d.plotScatter();
	//	d.display();
		d.save("F5.2.jpg");

	}
}
```

Q6.java

```java
package project;

import java.io.IOException;

public class Q6 {

	public static void main(String[] args) throws IOException {

		Q3.fillData(1/4d);
		double[] yData = Q3.yData.clone();
		Q3.fillData(3/4d);

		graphs.XYDataset d = new graphs.XYDataset("Q6", "x", "F(x)");
		d.addSeries(Q3.xData, yData, 2049, "p = 1/4");
		d.addSeries(Q3.xData, Q3.yData, 2049, "p = 3/4");
		d.plotLine(false);
	//	d.display();
		d.save("F6.1.jpg");

		Q3.fillData(1/2d);

		d = new graphs.XYDataset("Q6", "x", "F(x)");
		d.addSeries(Q3.xData, Q3.yData, 2049, "p = 1/2");
		d.plotLine(false);
	//	d.display();
		d.save("F6.2.jpg");

	}

}
```

Libraries

I used the external libraries JCommon v1.0.23 (http://www.jfree.org/jcommon/) and JFreeChart v1.0.19 (http://www.jfree.org/jfreechart/) , and some classes from my own library MyLibrary as shown below.

XYDataset.java

```java
package graphs;

import java.io.File;
import java.io.IOException;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.LogarithmicAxis;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.*;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;

public class XYDataset {

	public XYDataset(String chartTitle, String xLabel, String yLabel) {

		title = chartTitle;
		xLbl = xLabel;
		yLbl = yLabel;
		xAxis.setLabel(xLabel);
		yAxis.setLabel(yLabel);

	}
```

```java

    private XYSeriesCollection collection = new XYSeriesCollection();
    private String title, xLbl, yLbl;
    private NumberAxis xAxis = new NumberAxis();
    private NumberAxis yAxis = new NumberAxis();
    private JFreeChart chart;

    public void addSeries(double[][] data, int number, String key) {

        XYSeries series = new XYSeries(key);
        for (int i = 0; i < number; i++) series.add(data[i][0], data[i][1]);
        collection.addSeries(series);

    }

    public void addSeries(double[] xData, double[] yData, int number, String key) {

        XYSeries series = new XYSeries(key);
        for (int i = 0; i < number; i++) series.add(xData[i], yData[i]);
        collection.addSeries(series);

    }

    public void addSeries(XYData d1, String key) {

        XYSeries series = new XYSeries(key);
        for (int i = 0; i < d1.n; i++) series.add(d1.x[i], d1.y[i]);
        collection.addSeries(series);

    }

    public void useNumberAxis(char axis) {

        if (axis == 'x') xAxis = new NumberAxis(xLbl);
        if (axis == 'y') yAxis = new NumberAxis(yLbl);

    }

```

```
        public void useLogAxis(char axis) {

                if (axis == 'x') xAxis = new LogarithmicAxis(xLbl);
                if (axis == 'y') yAxis = new LogarithmicAxis(yLbl);

        }

        public void setRange(char axis, double lower, double higher) {

                if (axis == 'x') xAxis.setRange(lower, higher);
                if (axis == 'y') yAxis.setRange(lower, higher);

        }

        public void setTickUnit(char axis, double tick) {

                if (axis == 'x') xAxis.setTickUnit(new NumberTickUnit(tick));
                if (axis == 'y') yAxis.setTickUnit(new NumberTickUnit(tick));

        }

        public void display() {

                ApplicationFrame af = new ApplicationFrame(chart.getTitle().getText());

                ChartPanel chartPanel = new ChartPanel(chart);
            chartPanel.setPreferredSize(new java.awt.Dimension(1000, 600));

            af.setContentPane(chartPanel);
                af.pack();
                RefineryUtilities.centerFrameOnScreen(af);
                af.setVisible(true);

        }

        public void save(String file) throws IOException {

                ChartUtilities.saveChartAsJPEG(new File(file), chart, 1000, 600);
```

```
    }

    public void plotLine(boolean showShape) {

        chart = new JFreeChart(title, new XYPlot(collection, xAxis, yAxis, new XYLineAndShapeRenderer(true, showShape)));

    }

    public void plotScatter() {

        chart = new JFreeChart(title, new XYPlot(collection, xAxis, yAxis, new XYLineAndShapeRenderer(false, true)));

    }

}
```

Tools.java

```java
package utilities;

public class Tools {

	public static void mergeSort(double[] ori, int length) {

		if (length < 2) return;
		int midpt = length / 2;

		double[] left = new double[midpt];
		System.arraycopy(ori, 0, left, 0, midpt);

		double[] right = new double[length - midpt];
		System.arraycopy(ori, midpt, right, 0, length - midpt);

		mergeSort(left, midpt);
		mergeSort(right, length - midpt);

		int l = 0, r = 0, o = 0;
		while (l < midpt && r < length - midpt)
			if (left[l] < right[r]) ori[o++] = left[l++];
			else ori[o++] = right[r++];
		while (l < midpt)
			ori[o++] = left[l++];
		while (r < length - midpt)
			ori[o++] = right[r++];

	}

	public static byte[] extractBinary(long number, int length) {

		byte[] digits = new byte[length];

		for (int i = 0; i < length; i++)
			digits[i] = (byte) ((number & (1 << i)) >> i);

```

```
                return digits;

        }

}
```