An Approximation for the Coupon Collector's problem

A PROJECT
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Lin Dai

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master Of Science

BARRY JAMES, KANG JAMES

June, 2012

Acknowledgements

I would like to thank everyone who has helped me in those two years.

I especially would like to thank my advisors, Dr. Barry James, and Dr. Kang James, for their help and guidance. Without their support, I couldn't have finished.

Thanks to Dr. Bruce Peckham for being on my committee. I have learned a great deal from him.

I would also thank my family's support me when I am studying in Duluth.

Abstract

In probability theory, the coupon collector's problem relates to a person trying to collect a complete set of m different types of coupons, with each type having a certain probability of being collected. This problem asks what is the expected number of trials to get j different types of coupons $(E(C_j))$. This project introduces an approximation method proposed by Barry James, Kang James and Anthony Gamst, and attempts to illustrate the method by examples, which show how this method works for many distributions of the probability of the coupons. For "larger" j $(j > \frac{3}{4}m)$, this project suggest a modification of the approximation method, by giving a corrected approximation according to the previous method.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In probability theory, the coupon collector's problem relates to a person trying to collect a complete set of different types of coupons, with each type having a certain probability of being collected. This problem basically asks two questions: what is the expected number of trials to get all kinds of coupons,and what is the distribution of the number of the trials. A more general case is to find the expected number of trials until one gets a subset of coupons (j different kinds of coupon). So that a general problem would be: Suppose that there are m coupons, from which coupons are being collected with replacement. What is the expectation of the number of trials until j different kinds of coupons have been collected?

The coupon collector's problem has a long history, which started in 1708, when the problem related to the coupon collector's problem is first seen in the literature in *De Mensura Sortis* [1]written by A.De Moivre. In English it means 'On the measurement of Chance'. In 1938 the coupon collector problem appear in "A Problem in Cartophily"[2], written by F. G. Maunsell, and gives us a general idea of what the coupon collector problem is. In that paper the author mentioned an approximation for the coupon collector's problem in the equal probability for the expected value to collect a full set. In 1950, the coupon collector's problem was introduced in the book *An Introduction to Probability Theory and Its Applications*[3], written by Willam Feller, where Willam Feller considers the coupon collector's problem as a kind of urn problem, which treats each kind of coupon as an urn and treats each trial as a ball put into an urn. From then on, the coupon collector appears in many textbooks.

In 1954, Von Schelling[4] calculated the waiting time when the probability of collecting each coupon was not uniform in *Coupon Collecting for Unequal Probabilities.* This paper shows the value of the expected number of trials until getting j different kinds of coupons when the probabilities P_i are not equal. In 1960 D.J. Newman[5] calculated the waiting time to collect two complete sets of coupons in *The Double Dixie-Cup Problem.* Since another problem is the distribution of the number of trials, in 1970, the paper *On the Coupon Collector's Waiting Time*[6] shows some theorem related to the probability distribution of the waiting time. In 1986 Lars Holst[7] took the approach of using the Poisson process to calculate the waiting time in the book *On Birthday, Collector's Occupancy, and Other Classical Urn Problems*, and in 1992 Flajolet[8] in *Birthday paradox, coupon collector's, caching algorithms and self-organizing search* shows how to use generating function to calculate the solution of the coupon collector's problem.

The coupon collector's problem has many applications, which is mentioned in [9]. It has applications especially in electrical engineering and biology. In electrical engineering it is related to the cache fault problem, and also can be used in electrical fault detection. In biology, the coupon collector's problem can be use to estimate the number of species of animals.

In Chapter 2, I will introduce the exact value of the coupon collector problem. In Chapter 3 I will introduce the approximation method proposed by Barry James, Kang James and Anthony Gamst, and show examples of how well this approximation works in the equal case (with equal probability distribution) and unequal case (with nonuniform probability distribution) In particular I will compare the exact value and approximation value in heavy tail and light tail distributions. In Chapter 4 I will do some adjustment for the approximation introduced in Chapter 3, and use some examples to illustrate it.

Chapter 2

The Exact Value for the Coupon Collector's Problem

This chapter introduce two equations that are used to compute the exact value for the coupon collector's problem.

Following are the notations that will be used in this paper:

m— number of types of coupon

p_i— fixed probability of getting the ith type coupon in one trial

n— number of trials

C_j— the number of trials until the first time there are j different types of coupons

Y_n— the number of types of coupons collected in n trials

Von Schelling[4] in his paper uses the following equation to compute the excepted time for a size j subset collection.

$$E(C_j) = \sum_{k=m-j+1}^{m} (-1)^{j+k-m-1} \binom{k-1}{m-j} \sum_{|J|=k} \frac{1}{P_J}.$$

Notation: J is a subset of the set $\{p_1, p_2, \ldots, p_m\}$.

$$P_J = \sum_{p_i \in J} p_i$$

For a full collection, the equation becomes

$$\begin{aligned} E(C_m) &= \sum_{k=1}^{m}(-1)^{k-1}\sum_{|J|=k}\frac{1}{P_J} \\ &= \sum_{i=1}^{m}\frac{1}{p_i} - \sum_{1\le i<j\le m}\frac{1}{p_i+p_j} + \sum_{1\le i<j<k\le m}\frac{1}{p_i+p_j+p_k} - + \cdots . \end{aligned}$$

Flajolet *et al.* [8] present another method to calculate the exact value for the coupon collector's problem using variables defined over sequences of independent samples from a finite population. The equation is

$$E(C_j) = \sum_{q=0}^{j-1}[u^q]\int_0^\infty \prod_{j=1}^{m}(1+u(e^{p_i t}-1))e^{-t}dt.$$

Notation: Given a generating function

$$f(u) = \sum_q a_q u^q.$$

$[u^q]f(u)$ denotes the coefficient of u^q in f:

$$[u^q]f(u) = a_q.$$

In particular, for a full collection

$$E(C_m) = \int_0^\infty (1-\prod_{i=1}^{m}(1-e^{-p_i t}))e^{-t}dt$$

Which is the same as the value getting from Poisson method[10].

We outline the method in the appendix.

Chapter 3

Approximation Method for the Coupon Collector's Problem

3.1 Approximation Method Proposed by Barry James, Kang James, and Anthony Gamst

First let's think about another question: what is the number of different kinds of coupon we expect to get after n trials?

Suppose there are m different types of coupons. If Y_n is the number of coupons obtained after n trials, let I_i be the identity that we get coupon i in these n trials, so that,

$$I_i = \begin{cases} 1 & \text{if coupon } i \text{ is in these n trials} \\ 0 & \text{if coupon } i \text{ is not in these n trials.} \end{cases}$$

The probability of failing to get coupon i for one trial is $1 - p_i$. The probability that coupon i is not in those n trials is

$$P(i \text{ is not in n trials}) = (1 - p_i)^n,$$

so that

$$E(I_i) = 1 - (1 - p_i)^n.$$

Then the expectation of the number of coupons we get in these n trials is

$$\begin{aligned} E(Y_n) &= E(I_1 + I_2 + \cdots + I_m) \\ &= \sum_{i=1}^{m} E(I_i) \\ &= \sum_{i=1}^{m} (1 - (1 - p_i)^n) \\ &= m - \sum_{i=1}^{m} (1 - p_i)^n \end{aligned}$$

In the coupon collector's problem if $E(C_j)$ is the expected number of trials that first time getting j different kinds of coupons, it is likely that if we collect $E(C_j)$ coupon there is a huge chance the number of coupon we get is j.

For the above equation, let $E(Y_n) = j$. By solving for n we may get a good approximation for the coupon collector's problem.

The approximation proposed by Barry James, Kang James, and Anthony Gamst is to find the number of trials such that the expectation of the number of coupons in such an experiment is equal to j.

3.2 Equal Probability Case

In the equal probability case the probabilities of getting all kinds of coupon are equal. If there is a total of m types of coupons, the probability to get each kind of coupon is $p_i = \dfrac{1}{m}$. For this case the exact value is easy to calculate:

Treat T_1 as the number of trials until getting one coupon, T_2 as the number of trials after getting one coupon until getting a second type, T_i be the number of trials after getting $i - 1$ types of coupon until getting i types. After getting $i - 1$ different kinds of coupon there are $m - i + 1$ coupons that haven't been collected, so the probability of getting one of those coupons is $\dfrac{m - i + 1}{n}$. So T_i has a geometric distribution with $p = \dfrac{m - i + 1}{m}$,

and

$$E(T_i) = \frac{m}{m - i + 1}$$

Then the exact value is

$$\begin{aligned} E(C_j) &= E(T_1 + T_2 + \cdots + T_j) \\ &= \sum_{i=1}^{j} E(T_i) \\ &= \sum_{i=1}^{j} \frac{m}{m-j+1}. \end{aligned}$$

Approximation 1 (approximation using the method proposed by Barry James, Kang James, and Anthony Gamst)

According to the method introduced before, solve the equation

$$m - \sum_{i=1}^{m} (1 - \frac{1}{m})^n = j.$$

The approximation we get for $E(C_j)$ is

$$n = \frac{log(\frac{m-j}{m})}{log(\frac{m-1}{m})}.$$

Approximation 2 (approximation using the asymptotic expansion for the Harmonic numbers)

According to the exact value for the equal probability case, for the full collection

$$\begin{aligned} E(C_m) &= \sum_{i=1}^{m} \frac{m}{m-i+1} \\ &= m \sum_{i=1}^{m} \frac{1}{i} \\ &= mH_m \end{aligned}$$

Here, the nth harmonic number H_m is the sum of the reciprocals of the first m natural numbers.

The corresponding asymptotic expansion for the Harmonic numbers as $m \to \infty$ is:

$$H_m \approx \quad logm + \kappa,$$

where κ is Euler's constant $\kappa = 0.5772156649$

So we can get another approximation for $E(C_j)$

$$\begin{aligned}
n &= m\sum_{i=m-j+1}^{m}\frac{1}{i} \\
&= m(\sum_{i=1}^{m}\frac{1}{i}-\sum_{i=1}^{m-j}\frac{1}{i}) \\
&\approx m((logm+\kappa)-(log(m-j)+\kappa)) \\
&= mlog\frac{m}{(m-j)}
\end{aligned}$$

Example 3.1: $m = 20$, $p_i = 1/20$

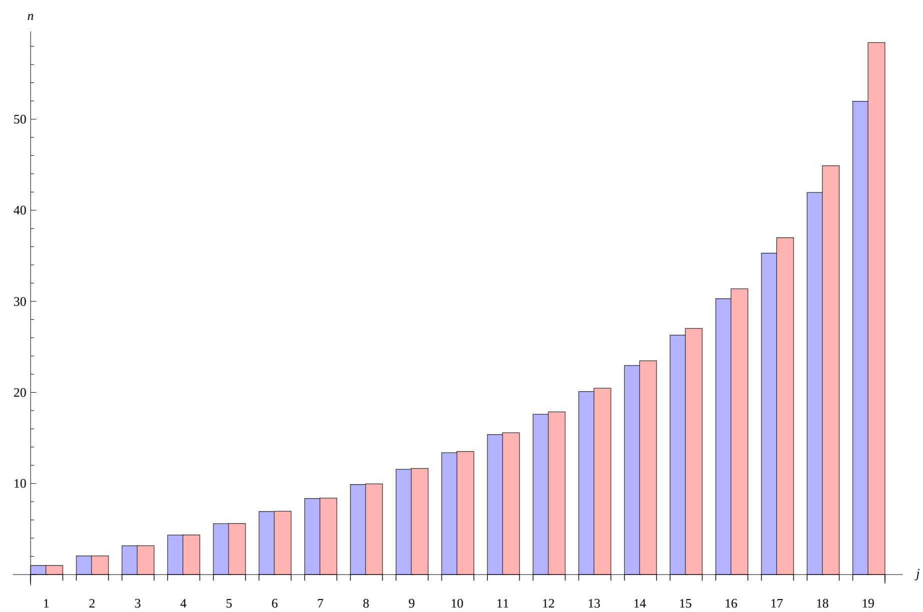

Figure 3.1: Bar chart of $E(C_j)$ and Approximation 1($m = 20$, $p_i = 1/20$)
Two bars in each block in order of Exact value, Approximation 1

j	$E(C_j)$	Approximation 1	Approximation 2	Relative Error 1	Relative Error 2
11	15.375	15.568	15.970	0.012	0.039
12	17.598	17.864	18.326	0.015	0.041
13	20.098	20.467	20.996	0.018	0.045
14	22.955	23.472	24.079	0.023	0.049
15	26.288	27.027	27.726	0.028	0.055
16	30.288	31.377	32.189	0.036	0.063
17	35.288	36.986	37.942	0.048	0.075
18	41.955	44.891	46.052	0.070	0.098
19	51.955	58.404	59.915	0.124	0.153

Table 3.1: Approximation value and relative error ($m = 20$, $p_i = 1/20$)

Figure-3.1 and Table-3.1 show a comparison of the exact, approximation 1, and approximation 2 when $m = 20$ and $j = 1, 2, ..., 19$. From the graph we can see that these two approximations are very close to the exact value, especially the first approximation. According to Table-3.1, when $j < 15$ it is still a good approximation (relative error less than 0.030). All of the relative errors for it are less than 0.124. We can see that when m is constant the relative error is improving when j becomes larger. In general, approximation one is better than approximation two.

Example 3.2: $m = 100$, $p_i = \dfrac{1}{100}$

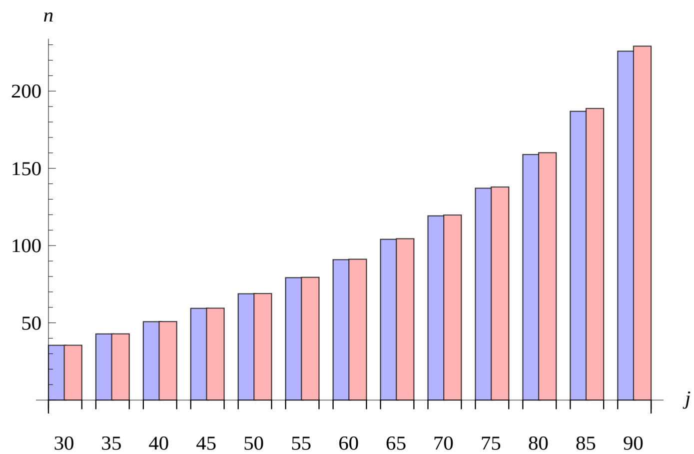

Figure 3.2: Bar chart of $E(C_j)$ and Approximation 1 ($m = 100$, $p_i = 1/100$)

j	$E(C_j)$	Approximation 1	Relative Error
30	35.4541	35.4889	0.001
40	50.7507	50.8267	0.001
50	68.8172	68.9676	0.002
60	90.8834	91.1702	0.003
70	119.239	119.794	0.005
80	158.964	160.138	0.007
90	225.841	229.105	0.014

Table 3.2: Approximation value and relative error ($m = 100$, $p_i = 1/100$)

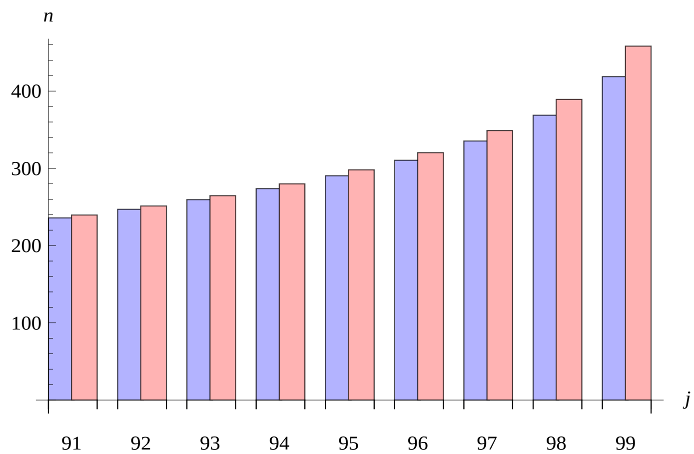

Figure 3.3: Bar chart of $E(C_j)$ and Approximation ($m = 100$, $p_i = 1/100$)

j	$E(C_j)$	Approximation 1	Relative Error
91	235.841	239.589	0.016
92	246.952	251.308	0.018
93	259.452	264.594	0.020
94	273.738	279.932	0.023
95	290.404	298.073	0.026
96	310.404	320.275	0.032
97	335.404	348.900	0.040
98	368.738	389.243	0.056
99	418.738	458.211	0.094

Table 3.3: Approximation value and relative error ($m = 100$, $p_i = 1/100$)

Graph-3.2 and Table-3.2 showing comparisons of the exact value and approximation 1 when $m = 100$ for the j value less than 90. They show that the approximation is very good. The relative error is no more than two percent when j is less than is 90. From Figure-3.3 and Table-3.3 we can get the results to Example 3.1: when j is very close to m, if j is increasing, the relative error increases dramatically (compare to the increase when j is small). Showing in Figure-3.3 and Table-3.3 is that the relative error is a little larger for the last five j values (when $j = 95, 96, 97, 98, 99$), but over all, it is a good approximation when the p_i are equal.

3.3 Unequal case

According to the method introduced in Section 3.1, by solving the equation we can get the approximation, so that the approximation is:

$$x = f^{-1}(j)$$

For ease in calculating we can use a linear interpolation method to find what the value is. The method is generally introduced below:

1. calculate Y_n when $n = 1, 2, 3, ...$
2. find n where $Y_n < j < Y_{n+1}$
3. Use linear interpolating, so that $C_j = n + \dfrac{j - Y_n}{Y_{n+1} - Y_n}$

Through Example 3, we will see whether using the linear interpolation method will affect the result of approximation.

Example 3.3: $m = 20$ $p_i = i/const$

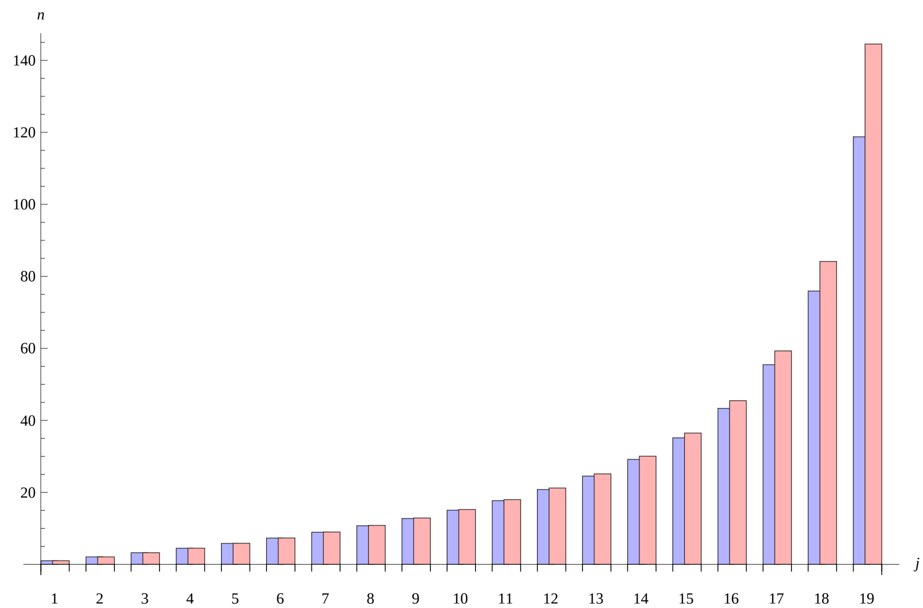

Figure 3.4: Bar chart of $E(C_j)$ and Approximation ($m = 20$ $p_i = i/const$)

j	$E(C_j)$	Approximation	Interpolate	Relative Error 1	Relative Error 2
5	5.812	5.838	5.842	0.004	0.005
6	7.827	7.329	7.336	0.006	0.007
7	8.911	8.976	8.977	0.007	0.007
8	10.714	10.811	10.816	0.009	0.010
9	12.736	12.879	12.882	0.011	0.011
10	15.033	15.238	15.243	0.014	0.014
11	17.679	17.972	17.973	0.017	0.017
12	20.787	21.204	21.208	0.020	0.020
13	24.522	25.123	25.125	0.024	0.025
14	29.151	30.032	30.033	0.030	0.030
15	35.130	36.468	36.474	0.038	0.038
16	43.305	45.459	45.464	0.050	0.050
17	55.447	59.282	59.286	0.069	0.069
18	75.914	84.120	84.122	0.108	0.108
19	118.761	144.521	144.523	0.217	0.217

Table 3.4: Approximation value and relative error ($m = 20$ $p_i = i/const$)

Figure-3.4 and Table-3.4 show the exact value compared with the approximation and the linear interpolated approximation. For the case when $m = 20$, $p_i = i/const$, the interpolated approximation is very close to the approximation value when we solve the equation. When j is small since $E(C_j)$ is small (it is always easy to get the first few coupons), the relative error of the two approximation is little different. When j is large the first three decimals of the relative error for the two approximations is exactly the same, which means that using the interpolated value to find the solution for the equation is acceptable. Example 3 also shows that the approximation is good, except when j is very close to m (for example $j = 19$).

Heavy-Tail and Light-Tail case

For the equal probability case and unequal case the relative error is different for each example, so we try to find in which case the approximation works better. The probabilities of the coupons in each example have different distributions, so we compare the two distribution to find if the relative error is related to the distribution of the probability of each type of coupon. The distributions we tried are a heavy-tail distribution ($p_i = i^{-\alpha}$/const) and a light-tail distribution ($p_i = e^{i\alpha}$/const). For heavy-tail distribution we chose the scale α from 0 to 2, and for the light-tail distribution the scale α is from 0 to 0.4. In the heavy-tail distribution the probability of getting the last few coupons is much greater that the probability of getting the coupons in the light-tail distribution. That means the T_j is much larger in the light-tail distribution if j is larger, since it usually costs a long time to collect the last few coupons. The Following graph shows how the approximation works for the heavy-tail distribution and for the light-tail distribution.

Example 3.4.1: $m = 20$, $j = 15$

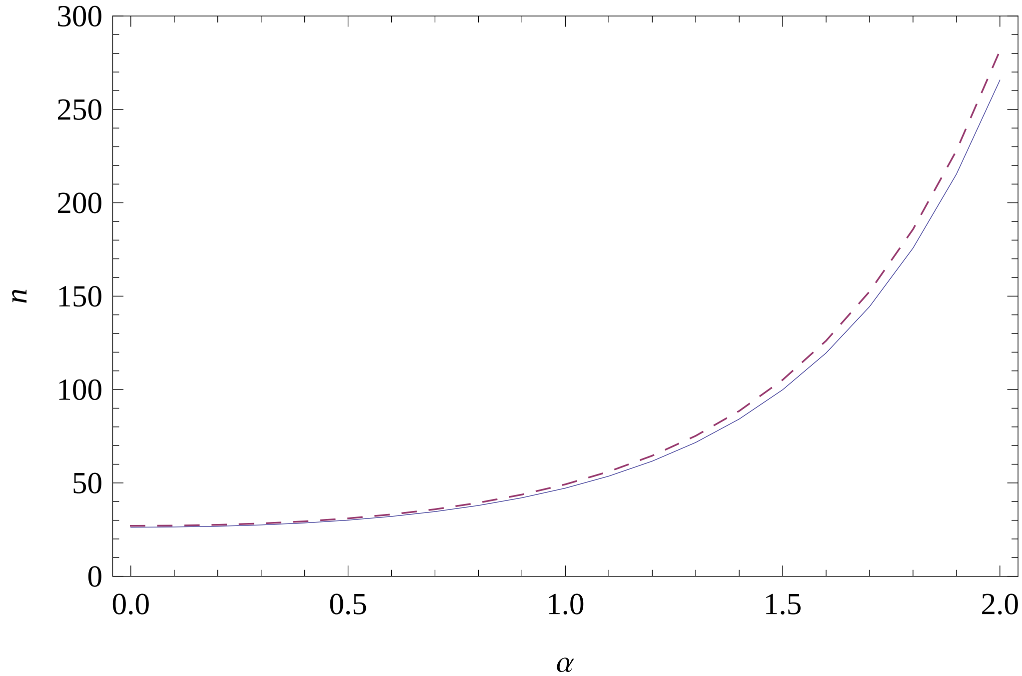

Figure 3.5: Plot of $E(C_j)$ and Approximation (heavy tail $m = 20$, $j = 15$)
Solid lines represents exact value, and dashed line represents approximation

α	$E(C_j)$	App	RError1
1.50	99.8789	105.175	0.0530232
1.60	119.582	126.11	0.0545914
1.70	144.395	152.483	0.0560131
1.80	175.722	185.791	0.0573001
1.90	215.373	227.965	0.0584666
2.00	265.682	281.497	0.0595276

Table 3.5: Approximation value and relative error (heavy tail $m = 20$, $j = 15$)

The above figure and table show the exact value and approximation in the heavy-tail distribution when $m = 20$ and $j = 15$. The approximation is good, as the relative error is less than 0.06. We can also see that the relative error increases when α is increasing.

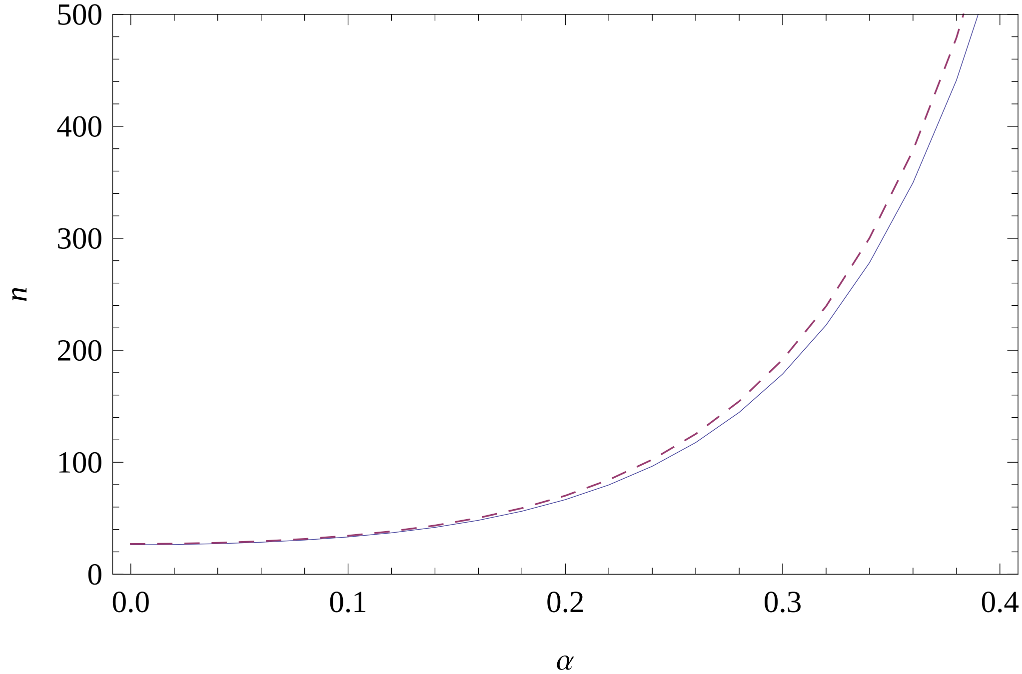

Figure 3.6: Plot of $E(C_j)$ and Approximation (light tail $m = 20$, $j = 15$)

α	$E(C_j)$	App	RError1
0.30	178.827	191.702	0.0719993
0.32	222.512	239.309	0.0754851
0.34	278.309	300.282	0.0789501
0.36	349.703	378.526	0.0824205
0.38	441.220	479.128	0.085918
0.40	558.732	608.717	0.0894607

Table 3.6: Approximation value and relative error (light tail $m = 20$, $j = 15$)

Figure-3.6 and Table-3.6 shows the approximation value and the exact value when $m = 20$ and $j = 15$ for the light-tail distribution. In Figure-6 the two lines are farther apart than the ones in the heavy-tail distribution case, and the relative errors are greater, meaning the approximation is better for the heavy-tail distribution compared to the light-tail distribution.

Example 3.4.2: $m = 20,\ j = 18$

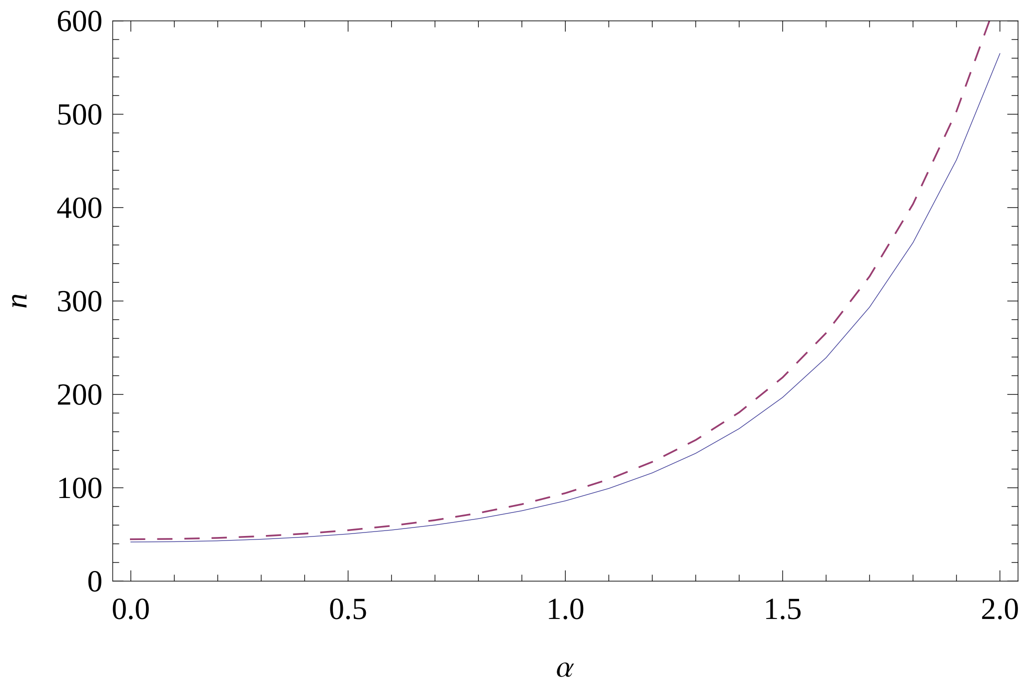

Figure 3.7: Plot of $E(C_j)$ and Approximation (heavy tail $m = 20,\ j = 18$)

j	$E(C_j)$	Approximation	Relative Error
1.50	196.805	218.161	0.109
1.60	239.287	265.715	0.110
1.70	326.318	326.318	0.112
1.80	362.534	403.796	0.114
1.90	451.129	503.140	0.115
2.00	564.974	630.879	0.117

Table 3.7: Approximation value and relative error (heavy tail $m = 20$, $j = 18$)

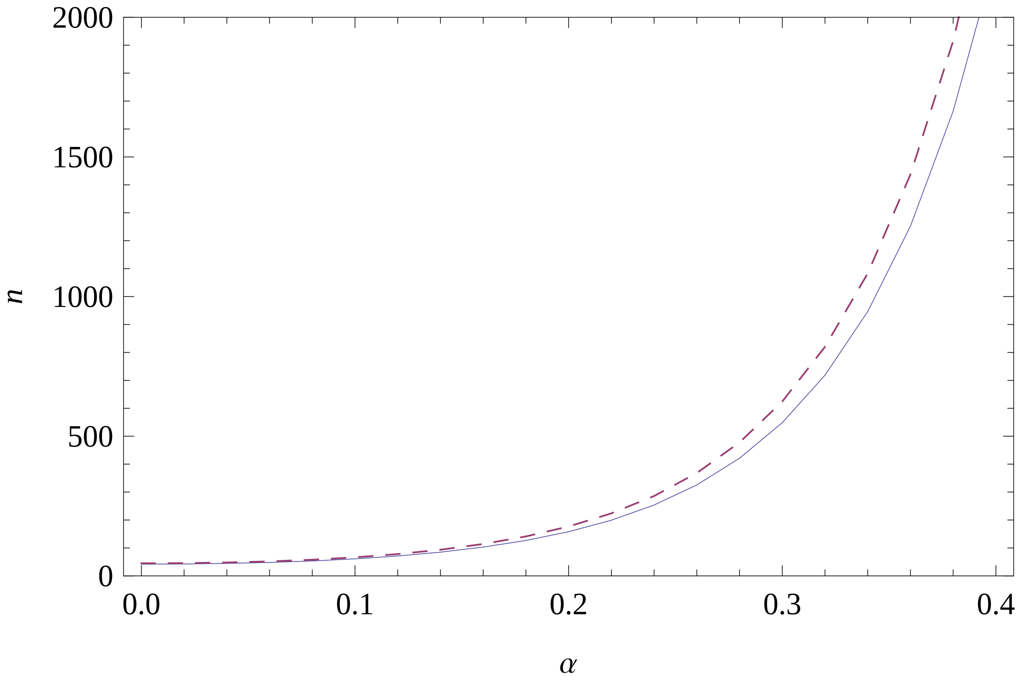

Figure 3.8: Plot of $E(C_j)$ and Approximation (light tail $m = 20$, $j = 18$)

α	$E(C_j)$	Approximation	Relative Error
0.30	548.533	624.154	0.138
0.32	718.588	820.047	0.141
0.34	946.377	1083.081	0.144
0.36	1252.352	1437.249	0.148
0.38	1664.295	1915.371	0.151
0.40	2220.361	2561.419	0.154

Table 3.8: Approximation value and relative error (light tail $m = 20$, $j = 18$)

When j is greater ($j = 18$), the distance between the two lines is greater, and from Table-3.7 and Table-3.8 the relative error is larger than when j is 15, especially for the heavy tail the relative error is more that 0.25.

Overall the approximation is good. But the results for larger j are not as good as for small j. And one disadvantage for this approximation is that it cannot calculate the approximation for the full collection of coupons. (There is no solution for the equation $\sum_{i=1}^{m}(1 - p_i)^x) = 0$, since the left side is always greater than 0.)

Chapter 4

Modified Approximation Method for the Coupon Collector's Problem

4.1 Adjusted Approximation Method

Since in some cases that the original approximation does not work well when $j = m-1, m-2, ...$, so we are trying to find an improvement for the approximation for the Coupon Collector's Problem.

The original approximation is to solve for

$$m - \sum_{i=1}^{m}(1 - \frac{1}{m})^n = j$$

The left side of the equation is $m - \sum_{i=1}^{m}(1 - \frac{1}{m})^n = E(Y_n)$, and if we want the solution of the equation to be $E(C_j)$, the value for the left side should be $E(Y_{E(C_j)})$. So we want to calculate the value for $E(Y_{E(C_j)})$ and do some correction to the right side of the equation.

According to the equation and the classical form for expectations of discrete random variables, we can get

$$\begin{aligned} E(Y_n) &= \sum_{i=1}^{m} P(Y_n \geq i) \\ &= \sum_{i=1}^{m} P(C_i \leq n) \end{aligned}$$

When $n = E(C_j)$ the equation will become

$$\begin{aligned} E(Y_{E(C_j)}) &= \sum_{i=1}^{m} P(C_i \leq E(C_j)) \\ &= j - \sum_{i=1}^{j} P(C_i > E(C_j)) + \sum_{i=1+j}^{m} P(C_i \leq E(C_j)) \\ &= j - P(C_j > E(C_j)) - \sum_{i=1}^{j-1} P(C_i > E(C_j)) + \sum_{i=1+j}^{m} P(C_i \leq E(C_j)). \end{aligned}$$

Define

$$P(C_j > E(C_j)) + \sum_{i=1}^{j-1} P(C_i > E(C_j)) - \sum_{i=1+j}^{m} P(C_i \leq E(C_j)) = f(j)$$

The function will change into

$$E(Y_{E(C_j)}) = j - f(j)$$

So that our goal is to find what is the approximation for $f(j)$.

When $j = 1$

$$C_1 = 1$$

$$E(Y_{E(C_1)}) = E(Y_1) = 1$$

So that

$$f(1) = 1 - E(Y_{E(C_1)}) = 0$$

When $j = m$

$$\begin{aligned} E(Y_{E(C_m)}) &= \sum_{i=1}^{m} P(C_i \leq E(C_m)) \\ &= m - \sum_{i=1}^{m} P(C_i > E(C_m)) \\ &= m - P(C_m > E(C_m)) - \sum_{i=1}^{m-1} P(C_i > E(C_m)). \end{aligned}$$

If C_j has a symmetric distribution, $P(C_j > E(C_j))$ will be about 0.5. Since we don't know what the distribution C_j exactly is, we treat it as 1/2. T_m, the time to get the last coupon, is usually large. According to that the differences between $E(C_m)$ and $E(C_i)$ $(i < n)$ are large, so that $P(C_i > E(C_m))$ is relatively small when $i < m$, so we treat these parts as 0. Then we get when $j = m$,

$$E(Y_{E(C_m)}) \approx m - \frac{1}{2},$$

So that

$$f(m) \approx 1/2.$$

According to the two points $(1, 0)$, $(m, 1/2)$ on $f(j)$, using linear interpolation we can get

$$f(j) \approx \frac{j-1}{2(m-1)}$$

so that we change the equation into

$$m - \sum_{i=1}^{m}(1-p_i)^n = j - \frac{j-1}{2(m-1)}.$$

By solving the equation above we try to get another approximation for the coupon collector's problem

4.2 Equal Probability Case

according to that, $p_i = 1/m$

by solving the equation

$$m - \sum_{i=1}^{m}(1-\frac{1}{m})^n = j - \frac{j-1}{2(m-1)}$$

we can get the corrected value is

$$n = \frac{log(m - j + \frac{j-1}{2(m-1)})}{log(\frac{m-1}{m})}$$

Next are the examples of the approximation in the equal probability case.

Example 4.1: $m = 20$ $p = 1/20$

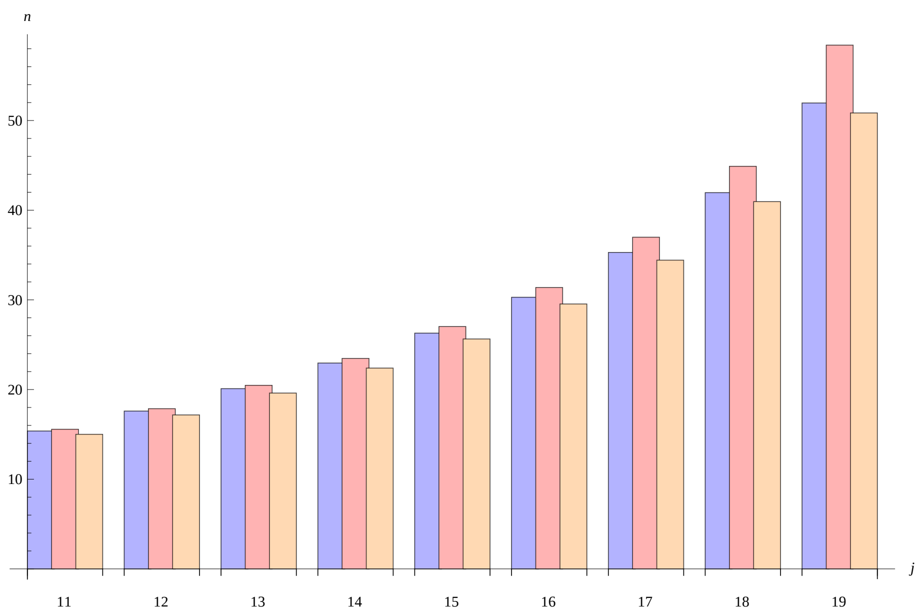

Figure 4.1: Bar chart of $E(C_j)$ and Approximation ($m = 20$ $p = 1/20$)
Three bars in each block in order of Exact value, Approximation value, Corrected Approximation value

11	15.3754	15.5675	15.0056	0.0124913	-0.0240524
12	17.5977	17.8638	17.1708	0.0151215	-0.0242573
13	20.0977	20.467	19.6068	0.01838	-0.0244233
14	22.9548	23.4723	22.3913	0.0225456	-0.0245497
15	26.2881	27.0268	25.6407	0.0280997	-0.0246263
16	30.2881	31.3772	29.5424	0.0359558	-0.0246226
17	35.2881	36.9857	34.4252	0.048107	-0.0244526
18	41.9548	44.8806	40.955	0.0699747	-0.0238293
19	51.9548	58.404	50.8442	0.124131	-0.0213761

Table 4.1: Approximation value and relative error($m = 20$ $p = 1/20$)

Table-4.1 and Figure-4.1 are showing that when $j < 15 = \frac{3}{4}(20)$, the original approximation works better. After 15, the corrected approximation works much better than the original approximation, especially when j =19, the relative error is reduced from 0.19 to 0.10.

Example 4.2 $m = 100$ $p = 1/100$

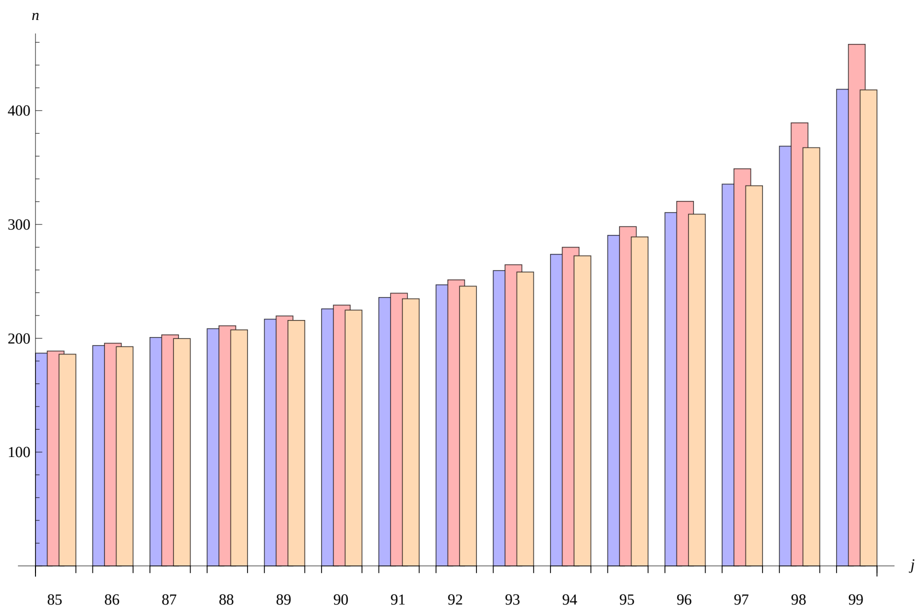

Figure 4.2: Bar chart of $E(C_j)$ and Approximation ($m = 100$ $p = 1/100$)

j	$E(C_j)$	Approximation	Corrected Approximation	Relative Error 1	Relative Error 2
85	186.915	188.762	185.987	0.010	-0.005
86	193.582	195.627	192.621	0.011	-0.005
87	200.724	203.000	199.73	0.011	-0.005
88	208.417	210.964	207.386	0.012	-0.005
89	216.75	219.622	215.681	0.013	-0.005
80	225.841	229.105	224.730	0.014	-0.005
91	235.841	239.589	234.686	0.016	-0.005
92	246.952	251.308	245.750	0.018	-0.005
93	259.452	264.594	258.200	0.020	-0.005
94	273.738	279.932	272.433	0.023	-0.005
95	280.404	298.073	289.047	0.026	-0.005
96	310.404	320.275	309.004	0.032	-0.005
97	335.404	348.800	333.993	0.040	-0.004
98	368.738	389.243	367.443	0.056	-0.004
99	418.738	458.211	418.203	0.094	-0.001

Table 4.2: Approximation value and relative error ($m = 100$ $p = 1/100$)

When m becomes larger ($m = 100$) it works extremely well when j is equal to 85 to 99. While the original approximation is from 0.010 to 0.094, the relative error for the corrected approximation is always less than 0.005. According to these results, when $j < 75 = \frac{3}{4}(100)$, the original approximation works as well as the corrected one, sometimes even better. We will use the corrected method when $j > \frac{3}{4}m$.

4.3 Unequal Case

Example 4.3: $m = 20$ $p_i = i/210$

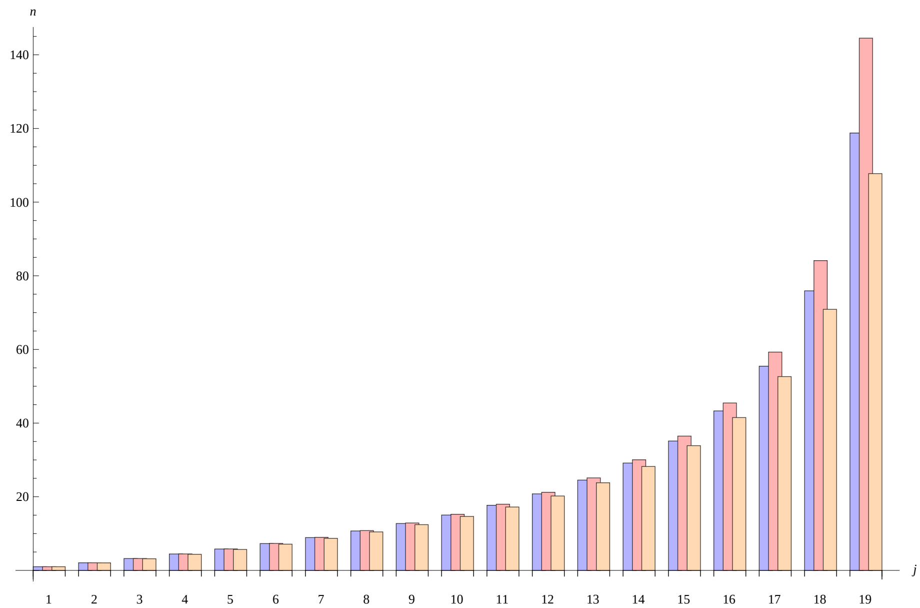

Figure 4.3: Bar chart of $E(C_j)$ and Approximation ($m = 20$ $p_i = i/210$)

j	$E(C_j)$	Approximation	Corrected Approximation	Relative Error 1	Relative Error 2
8	10.714	10.811	10.457	0.009	-0.023
9	12.736	12.879	12.422	0.011	-0.025
10	15.032	15.238	14.649	0.014	-0.026
11	17.679	17.972	17.210	0.020	-0.028
12	20.787	21.204	20.208	0.020	-0.028
13	24.522	25.123	23.796	0.024	-0.030
14	29.151	30.032	28.215	0.030	-0.032
15	35.130	36.468	33.870	0.038	-0.036
16	43.305	45.459	41.502	0.050	-0.042
17	55.447	59.282	52.626	0.069	-0.050
18	75.914	84.120	70.891	0.108	-0.066
19	118.761	144.521	107.744	0.217	-0.092

Table 4.3: Approximation value and relative error ($m = 20$ $p_i = i/210$)

The same as the equal case, when j is less than 15, the original approximation works better than the corrected approximation; for example, when j is 8, the relative error for the original one is 0.009 but after the correction is about 0.023; when j is 15, the relative error for both approximations is close, and after that the corrected approximation works better. We also find that while the original method usually overestimates, the corrected method usually under estimates.

Heavy-tail and light-tail case

To compare with the original approximation, we try the heavy-tail and light-tail case in the same distribution.

Example 4.3.1: $j = 15$

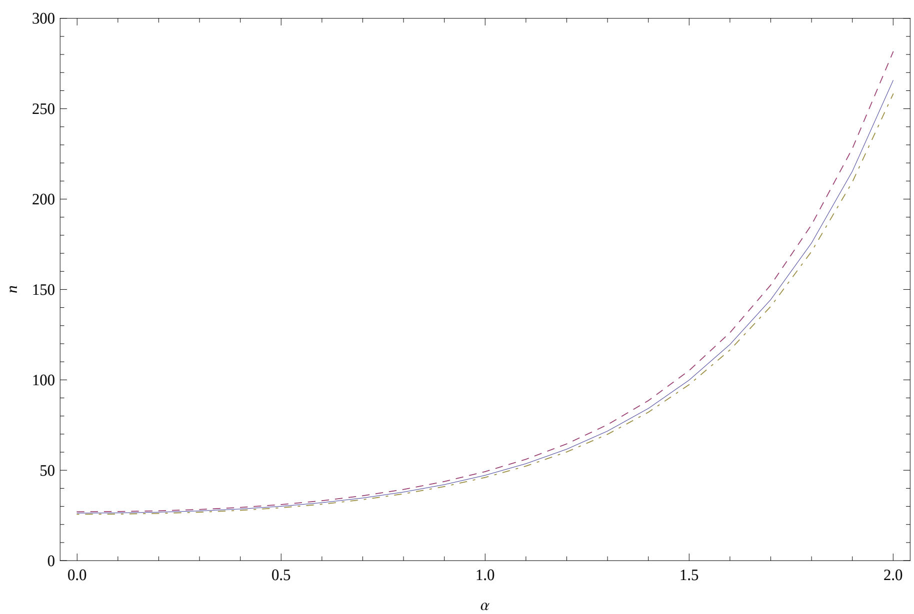

Figure 4.4: Plot of $E(C_j)$ and Approximation (heavy tail $m = 20$ $j = 15$)
Solid lines represents exact value, dashed line represents approximation value, and dot-dashed line represents corrected approximation value

α	$E(C_j)$	App	C App	RError1	RError2
1.50	99.8789	105.175	97.3654	0.0530	-0.0252
1.60	119.582	126.11	116.526	0.0546	-0.0256
1.70	144.395	152.483	140.629	0.0560	-0.0261
1.80	175.722	185.791	171.024	0.0573	-0.0267
1.80	215.373	227.965	209.45	0.0585	-0.0275
2.00	265.682	281.497	258.147	0.0595	-0.0284

Table 4.4: Approximation value and relative error (heavy tail $m = 20$ $j = 15$)

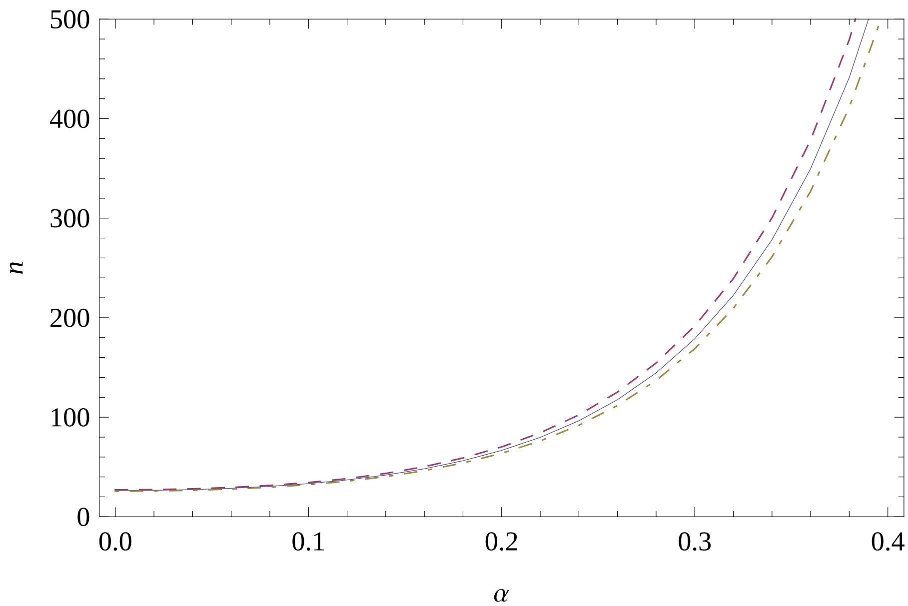

Figure 4.5: Plot of $E(C_j)$ and Approximation (light tail $m = 20$ $j = 15$)

α	$E(C_j)$	App	C App	RError1	RError2
0.30	178.827	191.702	168.87	0.0719993	-0.0557
0.32	222.512	239.309	209.469	0.0754851	-0.0586
0.34	278.309	300.282	261.159	0.0789501	-0.0616
0.36	349.703	378.526	327.092	0.0824205	-0.0647
0.38	441.22	479.128	411.341	0.085918	-0.0677
0.40	558.732	608.717	519.186	0.0894607	-0.0708

Table 4.5: Approximation value and relative error (light tail $m = 20$ $j = 15$)

When j is 15, the relative error for both approximations looks good from the graph, and the corrected one is better than the original one.

Example 4.3.2: $j = 18$

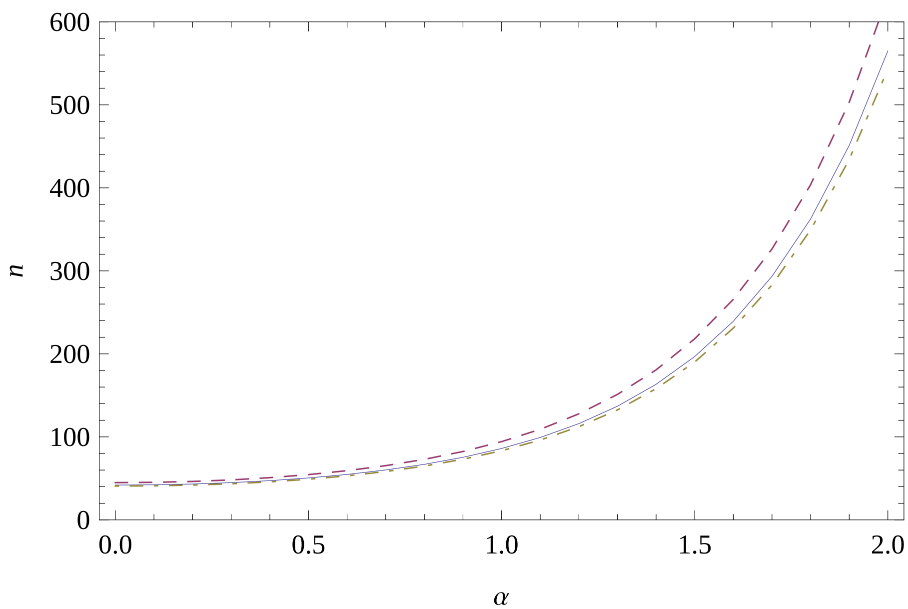

Figure 4.6: Plot of $E(C_j)$ and Approximation (heavy tail $m = 20$ $j = 18$)

α	$E(C_j)$	App	C App	RError1	RError2
1.50	196.805	218.161	191.599	0.109	-0.026
1.60	239.287	265.715	232.782	0.110	-0.027
1.70	326.318	326.318	285.172	0.112	-0.028
1.80	362.534	403.796	352.022	0.114	-0.029
1.80	451.129	503.140	437.572	0.115	-0.030
2.00	564.974	630.879	547.355	0.117	-0.031

Table 4.6: Approximation value and relative error (heavy tail $m = 20$ $j = 18$)

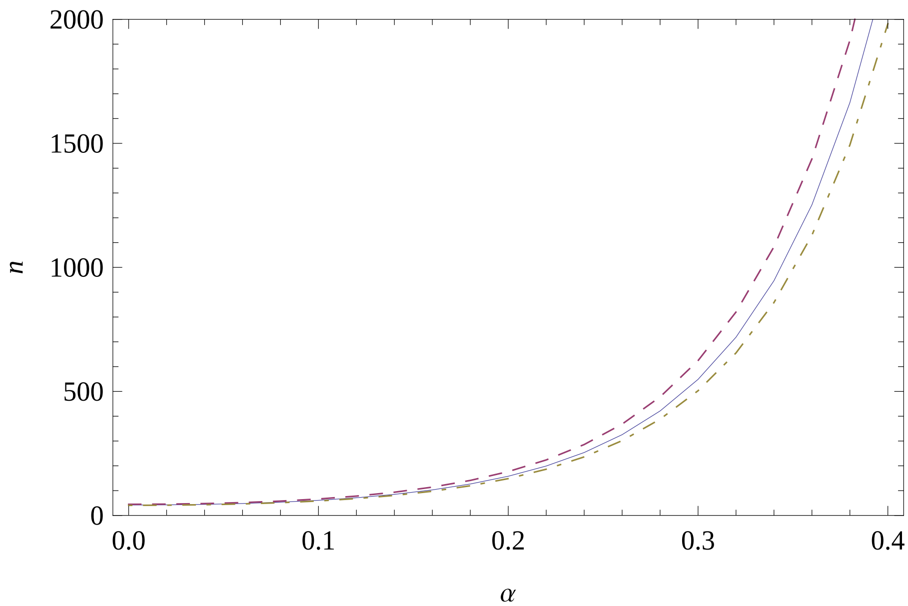

Figure 4.7: Plot of $E(C_j)$ and Approximation (light tail $m = 20$ $j = 18$)

α	$E(C_j)$	App	C App	RError1	RError2
0.30	548.533	624.154	508.579	0.138	-0.073
0.32	718.588	820.047	663.073	0.141	-0.077
0.34	946.377	1083.081	869.069	0.144	-0.082
0.36	1252.352	1437.249	1144.475	0.148	-0.086
0.38	1664.295	1915.371	1513.619	0.151	-0.091
0.40	2220.361	2561.419	2009.595	0.154	-0.095

Table 4.7: Approximation value and relative error (light tail $m = 20$ $j = 15$)

When j=18, according to the relative error shown in the table, the corrected approximation shrinks the relative error about a half compared to the original one, so we can treat it as a good improvement of the original approximation for the coupon collector's problem.

Example 4.3.3: full collection of the coupons

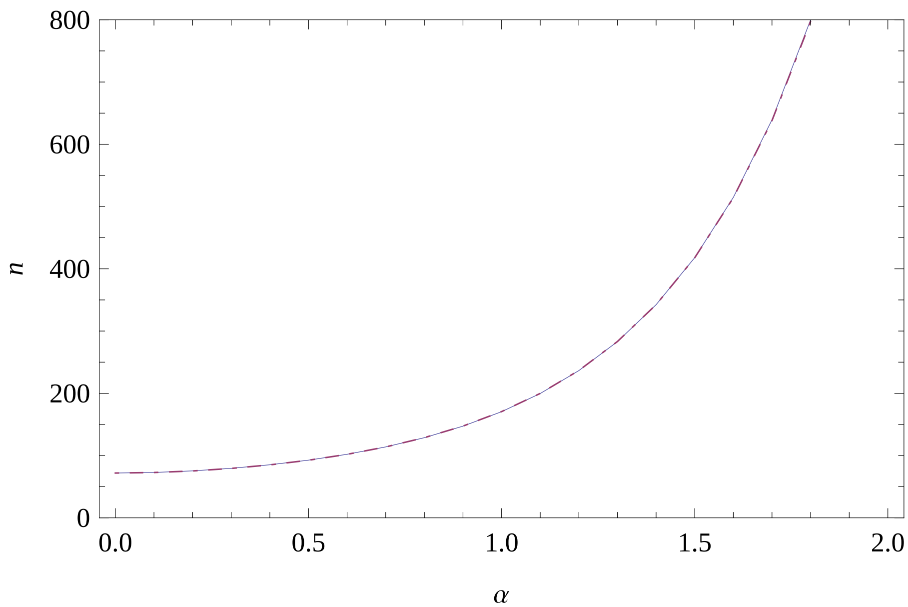

Figure 4.8: Plot of $E(C_j)$ and Approximation (heavy tail $m = 20$ $j = 20$)

α	$E(C_j)$	C App	RError2
1.50	417.933	417.733	0.000
1.60	514.767	514.179	-0.001
1.70	639.250	638.023	-0.002
1.80	799.822	797.586	-0.003
1.90	1007.605	1003.818	-0.004
2.00	1277.276	1271.156	-0.005

Table 4.8: Approximation value and relative error (heavy tail $m = 20$ $j = 20$)

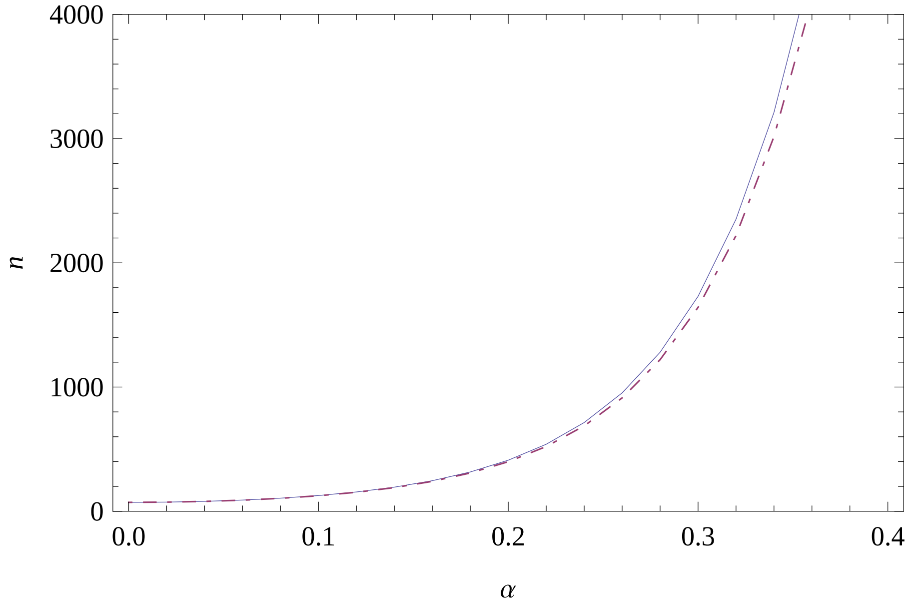

Figure 4.9: Plot of $E(C_j)$ and Approximation (light tail $m = 20$ $j = 20$)

α	$E(C_j)$	C App	RError2
0.30	1730.322	1641.144	-0.054
0.32	2351.042	2219.346	-0.060
0.34	3211.074	3015.856	-0.065
0.36	4404.358	4116.193	-0.070
0.38	6065.093	5640.253	-0.075
0.40	8382.083	7756.398	-0.081

Table 4.9: Approximation value and relative error (light tail $m = 20$ $j = 20$)

Since after the correction for the approximation, we can calculate the expected number of coupon needed to get a full collection of the coupons, Figure-4.8,4.9 and Table-4.8,4.9 show the approximation values and exact values for the heavy-tail case and light-tail case. The corrected approximation works well in the heavy-tail case, where it is shown in Table 4.9 that the relative error is less than 1 percent. While the relative error is a little larger in the light-tail case, it is still less than ten percent, so the result can be accepted.

Chapter 5

Conclusion and Discussion

The approximation found by Barry James, Kang James, and Anthony Gamst is a good, and easily calculated approximation for the coupon collector's problem. For different distributions of probability and different j, the relative error for the approximation is different. Basically, the relative error will increase when j is large and increasing. And the approximation works better for heavy-tail distributions rather than light-tail distributions.

For larger j, the corrected approximation shrink the relative error compared to the original approximation, and this method can also be used to find an approximation of the expectation of collecting the full set of coupon. As for the original approximation, this method also works better for heavy tails, rather than light tails.

References

[1] A. Hald *Commentary on 'De Mensura Sortis'* International Statistical Review, (1984)

[2] F. G. Maunsell *A Problem in Cartophily* The Mathematical Gazette, Vol. 22, No. 251, Oct, 1938 (1938)

[3] William Feller *An Introduction to Probability Theory and Its Applications, Vol.1, Third Edition* Wiley; 3rd edition (1968)

[4] Herman Von Schelling *Coupon Collecting for Unequal Probabilities* The American Mathematical Monthly, Vol. 61, No. 5, May, 1954 (1954)

[5] Donald J. Newman *The Double Dixie Cup Problem* The American Mathematical Monthly Vol. 67, No. 1, Jan., 1960, (1960)

[6] Bengt Rosen *On the Coupon Collector's Waiting Time* The Annals of Mathematical Statistics, Vol. 41, No. 6, Dec., 1970, (1970)

[7] Lars Holst *On Birthday, Collectors', Occupancy and Other Classical Urn Problems* International Statistical Review, Vol. 54, No. 1, Apr., 1986, (1986)

[8] Philippe Flajolet *Birthday paradox, coupon collectors, caching algorithms and self-organizing search* Discrete Applied Mathematics, Volume 39, Issue 3, 11 November 1992, (1992)

[9] Arnon Boneh, Micha Hofri *The coupon-collector problem revisited- a survey of engineering problems and computational methods* Computer Science Technical Reports. Paper 807. (1989)

[10] Sheldon M. Ross *Introduction to Probability Models, Tenth Edition* Academic Press; 10th edition (2009)

Appendix A

Proof of the exact value for CCP

Languages and strings:

Let $A = a_1, a_2, ..., a_m$ be a fixed set called the *alphabet*, letters $a_1, a_2, ..., a_m$ are the elements of A. Finite sequences of A called words or strings. A^* represents the set of all strings of A.Define L, a subset of A^*, be languages.

Generating functions:

If L is a language, we let $l_{n_1,...,n_m}$ be the number of words in L that have letter a_1 occurrences n_1 times,..., a_m occurrences n_m times. The generating function of L is

$$l(z_1, ..., z_m) = \sum_{n_1,...,n_m} l_{n_1,...,n_m} z_1^{n_1} z_2^{n_2} \cdots z_m^{n_m}$$

We add a fixed weight to each letters. Assume $p = (p_1, p_2, ..., p_m)$—the distribution over A is given, p_i is the weight of letter a_i. the weight of a word $w = a_{j_1} a_{j_2} \cdots a_{j_m}$ being taken as

$$\pi[w] = p_{j_1} p_{j_2} \cdots p_{j_m}$$

The function will be

$$\begin{aligned} l(p_1 z, ..., p_m z) &= \sum_{n_1,...,n_m} l_{n_1,...,n_m} p_1^{n_1} p_2^{n_2} \cdots p_m^{n_m} z^{n_1+n_2+\cdots+n_m} \\ &= \sum_{w \in A*} \pi[w] z^{|w|} \end{aligned}$$

it is called the ordinary generating function (OGF) of language L (with respect to weight p) and is denoted by $l(z)$. The exponential generating function (EGF) is defined by z^n

replaced with $z^n/n!$, denoted by $\hat{l}(z)$.

$$\begin{aligned}\hat{l}(z) &= \sum_{n_1,\ldots,n_m} l_{n_1,\ldots,n_m} p_1^{n_1} p_2^{n_2} \cdots p_m^{n_m} \frac{z^{n_1+n_2+\cdots+n_m}}{(n_1+n_2+\cdots+n_m)!} \\ &= \sum_{w \in A*} \pi[w] \frac{z^{|w|}}{|w|!}\end{aligned}$$

according to the classical relation

$$\int_0^\infty t^n e^{-t} dt = n!$$

Ordinary and exponential generating functions are related by the Laplace-Borel transform

$$l(z) = \int_0^\infty \hat{l}(zt) e^{-t} dt$$

Operations:

We define three operations on languages.

1. The union of L_1 and L_2, denoted by $L_1 + L_2$. Is the usual union of L_1 and L_2

2. The product of L_1 and L_2, denoted by $L_1 \cdot L_2$, is defined as

$$L_1 \cdot L_2 = \{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\}$$

3. The shuffle product defined as follows. If w_1 and w_2 are words, their shuffle, denoted by $(w_1 \in w_2)$, consist of all words obtained by merging their letters in all possible ways while preserving their order inside w_1 and w_2. It is defined by

$$(v \sqcup\!\sqcup\, \varepsilon) = (\varepsilon \sqcup\!\sqcup\, v) = \{v\})$$

$$(av_1 \sqcup\!\sqcup\, bv_2) = a(v_1 \sqcup\!\sqcup\, bv_2) \bigcup b(av_1 \sqcup\!\sqcup\, v_2)$$

with ε denoting the empty word. The shuffle of two languages L_1 and L_2 is

$$L_1 \sqcup\!\sqcup\, L_2 = \bigcup_{w_1 \in L_1, w_2 \in L_2} (w_1 \sqcup\!\sqcup\, w_2)$$

When three operations operate unambiguously on their arguments, the operations of union, product, star and shuffle product translate into generation functions.

1. $L = L_1 + L_(2) \Rightarrow l(z) = l_1(z) + l_2(z)$

2. $L = L_1 \cdot L_2 \Rightarrow l(z) = l_1(z) \cdot l_2(z)$

3. $L = L_1 \sqcup L_2 \Rightarrow \hat{l} = \hat{l_1} \cdot \hat{l_2}$

An operation on languages is unambiguous if every word of the resulting language is obtained only once.

The exact value:

The random variable C_j is number of trials until first times there is j different letters in a sequence of trials. Y_n is the number of different letters in a sequence of n trials. According to the definition for C_j and Y_n, the two probability distributions are related by

$$P(Y_n \geq j) = P(C_j \leq n)$$

so that we can get

$$P(Y_n < j) = P(C_j > n)$$

From this, the expectation of C_j is easily be written as

$$\begin{aligned} E(C_j) &= \sum_{n\geq 0} P(C_j > n) \\ &= \sum_{n\geq 0} P(Y_n < j) \\ &= \sum_{q=0}^{j-1} (\sum_{n\geq 0} P(Y_n = q)) \end{aligned}$$

Now we try to find the $\sum_{n\geq 0} P(Y_n = q)$. Let H_q be the language consist of words with exactly q different letters. Define α^k be the word $\alpha\alpha\cdots\alpha$, that only contain α and α repeated k times , so that

$$\alpha^{<k} = \epsilon + \alpha + \alpha^2 + \cdots + \alpha^{k-1}$$

and

$$\alpha^{\geq k} = \alpha^k \cdot \alpha^*$$

Language H_q is specified by

$$H_q = \bigcup_{I,J} (\alpha_{i_1}^{\geq k})(\alpha_{j_1}^{<k})$$

where the summation over all sets I, J of cardinality q and $r = m - q$ such that $I = i_1, ..., i_q, J = j_1, ..., j_r$ with $I \bigcap J = 0, I \bigcup J = 1, 2, ..., m$ if α is a letter with probability p, the EGFs of $\alpha^{<1}$ and $\alpha^{\geq 1}$ are

$$\hat{\alpha^{<1}}(z) = \hat{\epsilon}(z) = 1$$

and

$$\begin{aligned} \hat{\alpha^{\geq 1}}(z) &= \sum_{i \geq 1} p^i \frac{z^i}{i!} \\ &= \sum_{i \geq 0} p^i \frac{z^i}{i!} - 1 \\ &= e^{pz} - 1 \end{aligned}$$

Thus by theorem, the EGF of H_q is

$$\hat{H}_q(z) = \sum_{I,J} (e^{p_{i_1} z} - 1) \cdots (e^{p_{i_1} z} - 1) \cdot 1 \cdots 1$$

and noting the general expansion

$$[u^q] \prod_{j=1}^{m} (\lambda_i u + \mu_i) = \sum_{I,J} (\lambda_{i_1} \ldots \lambda_{i_q})(\mu_{j_1} \ldots \mu_{j_r})$$

Let $\lambda_{i_1} = e^{p_{i_1} z} - 1$ and $\lambda_{i_q} = 1$. we can express $\hat{H}_q(z)$ as

$$\hat{H}_q(z) = [u^q] \prod_{j=1}^{m} (1 + u(e^{p_i z} - 1))$$

Now the OGF of H_q is given by the Laplace-Borel transform,

$$H_q(z) = \int_0^\infty \prod_{j=1}^{m} (1 + u(e^{p_i z t} - 1)) e^{-t} dt$$

so that we have

$$\sum_{n \geq 0} P(Y_n = q) = H_q(1) = \int_0^\infty \prod_{j=1}^{m} (1 + u(e^{p_i t} - 1)) e^{-t} dt$$

and combine of equation and equation we get

$$E(C_m) = \sum_{q=0}^{j-1} \int_0^\infty \prod_{j=1}^{m} (1 + u(e^{p_i t} - 1)) e^{-t} dt$$

Appendix B

Code

Program 1 Calculate $E(c_j)$, Approximation and corrected Approximation

```
Table[Sum[20./(21 - i), {i, 1, j}], {j, 11, 19}],
Table[ Log[(20. - j)/20]/Log[19/20],  {j, 11, 19}],
Table[Log[(20 - j + (j - 1)/19/2)/20.]/Log[19/20], {j, 11, 19}],
Table[(Log[(20. - j)/20]/Log[19/20] - Sum[20./(21 - i), {i, 1, j}])/Sum[20./(21 - i), {i, 1, j}],
{j, 11, 19}],
Table[(Log[(20 - j + (j - 1)/19/2)/20]/Log[19/20] -
Sum[20./(21 - i), {i, 1, j}])/Sum[20./(21 - i), {i, 1, j}], {j, 11, 19}]

m = 20
t = Table[i/210, {i, 1, 20}];
f = Function[{j}, Total[1/(1 - Total[Subsets[t, {j}], {2}])]+ 0.];
a1 = Table[f[i], {i, 0, m - 1, 1}];
l = Function[{j}, Sum[(-1)^(j - q)*Binomial[m - q, m -j]*a1[[q]], {q, 1, j}] + 0.];
a = Table[l[j], {j, 1, m - 1, 1}];
g1 = Function[{j}, x /. FindRoot[m - Sum[(1 - t[[g]])^x, {g,1, m}] == j, {x, 0}]];
b1 = Table[g1[j], {j, 1, m - 1, 1}];
g2 = Function[{j}, x /. FindRoot[m - Sum[(1 - t[[g]])^x, {g,1, m}] == j - (j - 1)/(m - 1)/2, {x, 0}]];
b2 = Table[g2[j], {j, 1, m - 1, 1}];
BarChart[Transpose[{a, b1, b2}],
ChartStyle ->{Lighter[Blue, 0.7], Lighter[Red, 0.7], Lighter[Orange, 0.7]},
ChartLegends-> {"Exact", "Approximation", "CorrectedApproximation"},
ChartLabels -> {Table[i, {i, 1, 19}], None}, AxesLabel-> {j, n}, BarSpacing -> {-0.3, 1}]
```

Program 2 Calculate $E(C_j)$ $(m > 20)$

```
#include<iostream>
#include<string>
#include<list>
#include<stdio.h>
#include<math.h>
using namespace std;
double print( list<double> l);
double c=0;
double print(list<double> l){
    double b=0;
    for(list<double>::iterator it=l.begin(); it!=l.end(); ++it)
    b=b+*it;
    double a=1/b;
    c=c+a;
    return(c);
}
void subset(double arr[], int size, int left, int index, list<double> &l){
    if(left==0){
         print(l);
    }
    for(int i=index; i<size;i++){
        l.push_back(arr[i]);
        subset(arr,size,left-1,i+1,l);
        l.pop_back();
    }
}
int Factorial(int x) {
  if(x==0){return(1);}
  else{return (x == 1 ? x : x * Factorial(x - 1));}
}
int main(){
    double array[50]={the distribution of coupon (pi)};
    list<double> lt;
    double m[50];
    for (int j=0;j<50;j++){
        c=0;
        subset(array,50,j+1,0,lt);
        m[j]=c;
    }
   double mm[50];
   for (int i=0;i<50;i++){mm[i]=m[50-1-i];}
   double mmmm[51];
   for (int i=0;i<7;i++) {
       double mmm[51];
       for(int j=0;j<i;j++) {
           double k=pow(-1,i-1-j);
           mmm[j]=mm[j]*k*Factorial(50-j-1)/Factorial(50-i)/Factorial(i-1-j);
           mmmm[i]=mmm[j]+mmmm[i];
           }
    }
   for (int i=0;i<51;i++){
        cout<<mmmm[i]<<"\t";
        cout<<endl;
    }
    system("pause");
    return 0;
}
}
```

Program 3 Linear Interpolation method

```
int main ()
{
int number=200;
float a[number];
float error[number];
float b[number+1];
for(int i=0;i<=number;i++){
        b[i]=(1.0/number);
        }
for(int k=1;k<number; k++){
for(int j=1;j<=1000000; j++){
double c=0;
for(int i=0;i<=number;i++){
        c=c+1-pow(1-b[i],j);

        }
if(c>=k){
        error[j-1]=j-1;
        double d=0;
        for(int i=0;i<=number;i++){
        d=d+1-pow(1-b[i],j);
        }
        a[j]=j-1+(k-d)/(c-d);
        printf ("%10d \n", j);
        break;
}
}
}
system("pause");
}
```