Report Number: WUCSE-2005-27

2005-01-01

Efficient Estimation of Tighter Bounds for Worst Case Execution Time of Programs

Kelly Leahy

In this paper, we will present a framework for the statistical analysis of the execution time of program units. We will show alternative methods for computing the distribution of the execution times and provide justification for the use of each of the methods presented. We will estimate the worst-case execution time (WCET) of the program units using several methods and compare the results of these methods. We will also present a new method for estimating the WCET, based on the theory of extreme value distributions.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2005-27

Efficient Estimation of Tighter Bounds for Worst Case Execution Time of Programs (Masters Thesis)

Authors: Leahy, Kelly

December 15, 2004

Abstract: In this paper, we will present a framework for the statistical analysis of the execution time of program units. We will show alternative methods for computing the distribution of the execution times and provide justification for the use of each of the methods presented. We will estimate the worst-case execution time (WCET) of the program units using several methods and compare the results of these methods. We will also present a new method for estimating the WCET, based on the theory of extreme value distributions.

SEVER INSTITUTE OF TECHNOLOGY

MASTER OF SCIENCE DEGREE

THESIS ACCEPTANCE

(To be the first page of each copy of the thesis)

DATE: December 15, 2004

STUDENT'S NAME: Kelly P. Leahy

This student's thesis, entitled Efficient Estimation of Tighter Bounds for Worst Case Execution Time of Programs has been examined by the undersigned committee of three faculty members and has received full approval for acceptance in partial fulfillment of the requirements for the degree Master of Science.

APPROVAL: _____ Chairman

_____

_____

WASHINGTON UNIVERSITY

SEVER INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

EFFICIENT ESTIMATION OF TIGHTER BOUNDS FOR WORST CASE EXECUTION TIME OF PROGRAMS

by

Kelly P. Leahy

Prepared under the direction of Ron K. Cytron

A thesis presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

Master of Science

May, 2005

Saint Louis, Missouri

WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

EFFICIENT ESTIMATION OF TIGHTER BOUNDS FOR WORST CASE EXECUTION TIME OF PROGRAMS

by Kelly P. Leahy

ADVISOR: Ron K. Cytron

May, 2005
Saint Louis, Missouri

In this paper, we will present a framework for the statistical analysis of the execution time of program units. We will show alternative methods for computing the distribution of the execution times and provide justification for the use of each of the methods presented. We will estimate the worst-case execution time (WCET) of the program units using several methods and compare the results of these methods. We will also present a new method for estimating the WCET, based on the theory of extreme value distributions.

to my loving wife Nadia, for all of her patience and support

Contents

List of Tables

List of Figures

Acknowledgments

Thanks to the graduate students and faculty of Washington University that assisted me with programming tasks and other items related to this thesis, as well as acting as a sounding board for the various ideas herein. In particular, I'd like to thank graduate students Scott Friedman and David Olliges and faculty member Jeremy Buhler for their assistance. A special thanks to my thesis advisor Ron Cytron for his patience and support and to Stuart Klugman of Drake University for his assistance with various items related to mixture distributions..

Kelly P. Leahy

Washington University in Saint Louis
May 2005

Chapter 1

Introduction

The estimation of worst case execution times (WCET) for a program requires a distributional assumption for the random variable representing the run time. This assumption may be explicit, or implicit, and may involve the use of one or more of the well known probability distributions. Much of the existing literature on this subject centers on the use of an extreme value distribution, such as the Gumbel or Weibull distributions (see [7]). Other existing literature takes a low level approach to estimation of the WCET by attempting to model the precise architecture on which the task is running and simulate the program itself (see [3], [19], [23], [13], [9]). Ernst and Ye [10] present a model using basic blocks as the components of the execution time, but do not perform a statistical analysis of these times. Instead, they compute their estimates of the WCET deterministically by simply summing the estimates. We propose a method in which the distributions of the individual elements are computed, and then the total distribution is computed. The total distribution is then used to estimate the WCET.

The disadvantage of low-level methods is that they require an intimate knowledge of the target architecture and the source program. Often it is simply not feasible

to build such a model for each target architecture. On the other hand, a disadvantage of the methods using extreme value distributions is that they often give extremely high WCET estimates. These high estimates can lead to wasted cycles, since the tasks will often complete in a much shorter amount of time than that prescribed by the WCET model.

We present a method for estimating a more aggressive bound on the WCET that is more reasonable for a single scheduling of a given task. This more aggressive bound may cause the deadlines to be missed from time to time, but allow the scheduler to schedule many more tasks than the bounds given by the extreme value distributions. We also describe some methods of using a combination of the extreme value WCET bound and our single-sample WCET bound. Using a linear combination of the bound provided by the extreme value distribution and that provided by our method can alleviate some of the concern that our bound is too small. Of course, a scheduler should recalculate the deadline from time to time, if it finds that the deadlines are being missed with a greater frequency than is desired.

We describe a model in which a task is composed of several components that are assumed to have independent execution times (from each other). These components may represent separate function calls, basic blocks, loop iterations, or any other high-level components that satisfy the assumption of independence. Of course, it is possible that some components do not satisfy the independence assumption. We will assume, for the purposes of this thesis, that this assumption is satisfied.

1.1 The Model

Our model is one for describing the execution times of a given task. For such a task, we will assume that there are several components that are executed to perform this

task, and that that the execution time for the task is given by an equation similar to

$$T = \sum_{i=1}^{M} T_i \tag{1.1}$$

where T is the random variable representing the total execution time for a task, and the T_i are the execution times for the individual components. We assume that M is a constant, and that the individual T_i are mutually independent random variables (they need not be identically distributed).

We make no specific assumptions about the individual distributions of the T_i; instead, we provide a method for estimating the distribution of T from estimates of the distributions of the individual T_i. We describe methods for estimating the T_i in systems that can have multiple environment states that greatly affect the execution time of some components. An example of one such system is that in which the main processor has a data and/or memory cache. In such a system, each T_i may be dependent on the cache state when the component is to be executed (cold cache / hot cache). In this case, the assumption of independence of the T_i may be questionable, but we still assume that they may be independent, since the data cache and instruction cache state of one component is often independent of the cache states of another.

1.2 Our Methods

We describe several methods for computing the distribution of T, as given in Equation 1.1. Because T is a sum of a fixed number of random variables, its probability function (pf) is given by a convolution of the probability functions of the individual random variables on the right-hand side of Equation 1.1. Analytically, the convolution of two random variables X and Y, given by probability functions f_X and f_Y

respectively, is written as $f_X * f_Y$, and is defined as

$$f_{X+Y}(z) = (f_X * f_Y)(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z-x)\,\mathrm{d}x \tag{1.2}$$

where z is the value of the sum $X + Y$ at which the pf should be evaluated.

For some simple distributions (Gaussian, Gamma family), a closed-form expression for the convolution exists, permitting direct computation of the composite distribution (of T) when all components are of applicable type. For more complicated models, however, the composite distribution must be numerically estimated. It is these situations with which we are most concerned.

For distributions of the discrete type (we will use these as numerical approximations of those of the continuous type), the convolution of two random variables X and Y, given by probability functions f_X and f_Y respectively, is given by

$$f_{X+Y}(z) = (f_X * f_Y)(z) = \sum_{z=x+y} \sum f_X(x) f_Y(z-x) \tag{1.3}$$

where the summation is over all possible values of x and y that sum to z.

As can be seen from this formula for the calculation of the convolution, its apparent complexity is $O(\bar{X}\bar{Y})$, where $\bar{X}$ and $\bar{Y}$ denote the number of possible values for X and Y respectively. We next describe a few methods that can allow us to reduce the complexity of this computation, using Fourier analysis.

1.2.1 Convolution using Discrete Distributions

Our first method of computing the composite distribution of T involves an approximation of the distributions of the T_i by use of a discrete distribution. We assume that this is, in fact, how the data was already gathered when timing was performed on the

different components. For this reason, this is computationally the simplest method we have at our disposal. For our purposes, we assume that the discrete distribution is given as a frequency distribution. This method is described in Chapter 3. The input to this method is the sample data (as frequency distributions) corresponding to the components represented by the T_i, and the output is the approximate distribution of the random variable T (as a frequency distribution).

1.2.2 Convolution by Discrete Fourier Transform

Due to the computational complexity ($O(N^M)$ if each T_i has N samples) of the discrete convolution method described in the previous subsection, we are interested in other methods to compute the distribution of T. One such method, also using the discrete distributions for the T_i, is the use of the Fast Fourier Transform (FFT) of the probability functions to perform the convolution. The Fourier transform (the FFT is a fast implementation of the discrete Fourier transform (DFT)) transforms the input function from the time domain to the frequency domain. The advantage of this is that the convolution operation in the time domain is transformed into multiplication in the frequency domain. This property is known as the Fourier convolution theorem. Stated in the typical Fourier notation, we have

$$f_T(t) = (f_{T_1} * f_{T_2} * \cdots * f_{T_M})(t) = \mathcal{F}^{-1}\left[\prod_{i=1}^{M} \mathcal{F}\left[f_{T_i}\right]\right] \tag{1.4}$$

where $\mathcal{F}[f]$ represents the Fourier transform of a function f, and $\mathcal{F}^{-1}[\varphi]$ represents the inverse Fourier transform of a function φ. In this method (with the discrete distributions), we use the DFT, rather than the continuous Fourier transform (CFT). However, the convolution theorem is also true for the DFT, so we can use it for

the computation of the convolution of the individual T_i probability functions. This method is presented in Chapter 4.

1.2.3 Convolution by Characteristic Function

One of the disadvantages of convolution by the use of the DFT is that we must use discrete distributions as the input to the method. If, on the other hand, we wish to approximate the distributions of our components (T_i) using continuous distributions (standard or otherwise), we need to be able to compute the CFT of each probability function (for each T_i) and multiply these CFTs together to compute the CFT of the convolution. It turns out that the Fourier transform of the probability function of a distribution is nearly identical to the characteristic function (often well known and with many convenient properties) of the same distribution. We use the definition of the characteristic function for the distribution of a random variable T with distribution function $F(t) = P(T \leq t)$ from [27] (paragraph 4.1), written as

$$\varphi(\omega) = \int_{-\infty}^{\infty} e^{i\omega t} \, \mathrm{d}F(t) \tag{1.5}$$

whereas we use the definition of the Fourier transform from [17] (page 317), written as

$$\varphi(\omega) = \int_{-\infty}^{\infty} e^{i\omega t} f(t) \, \mathrm{d}t. \tag{1.6}$$

One may immediately see that these two functions are, in fact, identical, and so the characteristic function of a distribution may be used interchangebly with the continuous Fourier transform of the probability function of that distribution. It should be noted that most literature defines the CFT with $-itx$ in the exponent of the exponential factor of the transform, rather than itx as we have here. This point will

be important in Chapter 5 when we must produce input for the inverse FFT algorithm. Once we have the CFT of the pf of the distribution, we then must use some method to compute the inverse Fourier transform of this convolution. We describe a method by which we compute the product of the CFTs of the individual distributions of the T_i, then compute the inverse DFT of a discretized version of the convolution's CFT. This has the advantage of being more accurate than the fully-discrete method, in that sampling and truncation is performed only at the final step of the computation, rather than prior to the use of the DFT. It also allows us to quickly ($O(N)$) compute the CFT of the convolution, then compute the inverse DFT of the convolution using the FFT for a total cost below that of the fully-discrete method described above. This method is described in Chapter 5.

1.3 Examples

We will use one running example throughout all chapters that will illustrate all of the methods described herein. The example will be presented in parts. The data for the example will be presented in Appendix B, and each of the method chapters will cover the application of one of the methods to this example. The example involves a task built from 4 components. The first two of these components are dependent on the system environment (at the start of their respective execution); the other two are not. We will model the first two components with mixture distributions (described in Chapter 2) and the other two components with standard distributions.

In order to test our methods, we will use simulation to generate execution times consistent with our model of the task. We will train the system with these samples (that is, we will fit our distributions to these samples), and then we will

generate separate samples (from the same model) to test the ability of our estimators to predict the WCET.

1.4 Mathematical Preliminaries

Much of the theory put forth here requires some knowledge from Probability and Statistics, Fourier Analysis, and general Mathematics. We will present references to some of this knowledge, and will repeat the most important items in the chapters in which they are first used. For a general review of probability and statistics, we refer the reader to one of [15], [12], or [27].

1.5 Our Contribution

While others have investigated the WCET by use of extreme value distributions, we present methods for estimating the run-time distribution of the program based on a breakdown of several independent components. We then show how to proceed from this estimate of the run-time distribution to a WCET estimate using the non-asymptotic extreme value distribution. We show that this method produces a tighter bound on the WCET than that provided by the asymptotic Gumbel distribution. We also present several methods that can allow the calculation of the convolutions necessary in our model to be performed in a manner that is less computationally intensive than the traditional methods for computing convolutions. The CFT method we present here is not found elsewhere in Computer Science literature, so far as the author can tell.

Chapter 2

Estimating the Component Distributions

2.1 Introduction

The first step in the estimation of the total execution-time distribution of a task is the estimation of the distributions of the individual components that comprise the task. The time distributions for these components may be analytical distributions, estimated by the programmer based on hardware specifications, or they may be based on samples gathered during profiling of the task in question. For our purposes, we assume that the data come from the latter source. We do not concentrate on the methods of gathering this data, but rather refer the reader to [22] or [11] for ideas on how to gather timing samples for real-time analysis.

We use a simple program as the example for our task, given in Figure 2.1. This program has four main parts, which we call parts A, B, C, and D. We assume that timing data has been gathered from all of these parts. We present this timing data in Figure B.1, in Appendix B.

```
program example;
begin
    procA;                                  (part A)
    procB;                                  (part B)
    procC;                                  (part C)
    procD;                                  (part D)
end.
```

Figure 2.1: An Example Program

The first two parts (represented by `procA` and `procB`) are assumed to be dependent upon the system environment. The others (`procC` and `procD`), are assumed to be independent. The dependent parts will be modeled with mixture distributions, while the independent parts will be modeled with standard distributions. We determine the standard distributions that best fit parts C and D using the method of maximum likelihood estimation (MLE) of the parameters of the distributions, using the distribution with the greatest value of the likelihood function as our choice of distribution (see [28], [26], [8]). For simplicity, we will restrict our analysis to a relatively small number of continuous probability distributions. We will use three different distributions: Gamma, Normal, Log-Normal. In practice, one might use a catalog of many distributions, but the process we describe for choosing the distribution is the same, regardless of the number of distributions in our catalog.

2.2 Fitting the Simple Distributions

In order to choose the distribution to use for each of the two parts C and D, and to estimate the values of the parameters of these distributions, we must compute the MLEs of the parameters for each distribution based on the data samples. During the process of computing the MLEs, we also compute the value of the likelihood function

for these parameter values. This value is the maximum value of the likelihood function over all possible parameter choices, given the data as observed. First, we review the process of maximum likelihood estimation.

2.2.1 Maximum Likelihood Estimation

The process of computing the MLE estimate of the parameters of a distribution involves maximizing the likelihood function over all choices of parameter values. The likelihood function is defined as

$$L(\Psi; \mathbf{x}) = \prod_{i=1}^{N} f_X(x_i; \Psi) \tag{2.1}$$

where Ψ is the vector of parameters to the pf of the distribution, f_X is the pf of the distribution, and $\{x_i\}_{i=1}^{N}$ is the sample data from which Ψ is estimated. The MLE estimate of Ψ is the value of Ψ that maximizes the value of L over all possible values of Ψ. As it is often more convenient in optimization problems, we may instead maximize the log-likelihood function, defined as the logarithm of L written as

$$\ell(\Psi; \boldsymbol{x}) = \log L(\Psi; \mathbf{x}) = \sum_{i=1}^{N} \log f_X(x_i; \Psi). \tag{2.2}$$

The Normal Distribution

The MLE estimators for the Normal (Gaussian) distribution with pf

$$f_X(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2.3}$$

are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{2.4}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2. \tag{2.5}$$

Note that the estimator for σ is not the same as the traditional (unbiased) estimator s often used in statistical literature ([27], [28], [17]). We are working within the MLE framework, so we use the MLE estimator as our estimate of the variance.

The Log-Normal Distribution

The Log-Normal distribution is a distribution for which the logarithm of the random variable is normally distributed. It is given by the pf

$$f_X(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log x-\mu}{\sigma})^2} \tag{2.6}$$

with parameter MLE estimates given by

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} \log x_i, \tag{2.7}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(\log x_i - \hat{\mu})^2. \tag{2.8}$$

See Appendix A for derivation of the MLE of the Log-Normal distribution.

The Gamma Distribution

The Gamma distribution is defined by the pf

$$f_X(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta} \tag{2.9}$$

where $\Gamma(\alpha)$ is the well-known Gamma function (see [15], [31], or [17]) from advanced calculus.

Unfortunately, the Gamma distribution does not have a closed form expression for the MLE estimates of its parameters. We describe a method for estimating the parameters based on the Newton-Raphson method ([4], [29], [14], and [24]) in Appendix A.

2.2.2 MLE Estimates for the Training Data

Now that we have MLE estimators defined for the three different standard distributions we will be using, we may compute the MLE estimates of the parameters for each of the three distributions for each of our two data sets (for parts C and D), and compute the loglikelihood function for each of the distributions at these choices of parameters. The first step in computing these results is finding the values of the statistics used in our analysis. These statistics are the sample variance, the sample mean, the log-sample variance, and the log-sample mean (these are the terms we use for the sample variance and mean of the logarithm of the observations). We denote

these respectively by S_X^2, $\bar{x}$, $S_{\log X}^2$, and $\overline{l(x)}$, given by

$$S_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \qquad S_{\log X}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\log x_i - \overline{l(x)}\right)^2$$
$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \overline{l(x)} = \frac{1}{N}\sum_{i=1}^{N}\log x_i.$$

The values of these statistics are tabulated in Figure 2.2.

Table 2.2: Statistics for MLE Estimation

Part	S_X	$\bar{x}$	$S_{\log X}$	$\overline{l(x)}$
C	18.7735 6302	103.1843	0.1861 2334	4.6196 0194
D	14.6269 7266	148.7574	0.0968 7508	4.9975 8541

Normal and Log-Normal Distributions

Using these statistics, we can immediately estimate the parameters for the Normal and Log-Normal distributions. The estimates are given in Figure 2.3.

Table 2.3: Normal and Log-Normal Results (C, D)

	Normal			**Log-Normal**		
Part	$\hat{\mu}$	$\hat{\sigma}$	$-\ell(\Psi)$	$\hat{\mu}$	$\hat{\sigma}$	$-\ell(\Psi)$
C	103.1843	18.7736	435.1388	4.6196	0.1861	435.7195
D	148.7574	14.6270	410.1806	4.9976	0.0969	408.2191

Gamma Distribution

In order to estimate the parameters for the Gamma distribution, we use the iterative process from Section A.3.1. The results are tabulated in Figure 2.4.

Table 2.4: Gamma estimation using Newton's method

k	$\alpha^{(k)}$	$\beta^{(k)}$	$T_1^{(k)}$	$T_2^{(k)}$	$T_3^{(k)}$	$\delta^{(k)}$	$\text{NLL}^{(k)}$
Results for part C							
0	30.20883	3.41570	0.00027	0.00000	0.01673	0.49417	434.82077
1	29.71779	3.47122	-0.00027	-0.02726	0.01701	0.00783	434.81422
2	29.72562	3.47122	-0.00000	0.00000	0.01701	0.00006	434.81412
3	29.72568	3.47122	0.00000	0.00000	0.01701	0.00000	434.81412
4	29.72568	3.47122					434.81412
Results for part D							
0	103.43041	1.43824	-0.00011	0.00000	0.00485	2.35849	408.72488
1	105.78868	1.40544	-0.00052	-0.07733	0.00474	0.05207	408.71308
2	105.84075	1.40548	-0.00000	0.00000	0.00474	0.00305	408.71164
3	105.84379	1.40544	0.00000	-0.00000	0.00474	0.00000	408.71164
4	105.84379	1.40544					408.71164

Choosing the Distribution

Now that we have calculated the parameter estimates for each of our distributions in parts C and D, we can choose the distribution that best fits our data under the MLE criterion. The distribution (from our choices) that best fits the sample data will be the one with the smallest negative log-likelihood (NLL) or equivalently, the

largest likelihood. The NLL for each of the distributions for each part is tabulated in Figure 2.5.

Table 2.5: NLL for Parts C and D

Part	**Normal**	**Log-Normal**	**Gamma**	**Best Choice**
C	435.13882	435.71948	434.81412	Gamma
D	410.18058	408.21910	408.71164	Log-Normal

The best choice from the distributions in our catalog for parts C and D are Gamma and Log-Normal respectively. We take the parameters for these distributions from Figures 2.3 and 2.4. Our choices lead to the distribution assumptions given in Figure 2.6.

Table 2.6: Distributions for Parts C and D

Part	**Distribution**	**Parameters**
C	Gamma	$\hat{\alpha} = 29.72568$ $\hat{\beta} = 3.47122$
D	Log-Normal	$\hat{\mu} = 4.99759$ $\hat{\sigma} = 0.09688$

2.3 Fitting the Complex Distributions

We now have estimates of the distributions for Parts C and D – the simple distributions. Our goal now is to find applicable distributions and their parameters for the

environment-dependent parts of our program (Parts A and B). We will use a mixture model to approximate the distribution of Parts A and B.

A mixture model is a model that describes random variables that can come from one of a group of more than one distribution, where the distribution is randomly chosen. For example, consider an experiment in which we have two urns, each containing a mix of colored balls. In the first, we have 30% red balls and 70% black balls. In the second, we have 50% red balls and 50% black balls. The mixture model describes the probability of choosing a red (or a black) ball given that we randomly choose the urn from which to draw the ball. In this experiment, there is a 50-50 chance of drawing from a given urn, and a 30% chance of drawing red from the first urn (given it was the one selected) and a 50% chance of drawing red from the second (again, given it was the one selected). Our probability distribution (for the choice of a red ball) is given by

$$Pr(X = \text{red}) = 0.5(0.3) + 0.5(0.5). \tag{2.10}$$

In general, a mixture model is characterized by a pf f_Z of the form

$$f_Z(z;\Psi) = \sum_{i=1}^{N-1} \pi_i f_{X_i}(z;\Psi) + \left(1 - \sum_{i=1}^{N-1} \pi_i\right) f_{X_N}(z;\Psi) \tag{2.11}$$

or, if we define π_N such that $\sum_{i=1}^{N} \pi_i = 1$, then we can simply write

$$f_Z(z;\Psi) = \sum_{i=1}^{N} \pi_i f_{X_i}(z;\Psi). \tag{2.12}$$

We will use Equation 2.12 to describe the pf of a mixture distribution for the remainder of this document, under the understanding that π_N is a function of the π_i for $i = 1, 2, \ldots N-1$, rather than a parameter itself. The Ψ given in the equations is

the vector of parameters for all of the mixture components (each of the X_i is called a mixture component, with π_i known as its mixture proportion and f_{X_i} its pf).

The concept that the distribution of the program part (Parts A and B) depends on which state the system is in translates precisely to a mixture distribution. Each of the π_i represents the probability that the system is in state i with the f_{X_i} representing the distribution when the system is in state i. A mixture model is, in fact, a direct translation of the law of total probability. The law of total probability (see [15] for instance) states that if a set of events (A_i for $i = 1, 2, \ldots N$) is mutually exclusive and exhaustive[1] then

$$P(B) = \sum_{i=1}^{N} P(B|A_i)P(A_i) \tag{2.13}$$

where the notation $P(B|A_i)$ represents the probability of the event B occuring given that the event A_i is certain. To see the relationship between this law of total probability and the mixture pf given by Equation 2.12, we write $P(B) = f_Z(z)$, $\pi_i = P(A_i)$, and $P(B|A_i) = f_{X_i}(z)$. We have already assumed that the system may be in only one state at a time, and that the system must be in one of the states, so the A_i in our case are, of course, mutually exclusive and exhaustive events.

We refer the interested reader to [21] for an excellent discourse on the applications and theory of mixture models.

[1] This is the same as saying that their collective union is the sample space (all possible outcomes) and their intersection is the empty set. Written in mathematical notation this is

$$\begin{aligned} \bigcup_{i=1}^{N} A_i &= \Omega \\ \bigcap_{i=1}^{N} A_i &= \emptyset \end{aligned}$$

For the purposes of fitting the data from Parts A and B, we use a two-component mixture model, made up of components from each of our three distributions. We will not use "mixed" mixtures where the components are from different distributions, so we will have three possible models to fit. If the distributions of the components were likely to be different (as may very well be the case in a real system), we would need to fit each possible (or feasible) combination of component distributions. An example of such a mixture would be a distribution where 10% of the time, the distribution is a Normal distribution, and 90% of the time it is a Gamma distribution. We will not use such a mixture, though there is no theoretical reason why such a mixture could not exist. As in the previous section, the maximum likelihood criterion would be used to select which combination of mixture distributions provides the best fit to the data.

2.3.1 Maximum Likelihood Estimation

There are several ways in which the MLE estimates for the parameters of a mixture distribution may be obtained. All of these methods are numerical in nature, as there exist no closed form results for any of the distributions in which we are interested. We will concentrate on the use of the *EM* algorithm ([20], [21]) for the estimation of the mixture parameters.

In a mixture model, there are two types of explicit parameters: the component proportions, and the component parameters. The first of these refers to the π_i in the mixture pf. If the mixture has M components, there are $M-1$ of these parameters. The second type of parameter refers to the parameters of the component distributions themselves. If we assume that all component distributions are of the same type (as we will in this document), then the number of these parameters is MP where P is the

number of parameters for the component distribution. Therefore, we are estimating $3M - 1$ parameters for each of our types of component distributions (each has 2 parameters). In our case, $M = 2$, so there are 5 parameters to estimate for each type of component distribution.

The EM Algorithm

The EM algorithm is a two-stage algorithm for MLE in the face of missing data. While the reader may note that it doesn't appear that we are missing data (our samples aren't truncated or censored), our problem can be formulated as a missing-data problem by assuming that there are some data, denoted z_{ij}, associated with each of the data points x_i. These data z_{ij} are indicators that indicate which state the system was in for the execution corresponding to x_i. They are 0/1 variables, with $z_{ij} = 0$ indicating "not in state j during execution leading to time x_i" and $z_{ij} = 1$ indicating the alternative (*in* state j). Had we been able to gather this data, it would be quite simple to estimate the parameters of the components (since we would know the distribution to which each observation belongs) and the proportions (since we would know how many observations came from each system state).

For example, in the Normal distribution case, we would have

$$\hat{\mu}_j = \frac{\sum_{i=1}^{N} z_{ij} x_i}{\sum_{i=1}^{N} z_{ij}} \tag{2.14}$$

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^{N} z_{ij} (x_i - \hat{\mu}_j)^2}{\sum_{i=1}^{N} z_{ij}} \tag{2.15}$$

and

$$\hat{\pi}_j = \frac{1}{N} \sum_{i=1}^{N} z_{ij}. \tag{2.16}$$

The Process The process of applying the EM algorithm is as follows:

1. Compute an initial estimate of the parameters of the distributions based on an initial partitioning of the observations into groups corresponding to the mixture components.

2. Using the computed estimates for the component distribution parameters, compute the probabilities that each of the data points resulted from each of the mixture components. Use these probabilities to estimate the z_{ij} using the equation

$$\hat{z}_{ij} = \mathrm{E}[z_{ij}] = \frac{\pi_j f_{X_j}(x_i; \hat{\theta}_j)}{\sum_{k=1}^{M} \pi_k f_{X_k}(x_i; \hat{\theta}_k)} \tag{2.17}$$

   where $\hat{\theta}_j$ is the current estimate of the parameters for mixture distribution j, and M is the number of mixture components.

3. Using the estimates $\hat{z}_{ij}$ from step 2, estimate the parameters of the component distributions (see Equations A.5* and A.6* for the Normal distribution, Equations A.11* and A.12* for the Log-Normal distribution, or Equations A.13*, A.14* and A.15* for the Gamma distribution).

4. If the result has converged to within the desired tolerance, stop; otherwise return to step 2

2.3.2 MLE Estimates for the Training Data

For our purposes, we will initialize our estimates for the EM algorithm by simply dividing the training data in two halves, the fifty smallest and the fifty largest. We then use this division to estimate the parameters of our three distributions, for the starting point of the EM algorithm. The statistics computed on this data are given in Figure 2.7.

Table 2.7: Statistics for the EM Algorithm

Part	**Group**	S_X	$\bar{x}$	$S_{\log X}$	$\overline{l(x)}$	$\hat{\pi}_j$
A	Low	1.35859522	38.3052	0.03627907	3.64493752	0.5
	High	430.93016683	310.6080	1.42489530	4.62455229	0.5
B	Low	5.22430583	43.2008	0.13365915	3.75753905	0.5
	High	414.87919156	276.5026	1.21920932	4.67256012	0.5

The Normal Mixtures

First, we estimate the parameters for the Normal mixtures. The results for Part A are tabulated in Figure 2.8. The results for Part B are tabulated in Figure 2.9. The first row of each table (iteration one) is the initialization provided in Figure 2.7. The second and later iterations are the results of applying the EM algorithm to the initial estimates of the parameters given in row 1.

Table 2.8: Part A EM for Normal Components

Iter	π_1	$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\sigma}_1$	$\hat{\sigma}_2$	**NLL**
1	0.50000	38.30520	310.60800	1.35860	430.93017	433.64794
2	0.77469	41.06717	633.10545	2.54341	412.26863	349.04243
3	0.85860	39.86482	991.68282	2.26691	75.04354	306.90075
4	0.86000	39.83500	1001.41786	2.26794	20.22373	294.90933
5	0.86000	39.83500	1001.41786	2.26794	20.22373	294.90933

Table 2.9: Part B EM for Normal Components

Iter	π_1	$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\sigma}_1$	$\hat{\sigma}_2$	**NLL**
1	0.50000	43.20080	276.50260	5.22431	414.87919	518.24196
2	0.74908	49.99793	487.79806	7.38597	464.48401	443.34094
3	0.87641	49.40430	943.05710	8.30094	236.95495	422.04111
4	0.88999	48.94692	1057.08244	8.38641	31.95495	403.49570
5	0.89000	48.94685	1057.17273	8.38651	30.68959	403.47817
6	0.89000	48.94685	1057.17273	8.38651	30.68959	403.47817

The Log-Normal Mixtures

Now that we have estimated the Normal-component mixtures, we can move on to the Log-Normal and Gamma mixtures. The Log-Normal estimates are given by Figure 2.10 and 2.11.

Table 2.10: Part A EM for Log-Normal Components

Iter	π_1	$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\sigma}_1$	$\hat{\sigma}_2$	**NLL**
1	0.50000	3.64494	4.62455	0.03628	1.42490	422.93776
2	0.69615	3.92630	4.61232	0.26613	1.64867	490.01643
3	0.78568	3.80686	5.33674	0.13785	1.49541	430.57377
4	0.83998	3.70945	6.36724	0.06261	0.95026	352.38005
5	0.85951	3.68344	6.89583	0.05689	0.17912	318.42644
6	0.86000	3.68313	6.90897	0.05692	0.02004	294.63614
7	0.86000	3.68313	6.90897	0.05692	0.02004	294.63614

Table 2.11: Part B EM for Log-Normal Components

Iter	π_1	$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\sigma}_1$	$\hat{\sigma}_2$	**NLL**
1	0.50000	3.75754	4.67256	0.13366	1.21921	491.87104
2	0.64877	4.21344	4.21802	0.40855	1.55641	526.07859
3	0.72727	4.11043	4.49404	0.28738	1.64076	498.43958
4	0.79453	4.00911	5.01139	0.20661	1.62081	467.61674
5	0.83470	3.93301	5.63920	0.16739	1.51421	448.04447
6	0.85653	3.90063	6.09217	0.16192	1.38700	443.11100
7	0.86894	3.88811	6.38258	0.16395	1.23241	441.61787
8	0.87740	3.88290	6.59212	0.16795	1.03711	440.03889
9	0.88645	3.87777	6.84796	0.17535	0.57681	433.05540
10	0.89000	3.87542	6.96292	0.17781	0.02955	405.38911
11	0.89000	3.87542	6.96293	0.17781	0.02896	405.38472
12	0.89000	3.87542	6.96293	0.17781	0.02896	405.38472

The astute reader may notice that the NLL on the Log-Normal mixtures actually increases from the first to the second iteration. This is a result of the poor initial estimate given by the choice of the median as the "dividing line". One should notice that after the first iteration, the NLL decreases monotonically.

The Gamma Mixtures

As before, the process of estimating the Gamma mixtures is a bit more complicated than that of the Log-Normal and Normal mixtures, due to the fact that the Gamma MLEs are not closed-form. However, the general idea is the same, only the MLE (the

M step) is replaced by the solve for the Gamma MLEs based on the initial estimates given by $\bar{x}$ and S^2. The results are tabulated in Figure 2.12 for Part A and Figure 2.13 for Part B.

Table 2.12: Part A EM for Gamma Components

Iter	π_1	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\beta}_1$	$\hat{\beta}_2$	**NLL**
1	0.50000	771.60823	0.56034	0.04964	554.31932	423.01425
2[†]	0.72282	482.01746	0.65914	0.08166	799.15633	423.00408
3	0.83418	353.32292	5.98702	0.11252	142.32138	358.87962
4	0.86000	308.97979	2405.30944	0.12892	0.41634	331.93477
5	0.86000	308.97919	2477.36193	0.12892	0.40423	294.67962
6	0.86000	308.97919	2477.36193	0.12892	0.40423	294.67660

Table 2.13: Part B EM for Gamma Components

Iter	π_1	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\beta}_1$	$\hat{\beta}_2$	**NLL**
1	0.50000	60.26261	0.64396	0.71688	429.38127	496.82920
2	0.68900	56.31266	0.62847	0.83627	651.84801	492.82888
3[†]	0.81232	45.11140	1.66844	1.08018	384.08715	455.59212
4[†]	0.87498	36.67997	7.92605	1.33867	117.95610	440.82549
5	0.89000	32.82081	1188.05476	1.49134	0.88797	427.51405
6	0.89000	32.82080	1190.55002	1.49134	0.88797	404.19959
7	0.89000	32.82080	1190.55002	1.49134	0.88797	404.19958
8	0.89000	32.82080	1190.55002	1.49134	0.88797	404.19958

[†]The Gamma MLE solve for this iteration diverged for distribution 2, so $\hat{\alpha}_2$ and $\hat{\beta}_2$ are estimates using the Method of Moments.

Choosing the Distribution

As before, we choose the distribution family that provides the best MLE (lowest NLL) as the distribution to represent the data. The NLL values for the mixtures we have selected as candidates for our estimates are given in Figure 2.14.

Table 2.14: NLL for Parts A and B

Part	Normal	Log-Normal	Gamma	Best Choice
A	294.90933	294.63614	294.67660	Log-Normal
B	403.47817	405.38472	404.19958	Normal

2.4 Summary of Results

In the previous two sections, we estimated the parameters for the four distributions of Parts A, B, C, and D from our example program. Two of these distributions were estimated as mixtures, and two as simple distributions. Parts A and B were estimated as mixtures, with the parameters given by Figure 2.15. Parts C and D were estimated as simple distributions, with the parameters given in Figure 2.16.

Table 2.15: Distributions for Parts A and B

Part	**Distribution**	**Parameter Estimates**			
A	Log-Normal	$\hat{\pi}_1 = 0.86000$		$\hat{\pi}_2 = 0.14000$	
		$\hat{\mu}_1 = 3.68313$		$\hat{\mu}_2 = 6.90897$	
		$\hat{\sigma}_1 = 0.05692$		$\hat{\sigma}_2 = 0.02004$	
B	Normal	$\hat{\pi}_1 = 0.89000$		$\hat{\pi}_2 = 0.11000$	
		$\hat{\mu}_1 = 48.94685$		$\hat{\mu}_2 = 1057.17273$	
		$\hat{\sigma}_1 = 8.38651$		$\hat{\sigma}_2 = 30.68959$	

Table 2.16: Distributions for Parts C and D

Part	**Distribution**	**Parameters**
C	Gamma	$\hat{\alpha} = 29.72568$
		$\hat{\beta} = 3.47122$
D	Log-Normal	$\hat{\mu} = 4.99759$
		$\hat{\sigma} = 0.09688$

Chapter 3

Discrete Convolution

In this chapter, we present the method of discrete convolution. This method is the simple brute force method of computing convolutions. We describe this method in the context of two different data sources (both based on the example data given in Appendix B). The first data source we consider is the raw data from the example data set. The second data source is the empirical pf of the distribution, smoothed by kernel density smoothing [27].

3.1 The Convolution Process

Consider a set of data points with associated frequencies. Each point in the data set can be represented by the ordered pair (f_i, x_i), where f_i is the frequency associated with the point x_i. In the case of a regular data sample that is ungrouped, we may let $f_i = 1$ for all i. A data set gathered in this way is known as a *frequency distribution* if

$$\sum_i f_i = 1$$

and the x_i are unique. The first condition can be satisfied by a process known as "normalization". When normalizing a frequency distribution, each f_i is divided by $\sum_i f_i$ in order to cause all f_i to sum to one. The second condition can be satisfied by grouping all x_i with a given value together $((f_j, y_j) = (\sum_{x_i = y_j} f_i\,, y_j))$.

The convolution of two frequency distributions F and G, given in the normalized and unique-point form described above, is then

$$Z = \left\{ \; (u, z) \;\middle|\; \big(\exists\,(v, z) \in F \oplus G\,\big) \wedge u = \sum_{(a,z) \in F \oplus G} a \; \right\} \tag{3.1}$$

where the operator $\oplus$ is defined such that

$$X \oplus Y = \big\{\, (f \cdot g,\; x + y) \mid (f, x) \in X,\; (g, y) \in Y \,\big\}.$$

An Example. Consider the two frequency distributions F and G defined as

$$F = \{(0.1, 1),\ (0.5, 2),\ (0.3, 3),\ (0.1, 5)\}$$

and

$$G = \{(0.3, 1),\ (0.2, 3),\ (0.5, 7)\}.$$

The convolution of these two frequency distributions is given by

$$\begin{aligned} Z = \{\, &(0.03, 2),\ (0.15, 3),\ (0.11, 4),\ (0.1, 5),\ (0.09, 6), \\ &(0.07, 8),\ (0.25, 9),\ (0.15, 10),\ (0.05, 12)\,\}. \end{aligned}$$

A simple algorithm for computing the discrete convolution Z of F and G is given by Algorithm 1. The result of this algorithm will be a set Z that satisfies the two requirements given above for frequency distributions.

Algorithm 1 Compute the convolution $Z = F * G$

Require: F, G frequency distributions
Ensure: Z is frequency distribution of $F * G$
  $Z \leftarrow \emptyset$
  for all $(f, x) \in F$ **do**
    for all $(g, y) \in G$ **do**
      if $\exists (u, x + y) \in Z$ **then**
        $Z \leftarrow (Z - \{(u, x + y)\}) \cup \{(u + fg, x + y)\}$
      else
        $Z \leftarrow Z \cup \{(f \cdot g, x + y)\}$
      end if
    end for
  end for

3.2 Extension to M Distributions

The algorithm given in the previous section for computing the convolution is only for two frequency distributions. In order to compute the convolution of more than two distributions, we recognize that the output of the convolution operation is itself a distribution, so we perform the convolutions in series. For instance, if we wish to compute the convolution

$$Z = F_1 * F_2 * F_3$$

we notice that we may simply compute

$$Z_2 = F_1 * F_2$$

and then compute

$$Z = Z_2 * F_3$$

using Algorithm 1. It is easy to see that this can be extended to any finite number of distributions being convolved together.

In general, to convolve M distributions, the convolution operation must be performed $(M-1)$ times. Since the convolution operation can be shown to be $O(N_1 N_2)$ if there are N_1 points in the first distribution and N_2 points in the second, the convolution of M distributions is $O(N^M)$. This point is not obvious, but is apparent after recognizing that each convolution potentially produces an output that is the size of the Cartesian product of the elements of the two input sets.

In later chapters, we present other methods of computing the convolution that require less computation time than this method. One advantage of this method, however, is that it is the only method that can handle arbitrary frequency distributions without any sampling (approximation) error. The other methods will require points to be equally spaced on the number line. Such a requirement is not necessary for the brute-force convolution method.

3.3 Using the Results of Convolution

One of the disadvantages of discrete convolution, as described here, is that it has the tendency to magnify the effects of sampling artifacts in the component distributions. Sampling artifacts are due to the fact that we are approximating the real distribution of a component run-time with a discrete set of samples. These samples are not, in themselves, necessarily very representative of the source distribution. Without some form of smoothing, either by grouping, approximation by analytical function, or kernel methods, the distribution does not assign probability at values for which no observation was encountered.

As an example of sampling artifacts, consider an experiment in which a fair coin is tossed 50 times, with the number of heads recorded. If this experiment is

performed 10 times, resulting in the sample values

$$29 \quad 29 \quad 19 \quad 21 \quad 32 \quad 28 \quad 23 \quad 26 \quad 22 \quad 23$$

we obtain the frequency distribution

$$(19, 0.1),\ (21, 0.1),\ (22, 0.1),\ (23, 0.2),\ (26, 0.1),\ (28, 0.1),\ (29, 0.2),\ (32, 0.1)$$

which doesn't even include the most likely value (the mode – 25) of the true distribution. If we use this frequency distribution, without any smoothing or grouping, we would be led to believe that the distribution is bimodal with modes as 23 and 29, and no probability of the values 1 through 18, 20, 24, 25, 27, 30, 31, or 33 through 50.

If we convolve the distributions in their current form (as unsmoothed frequency distributions), the errors given by the original samples propagate to the entire distribution and produce even more holes than those found in the original frequency distributions. Consider, in reference to our example, that we would like to predict the number of "heads" given by two separate trials of the experiment described above. If we use the frequency distribution, we would have the convolution of this distribution with itself. This convolution is the frequency distribution given by Figure 3.1. In the table, $\hat{p}(x)$ is the empirical distribution, and $p(x)$ is the probability function of the true distribution of the convolution at these points. $\tilde{p}(x)$ is described later in this paragraph. Those points with values significantly close to zero or one are left out of the table. Here, p and $\hat{p}$ are the probability functions of the distribution (true and empirical). We measure the error by taking the sum of the squared differences between the empirical and actual probability functions. Using this error measure, the error in the approximation of p by $\hat{p}$ is 0.01135. If instead, we estimate (using

MLE) the parameter π of the binomial distribution (assuming the number of trials is known to be 50), we estimate from our original distribution of values the result $\hat{\pi} = 0.504$. Using this estimate to compute $\tilde{p}$ (the distribution of the convolution of two $b(50, 0.504)$ random variables), we have the $\tilde{p}$ given in Figure 3.1. The error given by the same measure applied to $\tilde{p}$ (versus p) is 0.00018. The reason for this substantial decrease in measured error is that the distribution does not have so many holes, since it was smoothed by using an analytical distribution before convolution occurred. Of course, often we are unable to determine the underlying analytical distribution of a dataset (if any is appropriate at all). In these more common cases, we may still attempt to smooth the empirical data by kernel methods as discussed in the next section. The entries in the fifth column of Figure 3.1 are described in the next section.

Table 3.1: Discrete Convolution (Binom. Example)

X	$\hat{p}(x)$	$p(x)$	$\tilde{p}(x)$	$\tilde{\tilde{p}}(x)$
≤ 28	0.00000	0.00000	0.00000	0.00000
29	0.00000	0.00001	0.00001	0.00000
30	0.00000	0.00002	0.00002	0.00000
31	0.00000	0.00005	0.00004	0.00000
32	0.00000	0.00011	0.00008	0.00000
33	0.00000	0.00023	0.00018	0.00000
34	0.00000	0.00046	0.00035	0.00000
35	0.00000	0.00086	0.00068	0.00000
36	0.00000	0.00156	0.00124	0.00150
37	0.00000	0.00270	0.00218	0.00200

Table 3.1: Discrete Convolution (Binom. Example)

X	$\hat{p}(x)$	$p(x)$	$\tilde{p}(x)$	$\tilde{\tilde{p}}(x)$
38	0.01000	0.00447	0.00368	0.00600
39	0.00000	0.00711	0.00594	0.00900
40	0.02000	0.01084	0.00921	0.01900
41	0.02000	0.01587	0.01370	0.02300
42	0.05000	0.02229	0.01955	0.03350
43	0.02000	0.03007	0.02680	0.03800
44	0.05000	0.03895	0.03527	0.04450
45	0.06000	0.04847	0.04460	0.04500
46	0.04000	0.05796	0.05419	0.04850
47	0.04000	0.06659	0.06327	0.05000
48	0.06000	0.07353	0.07098	0.05300
49	0.06000	0.07803	0.07654	0.06300
50	0.06000	0.07959	0.07933	0.07250
51	0.10000	0.07803	0.07903	0.07200
52	0.09000	0.07353	0.07568	0.06600
53	0.02000	0.06659	0.06964	0.05900
54	0.04000	0.05796	0.06159	0.04700
55	0.08000	0.04847	0.05234	0.04300
56	0.01000	0.03895	0.04274	0.04200
57	0.04000	0.03007	0.03352	0.03800
58	0.06000	0.02229	0.02526	0.03050
59	0.00000	0.01587	0.01827	0.02800

Table 3.1: Discrete Convolution (Binom. Example)

X	$\hat{p}(x)$	$p(x)$	$\tilde{p}(x)$	$\tilde{\tilde{p}}(x)$
60	0.02000	0.01084	0.01268	0.02300
61	0.04000	0.00711	0.00845	0.01600
62	0.00000	0.00447	0.00540	0.01250
63	0.00000	0.00270	0.00331	0.00800
64	0.01000	0.00156	0.00195	0.00300
65	0.00000	0.00086	0.00109	0.00200
66	0.00000	0.00046	0.00059	0.00150
67	0.00000	0.00023	0.00030	0.00000
68	0.00000	0.00011	0.00015	0.00000
69	0.00000	0.00005	0.00007	0.00000
70	0.00000	0.00002	0.00003	0.00000
71	0.00000	0.00001	0.00001	0.00000
72	0.00000	0.00000	0.00001	0.00000
$\geq$ 73	0.00000	0.00000	0.00000	0.00000

Since $\hat{p}$ was never smoothed, it is not really fair to compare the error in $\hat{p}$ to that of $\tilde{p}$. If instead, we compute a smoothed version of $\hat{p}$ by assuming that the resulting distribution is binomial[1], our results should be much better. This is relatively easy in this situation because a convolution of two instances of a binomial distribution with the same success rate (π) is simply a binomial distribution with the trials equal to the sum of the trials for the convolved distributions. Therefore, if we convolve

[1]The number of heads resulting from a fixed number (M) of tosses of a fair coin is represented by the Binomial distribution with parameters $n = M$ and $\pi = 0.5$.

$b(50, \pi)$ with $b(50, \pi)$ we get $b(100, \pi)$. We can then estimate π from our empirical distribution by using the MLE estimator of π. This gives us an estimate of π that is 0.504. It should come as no surprise that this value is the same as the estimate we obtained from the individual distributions, since the expected value of the sum of two random variables is the sum of the expected values, and the MLE estimate of π is based on the sample mean, divided by the number of trials. Before, we estimated p by $\mathrm{E}_X[X]/N$, now we estimate it by $2\mathrm{E}_X[X]/2N$.

3.3.1 Smoothing Using a Kernel

We were able to smooth the empirical distribution by estimating with an analytical distribution. However, often there is no obvious analytical distribution that is applicable to the data. In this case, we often want to smooth the empirical distribution. We can turn to a kernel function to help us smooth the distribution. One use of a kernel function is to smooth the distibution so that there are no "holes" in the empirical distribution. Holes are points of zero probability between points of non-zero probability, where we would expect the true distribution to have non-zero probability. For instance, in the empirical distribution of Figure 3.1, there is zero probability of the value 39, while there is non-zero probability for 38 and 40. In the real distribution, we can see that there is probability of 0.00711 for the value 39. Even without knowing the true distribution, we might suspect that the distribution should be somewhat bell shaped. Such an assumption leads to the belief that there should be no holes in intermediate values of the probability function.

There are many classes of kernel functions [30] [25]. The main requirements of a kernel function are that the sum of the kernel function over all values is equal to

one. For illustrative purposes, we use a simple discrete kernel function defined as

$$K_j(u) = \{(-2, 0.15), (-1, 0.2), (0, 0.3), (1, 0.2), (2, 0.15)\} \tag{3.2}$$

where u is the signed distance between x and the point x_j $(x - x_j)$. The kernel function is used to weight nearby frequencies. This particular kernel function eliminates holes that are within 2 units of a non-zero probability point. For instance, to compute the estimate of F at $x = 39$, we use the points $x = 37 \ldots 41$. The estimate is defined by the formula

$$\tilde{\hat{p}}(39) = \sum_{i=-2}^{2} K_{(39)}(i)\hat{p}(39 + i)$$

where $K_{(39)}$ is the kernel function defined above.

The estimate at 39 is 0.009. Using this kernel function for an estimate of p, denoted $\tilde{\hat{p}}$, and the error measure of the previous sections, we have a total error of 0.00229. This error is, of course, the smaller of the two errors of the non-analytical estimates of p. Mostly, this is the result of the "filling" of the holes in the distribution. Due to the relatively small number of sample values in our empirical distribution, we see a marked increase in the accuracy of the estimate using smoothing. The larger the number of samples in the empirical distribution, the less the need for smoothing.

3.4 An Example

We now provide an example of the topics of this chapter using the data provided in Appendix B. Our first example will use the raw data from the Appendix, while our second will show a method for smoothing the frequency distribution prior to convolution. We will also contrast the effects of smoothing before the convolution to those of smoothing the convolution of unsmoothed empirical distributions. In both

the smoothed and unsmoothed versions of the distribution, we will use a discrete frequency distribution with values at the non-negative integers.

In order to confine our distribution to the non-negative integers, we will round each observation to the nearest whole number. Alternatively, we could scale the distribution, but since we are talking about the number of heads in a series of coin tosses, it makes sense to confine our results to whole integers). We then will count the number of observations at each whole number. We will then divide the count at each whole number by the total number of observations (100 for each distribution). The result will be the frequency corresponding to the point in the frequency distribution. After computing this frequency distribution, the result is shown in Tables B.2, B.3, B.4 and B.5 in Appendix B.

After preparing the unsmoothed frequency distributions of the four parts, we can compute the convolution of these frequency distributions. This convolution is computed in the traditional manner as described in this chapter. The result of this convolution is tabulated in Figure B.6 in Appendix B. The convolution p.f. is shown in comparison to the empirical probability function in Figure 3.2.

We now wish to compute the convolution of the smoothed versions of the individual probability functions. We will use the smoothing function defined by Equation 3.2 above. A comparison of the lower mode of part A's empirical probability function before and after smoothing can be found in Figure 3.3.

Once we have computed the smoothed versions of each of the probability functions (listed in Appendix B as Tables B.7, B.8, B.9 and B.10), we can compute the convolution of these smoothed distribution (shown in Figure B.11).

It can seen by the results of both the unsmoothed convolution and the convolution of the smoothed components that the distribution is much smoother and much more precise than the original empirical distribution of the total run times (the last

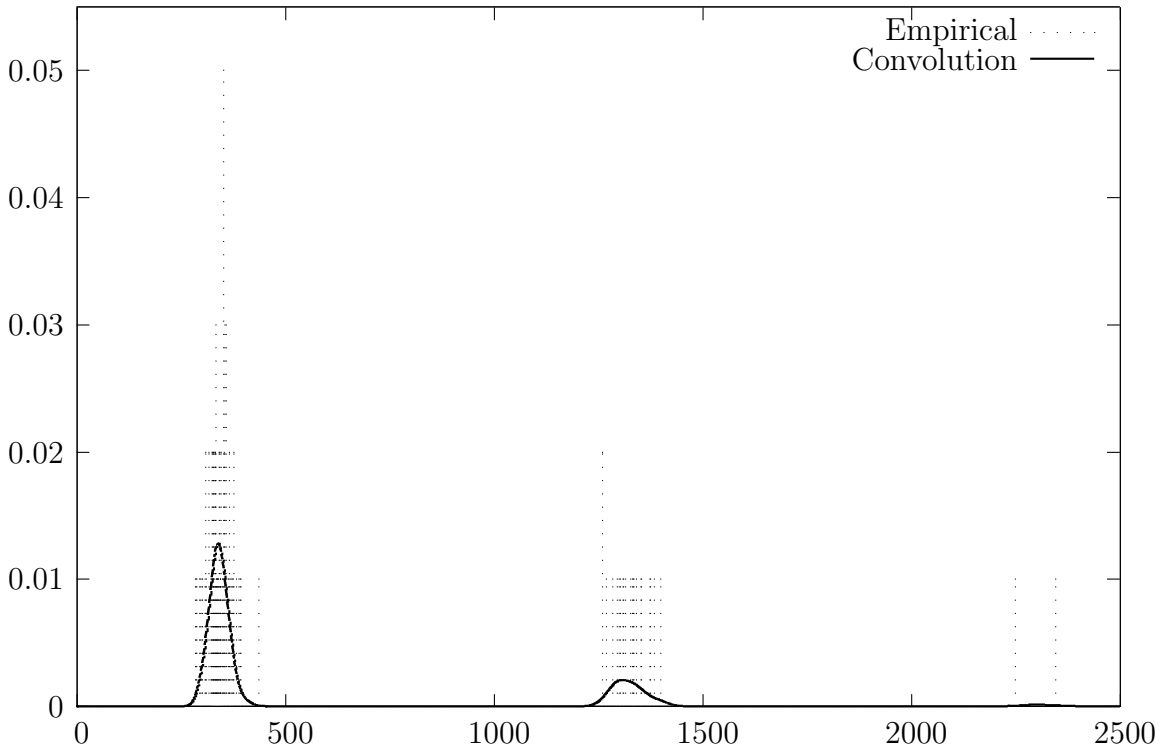

Figure 3.2: Unsmoothed Convolution Distribution

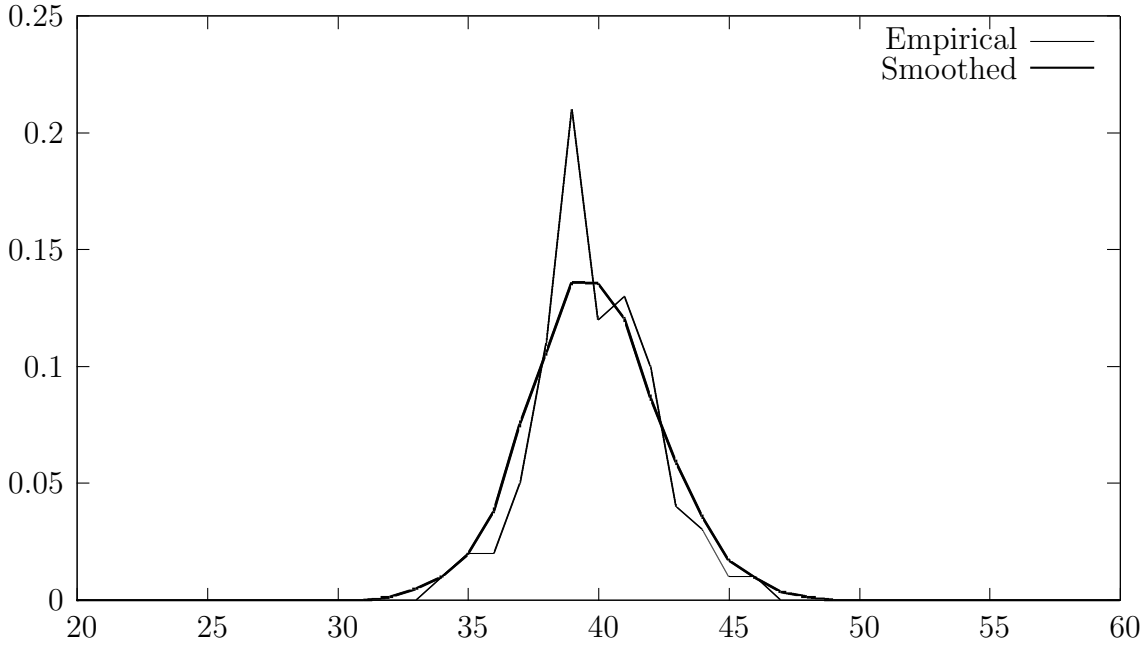

Figure 3.3: Smoothing Comparison for Part A

column in Figure B.1). In fact, from looking at the results of the unsmoothed convolution, one might suspect that smoothing is not even necessary in order to provide a relatively accurate representation of the total run-time distribution.

Chapter 4

Convolution by Discrete Fourier Transform

In the previous chapter, we described a method for computing the discrete convolution of two distributions in the traditional manner. In this chapter, we describe a method for computing the convolution that requires far less computation than the traditional method. Since this method is applicable to discrete distributions, it can be used on frequency distributions or discretized continuous distributions. This method is particularly applicable to frequency distributions built from large samples and smoothed frequency distributions. For frequency distributions built from smaller samples, smoothing should typically be performed before applying the methods described in this chapter. If such smoothing is not applied, the result of the convolution (by Discrete Fourier Transform) will suffer similar problems to those seen in the previous chapter.

4.1 The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) of a function f is defined by

$$\hat{\phi}_k = \hat{\mathcal{F}}\left[f\right](k) = \sum_{j=0}^{N-1} f_j \exp\left(\frac{2\pi i j k}{N}\right) \tag{4.1}$$

where $f_j = f\left(A + j\Delta x\right)$, $\Delta x = \frac{B-A}{N}$, and f is a periodic function defined over the interval $[A, B)$. While the definition of the DFT requires a periodic function, the theory can be applied to nonperiodic functions by a process of mirroring the function in other ranges. For instance, if the function f is defined over $[A, B)$, then we can make the function f a periodic function satisfying the requirements of the DFT by defining the function $\tilde{f}$ recursively as

$$\begin{aligned} \tilde{f}(x) &= f(x) && \text{for } x \in [A, B) \\ \tilde{f}(x) &= \tilde{f}(x - (B - A)) && \text{for } x \geq B \\ \tilde{f}(x) &= \tilde{f}(x + (B - A)) && \text{for } x < A. \end{aligned} \tag{4.2}$$

With these definitions, $\tilde{f}$ is a periodic function defined at the points $A \pm j\Delta x$ for all natural numbers j. The function $\tilde{f}$ corresponds to f within the interval $[A, B)$.

The DFT has several useful properties. The Discrete Fourier Transform operator is a linear operator. This means that

$$\hat{\mathcal{F}}[af + bg] = a\hat{\mathcal{F}}[f] + b\hat{\mathcal{F}}[g] \tag{4.3}$$

for any scalars a and b and functions f and g suitable for DFT application. These functions must, of course, have the same period. If they do not have the same period, one of the functions may be zero padded to the same range as the other and then

converted to periodic functions so that they have the same period. Another property of the transform is that the transform of a convolution is the product of the transforms of the convolved functions. This is known as the Fourier Convolution Theorem and has application to both the DFT and the Continuous Fourier Transform (CFT). The discrete form of this theorem can be stated mathematically as

$$\hat{\mathcal{F}}[f * g] = \hat{\mathcal{F}}[f]\hat{\mathcal{F}}[g] \tag{4.4}$$

where again, f and g are functions of the same period. The convolution operator $*$ is defined for two functions f and g as

$$(f * g)(x) = \sum_{t_1} \sum_{t_2} f(t_1)g(t_2)\delta(x - t_1 - t_2) \tag{4.5}$$

with $\delta(u)$ defined as

$$\delta(u) = \begin{cases} 1, & u = 0 \\ 0, & u \neq 0. \end{cases} \tag{4.6}$$

4.2 The Fast Fourier Transform

The Fast Fourier Transform (FFT) is a fast algorithm for the implementation of the DFT. It requires an input array of a size that is a power of 2. It is capable of computing the DFT for an array of size N on the order $O(N \log N)$. With the use of the Fourier Convolution Theorem, this means we can compute the convolution of M distributions on the order of $O(MN \log N)$ which is, of course, the same as $O(N \log N)$ as M is assumed to be fixed for a given problem.

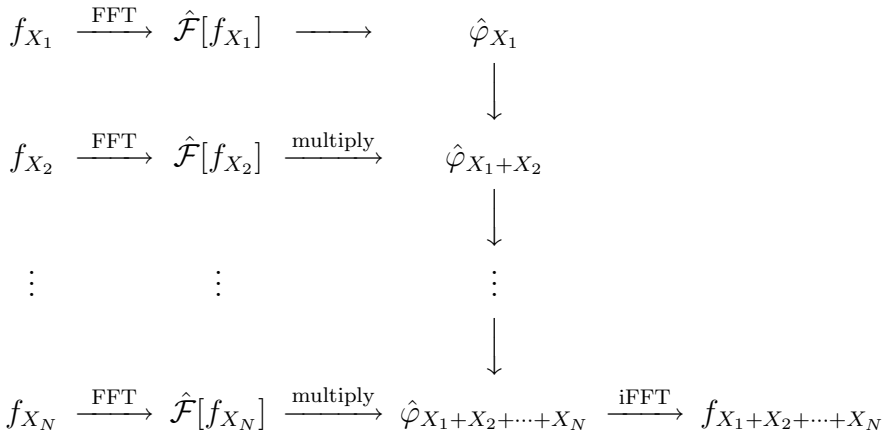

Figure 4.1: The DFT Method

4.3 The DFT Method

The method of convolution by DFT involves steps for building the input array to the FFT algorithm (for each distribution to be convolved), running the FFT algorithm, multiplying the results of the FFT together, and running the inverse FFT algorithm (nearly identical to the FFT algorithm) to compute the results of the convolution. The results are illustrated in Figure 4.1.

4.3.1 Step 1 — Prepare the FFT Input

As mentioned previously, the FFT algorithm requires an array of a length that is a power of two. Also, the points at which the FFT is evaluated must be equidistant. Since the results of the FFT will be used to evaluate the convolution of the distributions, the range of the arrays must be sufficient to represent all non-zero probability of the convolution distribution.

Consider, for instance, the distribution of the previous chapter. The initial distributions were approximately binomial and could have non-zero probability only in the range $[0, 50]$. The convolution (since we were only doing one convolution) could

have non-zero probability on the range $[0, 100]$ (from the sum of the minimum values to the sum of the maximum values). If we wish to represent only integral values (as we do), the smallest array we can use to represent the input items is the array with indices 0 through 127 (128 entries).

In order to prepare the input array for the binomial distribution, we build an array with 128 elements, zeroing all elements except those with probability. We would fill the array using the entries in Figure 3.1. After filling the array as appropriate, we would proceed to the other steps of the DFT method.

This step should be done for each input distribution. If convolving a distribution with itself, this step must be performed only once.

4.3.2 Step 2 — Compute the FFT

There are several commercially available and public domain implementations of the FFT algorithm. For a description of the algorithm, see [6], [5] or [24]. For the DFT method described in this chapter, the implementation used must only be compatible with the inverse implementation used. However, in the CFT method, attention must be paid to the FFT method used as there are some differences in the format of the output array between various implementations. We will discuss this further in Chapter 5.

This step should be performed for each of the outputs of Step 1. If the same distribution is being convolved multiple times, only one FFT computation is required for that distribution.

4.3.3 Step 3 — Multiply the FFT Output

Once the FFT is computed for each input distribution, there should be two arrays for each distribution (the real part of the complex value and the imaginary part). They should be the same length as the input arrays produced in Step 1. In this step, we must multiply the complex outputs of the FFT with each other, once for each distribution. For instance, if we have three distributions, the FFT results from the first must be multiplied (one entry at a time) with the FFT results from the second. This result (of the multiplication) must then be mupltiplied by the FFT results from the third distribution. When performing the multiplication, it should be performed as a functional multiplication, not a dot product. The results of the functional multplication will be a real / imaginary pair of arrays of the same length as the original input arrays.

This step will be performed one time for each distribution other than the first. Therefore, if there are M distributions to be convolved, it will be performed $M - 1$ times.

4.3.4 Step 4 — Compute the iFFT

Given the output of Step 3, a single pair of real / imaginary arrays, we compute the inverse FFT of these results to obtain the probability function for the resulting distribution. This probability function will be the discrete probability function corresponding to the convolution of the individual probability functions of the input distributions. As mentioned in Step 2, the implementation of the iFFT should come from the same source as the implementation of the FFT, but otherwise, the choice of implementation is most likely not significant. The output of the iFFT will be a

pair of real/imaginary arrays, but if everything was done with sufficient precision the resulting imaginary array should be zero in all elements.

4.4 An Example

Returning to our example dataset, we can compute smoothed discrete distributions from the sample points by grouping our data into integral values and computing frequency distributions over these values. We will use the Gaussian kernel function

$$K_h(\delta) = \exp\left(-\frac{\delta^2}{2h^2}\right) \tag{4.7}$$

with $h = 10$ for each of the four samples (for parts A through D). Because we are not covering the entire space of possible values for the kernel function (the kernel has a range of $(-\infty, \infty)$), we will need to normalize the distributions after applying the kernel. This has the effect of redistributing all probability from the tails of the distribution to the center of the distribution in proportion to the probability function at each point. For instance, if the tails of the distribution account for 20% of the total probability, then each point of the distribution will be "grossed up" by 25% ($\frac{1-0.8}{0.8}$). Figure 4.2 contains several example points from the smoothed distributions.

Since it will be necessary to have arrays of a length that is a power of 2 in order to perform the FFT step, and these arrays must be of sufficient size to avoid aliasing, we choose an array length of 4096 entries. The approximate maximum of $A + B + C + D$ is 4020 or less.

Table 4.2: Example Points from the Smoothed Distributions

X	$\hat{p}_A(x)$	$\hat{p}_B(x)$	$\hat{p}_C(x)$	$\hat{p}_D(x)$
37	0.032216	0.018057	0.000065	0.000000
100	0.000000	0.000011	0.020315	0.000257
150	0.000000	0.000000	0.001029	0.023462
990	0.002727	0.000190	0.000000	0.000000
1045	0.000760	0.001757	0.000000	0.000000

4.4.1 Computing the DFT of the Probability Functions

Now that we have computed the smoothed versions of the probability functions for the sample dataset, we can use the FFT algorithm to compute the DFT of each of the input arrays. We use the Fourier Analysis feature of Microsoft Excel® to compute the FFT. Several example points from this calculation are given in Figure 4.3. The Excel implementation, as do most, produces the array with positive angles first, followed by negative angles.

Table 4.3: Example Points from the DFTs

X	$\hat{\varphi}_A(x)$	$\hat{\varphi}_B(x)$	$\hat{\varphi}_C(x)$	$\hat{\varphi}_D(x)$
0	$1.0000 + 0.0000i$	$1.0000 + 0.0000i$	$1.0000 + 0.0000i$	$1.0000 + 0.0000i$
1	$0.8631 - 0.1924i$	$0.8817 - 0.1765i$	$0.9870 - 0.1575i$	$0.9737 - 0.2261i$
2	$0.7138 - 0.1145i$	$0.7704 - 0.1219i$	$0.9483 - 0.3106i$	$0.8963 - 0.4400i$
3	$0.8302 - 0.0181i$	$0.8825 - 0.0909i$	$0.8851 - 0.4550i$	$0.7721 - 0.6302i$

Table 4.3: Example Points from the DFTs

X	$\hat{\varphi}_A(x)$	$\hat{\varphi}_B(x)$	$\hat{\varphi}_C(x)$	$\hat{\varphi}_D(x)$
4093	$0.8302 + 0.0181i$	$0.8825 + 0.0909i$	$0.8851 + 0.4450i$	$0.7721 + 0.6302i$
4094	$0.7138 + 0.1145i$	$0.7704 + 0.1219i$	$0.9483 + 0.3106i$	$0.8963 + 0.4400i$
4095	$0.8631 + 0.1924i$	$0.8817 + 0.1765i$	$0.9870 + 0.1575i$	$0.9737 + 0.2261i$

4.4.2 Computing the DFT of the Convolution

Once we have calculated the DFT of each of the four distributions, we can convolve the distributions' probability functions by multiplying their DFTs. The multiplication performed is a complex product. Some example points from this product are given in Figure 4.4.

Table 4.4: Example Points from the Convolution DFT

X	$\hat{\varphi}_{A+B+C+D}(x)$	X	$\hat{\varphi}_{A+B+C+D}(x)$
0	$1.0000 + 0.0000i$	2048	$0.0000 + 0.0000i$
1	$0.5516 - 0.5717i$	4093	$0.2068 + 0.7008i$
2	$0.2604 - 0.4979i$	4094	$0.2604 + 0.4979i$
3	$0.2068 - 0.7008i$	4095	$0.5516 + 0.5707i$
2047	$0.0000 + 0.0000i$		

4.4.3 Computing the Probability Function

In order to complete our process of computing the estimate of the probability function of the sum distribution, we must invert the DFT of the convolution computed in the previous section. Fortunately, this is quite simple as all FFT implementations can be used also to invert the results of the FFT to compute the inverse DFT. In the previous section, we multiplied all of the four DFTs of the individual components together to compute the DFT of the convolution. We now compute the inverse DFT of that product in order to obtain the convolution probability function. We again use the FFT implementation built into MS Excel to compute this inverse transform, and give some results from the array in Figure 4.5.

Table 4.5: Example Points from the Probability Function of $A + B + C + D$

X	$\hat{p}_{A+B+C+D}(x)$	X	$\hat{p}_{A+B+C+D}(x)$
250	0.00015262	1380	0.00079850
300	0.00428857	1460	0.00003856
350	0.00913601	2200	0.00007509
1300	0.00181381	2300	0.00012591

4.5 Computational Cost of the Method

Referring to Figure 4.1, we see that there are M computations of the FFT algorithm in order to compute the DFTs of the input distributions. Since the cost of the FFT algorithm is $O(N \log N)$, and M is a constant for each given problem (not related to the number of points at which the distribution is evaluated), the computation of the

FFT of all M distributions is also $O(N \log N)$. This concludes step 2 of the method. We purposely ignore step one of the method as smoothing is optional and the cost of smoothing is very dependent on the choice of smoothing method. Additionally, smoothing would presumably be necessary for any method to be used if the sample size warrants smoothing. The other parts of step one (organizing the data into equally spaced buckets and appending zeroes as necessary to extend the array length to a power of two) are of negligible cost and so are also ignored.

The cost of computing step 3's product is on the order of $O(N)$ for each multiplication, and by the same argument as that for step 2 is in total $O(N)$. Therefore, the overall cost of the combination of steps 2 and 3 is also $O(N \log N)$ as $N \log N$ dominates N. The action taken in step 4 is simply the execution of the FFT algorithm (with the inverse option) which has the same cost as the forward FFT. Therefore, it is also $O(N \log N)$ keeping our total cost at $O(N \log N)$. By comparison, the computation required for the discrete convolution method is $O(N^2)$ due to the nature of the convolution operation.

4.5.1 The Cost of Smoothing

It should be noted, with all this discussion of computation cost of the method, that the choice of kernel function is very important if smoothing is to be performed. For example, the kernel we chose (Equation 4.7) for this example is very costly in terms of computation time. Ignoring the cost of computing the exponentials in the function, the evaluation of $p(x)$ using this kernel function requires evaluation at every sampled point for each frequency bucket. This feature alone makes the smoothing operation on the order of $O(N^2)$. On the other hand, the kernel used in the binomial example of Chapter 3 (see Equation 3.2) puts the smoothing operation on the order of $O(N)$.

From the results of the discrete unsmoothed convolution in Chapter 3, we can see that convolution is not even necessary if a less smooth resulting is satisfactory for the purposes of the analysis. If smoothing is desirable, but the performance burden is too great, a discrete kernel such as the one in Equation 3.2 can be used on the output of the convolution. The results of such a *late smoothing* will not be as profound as those of an early smoothing, but will still help to smooth the distribution a bit. Of course, if the goal of computing the output distribution is to compute a particular percentile of the distribution, such smoothing is not likely to be very useful.

Chapter 5

Convolution by Characteristic Function

In the previous chapter, we introduced a method of computing the convolution of several component distributions by means of the Discrete Fourier Transform (DFT). In this chapter, we present a method by which the convolution can be computed by means of the Continuous Fourier Transform (CFT) of the probability function of a continuous distribution. The CFT of the probability density function is known in probability theory as the Characteristic Function (CF). Accordingly, we will use the terms CFT and CF interchangeably throughout this chapter.

In order to use the method presented in this chapter, we must approximate the input distributions with a continuous distribution for which the CF is known. Unfortunately, the CF is not known in closed form for all distributions. A well known example of the lack of closed form CF is that of the Lognormal Distribution. The Lognormal is known (see [16] and [18]) to have a CF that is difficult to compute, due to the inability to expand the CF as a Taylor series based on the moments. The Taylor series can be shown to diverge and as such makes the CF difficult to compute. At

this time, the current body of research seems to imply a tendency towards believing a closed-form representation of the CF that is useful for computational purposes may not exist. At the very least, it has not yet been found.

However, for most distributions, the CF is known and is relatively easy to compute. For those distributions that have an easy to calculate CF, we will use the analytical CF. For those that have a CF that is difficult to compute, we will use the DFT of a discretized version of the PDF in place of the CF.

5.1 CFs of Common Distributions

In this paper, we have used three common continuous distributions: Normal, Gamma, and Lognormal. Of these three distributions, the first two have closed-form analytical CFs that can be easily computed. The third — Lognormal — does not. For the Normal distribution, the CF is given by

$$\varphi(t) = e^{i\mu t - \sigma^2 t^2/2} \tag{5.1}$$

where μ and σ are the well-known parameters of the distribution. For the Gamma distribution, given by probability density function

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta} \tag{5.2}$$

the CF is given by

$$\varphi(t) = \frac{1}{(1 - i\beta t)^\alpha}\,. \tag{5.3}$$

In Chapter 2, we discussed mixture distributions and their properties. Given a mixture distribution with the probability density function

$$f_Z(z) = \sum_{j=0}^{M-1} \pi_j f_{X_j}(z)\,, \tag{5.4}$$

which represents a mixture with M components having the density functions f_{X_j} and mixture proportion π_j, a little algebra will show that the CF of the mixture is given by

$$\varphi_Z(t) = \sum_{j=0}^{M-1} \pi_j \varphi_{X_j}(t) \tag{5.5}$$

where the φ_{X_j} are the CFs corresponding to the f_{X_j}.

5.2 The CFT Method

The characteristic function method described in this chapter is similar to the DFT method described in the previous chapter. The major difference between the methods is that the CF method requires fitting a continuous distribution to the data of the component distributions, and then uses these resulting distributions to compute the CF of the convolution. A pictoral description of the method is given in Figure 5.1.

Both the DFT method and the CF method eventually use the inverse FFT to compute the approximate probability function. However, the CF method assumes that a continuous distribution more accurately represents the underlying population from which the sampled times come. Once these continuous distributions are estimated, the CF method samples the CF of the convolution by computing the CFs of each of the components at the same set of points at which the iFFT is to be evaluated. Essentially, we are sampling in the frequency domain, rather than in the time domain, as is done in the DFT method.

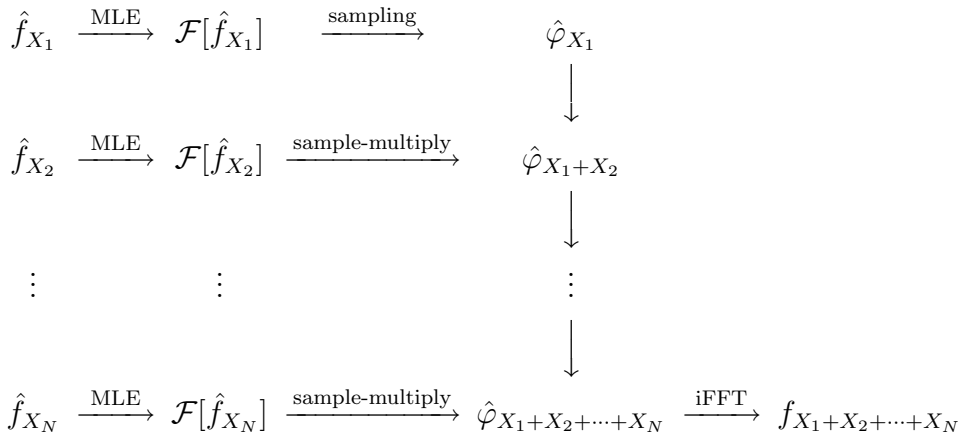

Figure 5.1: The CF Method

The basic steps of the method are

1. Select appropriate continuous distributions for each of the components and determine the MLE estimates of their associated parameters,

2. Sample the CF of each of these component distributions at the appropriate points for iFFT application,

3. Compute the product of the CFs at the calculated points,

4. Use the results of step 3 as input to the iFFT algorithm. Normalize the output. The resulting array is an approximation to the probability function of the convolution.

5.2.1 Step 1 — Estimate the Distributions

In Chapter 2, we described how to estimate the underlying component population distributions with continuous distributions. We tabulated our results in Section 2.4. Using these results, we can find that for the distributions of Parts A and B, we need the CF of the Lognormal and Normal distributions respectively. Since, as we mentioned in the beginning of this chapter, the Lognormal CF is not easy to calculate,

we will leave Part A out of our current consideration. The next chapter will show how to integrate the DFT and CF methods so that distributions for which the CF is unknown can also be used. For parts C and D, we need the Gamma and Lognormal CFs respectively. Again, we will defer the computation of part D to the next chapter. Since we are deferring parts A and D to the next chapter, we will compute the convolution of B and C in this chapter.

For part B, the CF is the CF of a mixture of two Normal (Gaussian) components, with mixture proportions and parameters given in Figure 2.15. The resulting CF is

$$\begin{aligned} \varphi_B(t) &= 0.89 \exp\left(48.94685ti - (8.38651t)^2/2\right) \\ &+ 0.11 \exp\left(1057.17273ti - (30.68959t)^2/2\right) \end{aligned} \tag{5.6}$$

where $i = \sqrt{-1}$. With a little algebra, it can be shown that the real and imaginary parts of φ_B are

$$\Re\varphi_B(t) = \pi_1 \cos(\mu_1 t) e^{-\frac{\sigma_1^2 t^2}{2}} + \pi_2 \cos(\mu_2 t) e^{-\frac{\sigma_2^2 t^2}{2}} \tag{5.7}$$

$$\Im\varphi_B(t) = \pi_1 \sin(\mu_1 t) e^{-\frac{\sigma_1^2 t^2}{2}} + \pi_2 \sin(\mu_2 t) e^{-\frac{\sigma_2^2 t^2}{2}} \tag{5.8}$$

where the π_j, μ_j, and σ_j are those from Figure 2.15. Using the equations above, we can calculate the entries in the *real* and *imaginary* arrays that are needed for input into the iFFT algorithm.

Similarly, we can compute the CF of part C's distribution (quite easily) as

$$\varphi_C(t) = \left(\frac{1}{1 - 3.47122it}\right)^{29.72568} \tag{5.9}$$

which has real and imaginary parts given by

$$\Re\varphi_C(t) = \left(1+\beta^2t^2\right)^{-\alpha/2}\cos\left(\alpha\arctan\beta t\right) \tag{5.10}$$

$$\Im\varphi_C(t) = \left(1+\beta^2t^2\right)^{-\alpha/2}\sin\left(\alpha\arctan\beta t\right) \tag{5.11}$$

5.2.2 Step 2 — Sample the CF

In this step, we need to sample the CFs determined in step 1 for each of the component distributions (in our current example, just parts B and C) at the appropriate points for evaluation of the iFFT. At this point, we need to discuss what such appropriate points might be. The FFT algorithm that we are using outputs two arrays, a *real* array and an *imaginary* array. These arrays are the real and imaginary parts of the DFT, respectively. Since we are skipping the application of the FFT and going straight to the CF, but wish to still use the iFFT, we must put our CF samples in the same form expected by the iFFT algorithm (which, of course, is the same form output by the FFT algorithm).

The astute reader may notice that we started with samples, so it might seem strange that we are again sampling. The reason we are sampling again is that the original sample data was used to estimate the parameters for the analytical distribution we use to represent the component, and the sampling we are doing here is to transition from the analytical CF to an array of values that can be used in the FFT algorithm.

The order of the elements in the arrays is, of course, significant to the iFFT algorithm. The FFT implementation we use for this paper puts the positive part of the DFT domain $(0, 1, \ldots, N/2-1)$ in the first $N/2$ entries of the array, and the negative part of the domain in the second $N/2$ entries. The negative part of the

domain is output in normal order, so the element at index $N/2$ of the output array (0-based) corresponds to the DFT at $-N/2$, and index $N/2+1$ to $-(N/2-1)$, and so on. It should also be noted that the true domain of the Fourier Transform is from $-\pi$ to $+\pi$ in the frequency domain, and that this means that an argument of k (in the DFT domain) corresponds to $k\frac{2\pi}{N}$ in the Fourier domain.

As mentioned in Chapter 1, the definition of the Fourier Transform also comes into play. Our definition of the Fourier Transform (Equation 1.6) does not correspond to the one used by our FFT algorithm (MS Excel). To be precise, the CFT as we defined it is evaluated at the opposite of the argument at which Excel evaluates. Therefore, we must sample at the points $t^* = -t$ when sampling our CF.

5.2.3 Step 3 — Multiply the Results

We are now armed with the facts necessary to build the CF sample that corresponds to the output of the FFT algorithm (and of course to the input of the iFFT). Since we wish to compute the convolution of part B's distribution with that of part C, we simply multiply the CFs of B and C at each point at which we evaluate them. Accordingly, we need to evaluate the CF

$$\varphi(t) = \varphi_B(t)\varphi_C(t) \tag{5.12}$$

at the points

$$t^* = -t = -\{[(j + N/2) \mod N] - N/2\}\frac{2\pi}{N} \tag{5.13}$$

for $j = 0, 1, \ldots, N-1$.

In order to do this calculation, we need to multiply $\varphi_B(t)$ with $\varphi_C(t)$. Let us compute one of the entries from our set of arrays as an example of the computation.

For example, for $j = 3$, we have

$$\begin{aligned} t &= \{[(3 + 4096/2) \mod 4096] - 4096/2\}\frac{2\pi}{4096} \\ &= 0.00460194 \end{aligned}$$

so that

$$\begin{aligned} \Re\varphi_B(-t) &= 0.88343247 \\ \Im\varphi_B(-t) &= -0.09099224 \\ \Re\varphi_C(-t) &= 0.88601410 \\ \Im\varphi_C(-t) &= -0.45543744 \end{aligned}$$

and

$$\begin{aligned} \Re\varphi(-t) &= \Re\varphi_B(-t)\Re\varphi_C(-t) - \Im\varphi_B(-t)\Im\varphi_C(-t) \\ &= 0.74129235 \\ \Im\varphi(-t) &= \Re\varphi_B(-t)\Im\varphi_C(-t) + \Im\varphi_B(-t)\Re\varphi_C(-t) \\ &= -0.48296863. \end{aligned}$$

5.2.4 Step 4 — Calculate the iFFT

The values of t^* given by Equation 5.13 range from $-\pi$ to π in equal steps. Having computed all of the entries in the array, we can compute the iFFT of this array to invert the CF of the convolution distribution.

Given the output of Step 3 (an array of numbers in FFT-output order), we can compute the iFFT of this array and obtain the estimate of the probability function

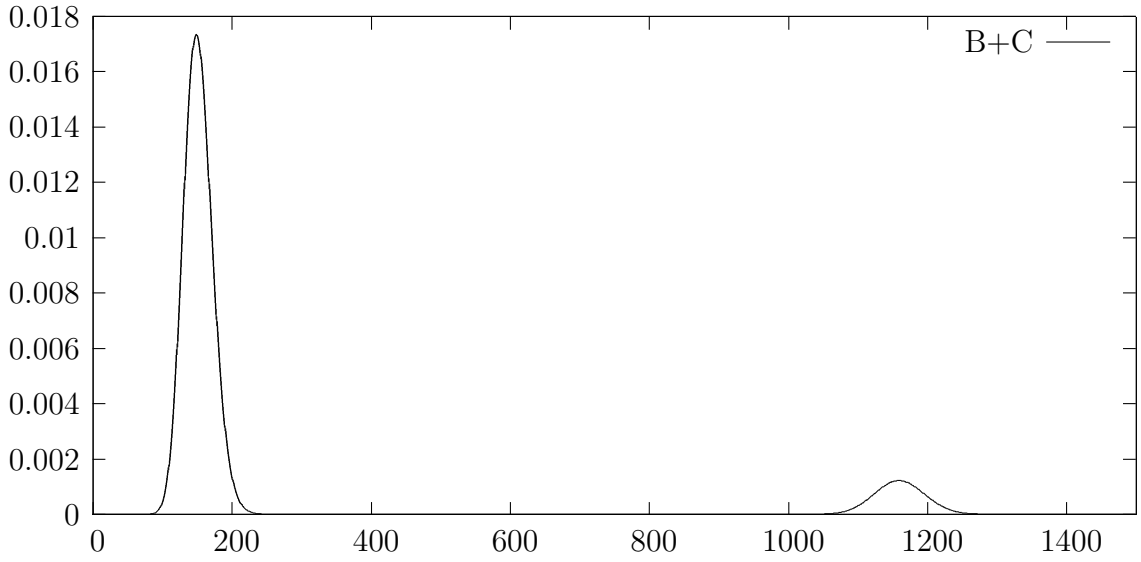

Figure 5.3: The Convolution PDF of B+C

for $B + C$. Some selected entries from the resulting probability function are given in Figure 5.2, and a plot of the results are given in Figure 5.3.

Table 5.2: Selected Entries from $\hat{f}_{B+C}$.

z	$\hat{f}_{B+C}(z)$	z	$\hat{f}_{B+C}(z)$	z	$\hat{f}_{B+C}(z)$
100	0.00041065	101	0.00048661	102	0.00057386
148	0.01731210	149	0.01734352	150	0.01733304
151	0.01728132	210	0.00054769	1148	0.00115884
1153	0.00119979	1160	0.00121896	1165	0.00120454
1172	0.00114709	1178	0.00106790	1182	0.00100294

5.3 Computational Cost of the Method

The computational cost of computing the CFT is $O(N)$ for each component distribution, once the component distribution is approximated with a continuous distribution. Even in the case of the Lognormal distribution, the order cannot be $O(N^2)$ to compute the characteristic function because N is not related in any way to φ once the fit is performed (the only parameters to φ are t, σ, and μ). Since again we assume there is a fixed number of convolutions to be performed, the total convolution process is $O(N)$. However, since we need to compute the iFFT to return to the "time" domain (the x-domain), we need one iFFT computation at cost $O(N \log N)$. This brings our total cost of computation to $O(N \log N)$. While this total cost is the same as the DFT method, we believe that when the DFT method is used on continuous fitted distributions (as this method is), the actual costs will be less for this method, assuming the CFT is not very difficult to compute. The difference in the cost of computation between the two methods is based on the difference between the cost of computing the CFT for the values of t needed versus the cost of sampling and computing the DFT at the sampled points. In the case of using the DFT method with raw sampled data (without MLE fitting the component distributions and sampling from the probability functions) or smoothed data with a very simple smoothing kernel, the DFT method will probably be faster, though it may be less accurate (depending on the sample size).

Chapter 6

Combining the DFT and CFT Methods

In the previous two chapters, we described methods for computing an approximation to the convolved distribution by use of the DFT and CFT respectively. In Chapter 5, we could not apply the CFT method to Part A or Part D due to the fact that the Lognormal distribution does not have an easy to compute closed form CF (φ). In this chapter, we show how to use the DFT method to compute the approximation to the CF of these two parts, while using the CFT method for the other two parts. The total cost of this method will also be $O(N \log N)$ since each of the two parts are of this order.

6.1 The Combined Method

Essentially, this method uses the DFT method from Chapter 4 to compute the convolution of Parts A and D, while using the CFT method from Chapter 5 to compute the

convolution of Parts B and C. The application of the DFT method to this problem is slightly different in this method, however.

6.1.1 Preparing the Distributions

In order to apply the DFT method in this problem, we would like to prepare our distributions of Parts A and D to match, as closely as possible, the results that we would get from applying the CFT method to these distributions. Instead of applying smoothing to our samples or using the DFT method directly on the raw sampled data, we wish to prepare input arrays with the probability functions of interest. In this case, we wish to prepare input arrays for Part A that closely match the probability function of the Lognormal mixture that we fit to Part A's data. In the case of Part D, we wish to similarly prepare input arrays that closely match the probability function of the Lognormal distribution that we fit to the data.

In order to prepare the arrays, we sample the probability function of the applicable distributions at the appropriate points in the array. Since both of these distributions have the positive real line as their domain, we must cut them off at a certain point to avoid aliasing problems in the FFT. We will discuss the procedure for aliasing avoidance later in this chapter.

6.1.2 Computing the Convolution Distribution

After sampling from the probability function, we will apply the DFT method as in Chapter 4, obtaining the results of the convolutions of the distributions that are difficult for the CFT method. We will not apply the final step of the DFT method, however, so we will leave the results in the Fourier domain (the products of the FFTs).

For the distributions to which we can apply the CFT method, we apply the method. Again, as in the application of the DFT method to the other distributions, we will not apply the final step. Thus, after applying the CFT method, minus the final step, we will have the product of the approximated CFTs of the probability functions in the Fourier domain. We may now complete our computations by computing the product of the results from the DFT method and those from the CFT method. This product will be the approximated CFT of the total convolution, to which we may now apply the iFFT once (the final step of both methods). The result of the iFFT of this product will be the approximation of the probability function of the convolution.

6.2 An Example

We will use the CFT results obtained in Chapter 5 without adjustment, as they are precisely the same results that would be obtained by the combined method for Parts B and C. We take the results just prior to applying the final iFFT in the CFT method, as prescribed in Section 6.1. For the DFT results, we must compute the sampled probability functions and then perform the DFT method on them, stopping prior to the final step.

The probability function for Part A is the mixture probability function

$$f_Z(z) = \frac{\pi_1}{z\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log z - \mu_1}{\sigma_1})^2} + \frac{\pi_2}{z\sigma_2\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log z - \mu_2}{\sigma_2})^2} \tag{6.1}$$

where the π_i, μ_i, and σ_i are those given in Figure 2.15. Whereas the probability function for Part B is the lognormal probability function given by

$$f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log x - \mu}{\sigma})^2} \tag{6.2}$$

where μ and σ are those given in Figure 2.16.

We need to compute these functions at the same input values as our other distributions, but we need to clamp the distributions values to a range that will not allow overflow. For instance, since we chose an output array size of 4096 values (range of $[0, 4095]$) we could restrict each of our distributions (rather arbitrarily) to a range of $[0, 1023]$. However, this would not be the best choice, since two of our distributions have significant probability outside of this range and the others don't. A more appropriate choice might be to limit the two mixture distributions (which have significant probability at values above 1000) to the range $[0, 1499]$ and the other two distributions to the range $[0, 499]$. This way, the largest values from each of the distributions cannot produce a value with positive probability above 4095. Sometimes, depending on the tail weight of the distributions in question, such clamping is not necessary. This happens to be the case in our situation, since the probability of the total resulting distribution exceeding 4095 is negligible.

Once we have evaluated the probability function at the points in our array, we must normalize the array (so it sums to unity) by dividing each entry by the sum of all entries in the array. This, in effect, creates a discrete frequency distribution that is *similar* to the original continuous distribution that can be used as input to the FFT algorithm. We then apply the FFT algorithm to each of the two newly discretized distributions to obtain approximations to the CFTs. According to the DFT method and the adjustments prescribed in this chapter, we should also take the complex product of the two sets of output arrays prior to using the results in the combined method, but this is not important. We could just as easily treat these output arrays as sampled versions of the CFTs of the two distributions, and use these CFTs in the the multiplication step of the CFT method directly. In either case, once we take the product of these results with the product of the CFTs given by the CFT method, we can then apply the iFFT algorithm to obtain the probability function of

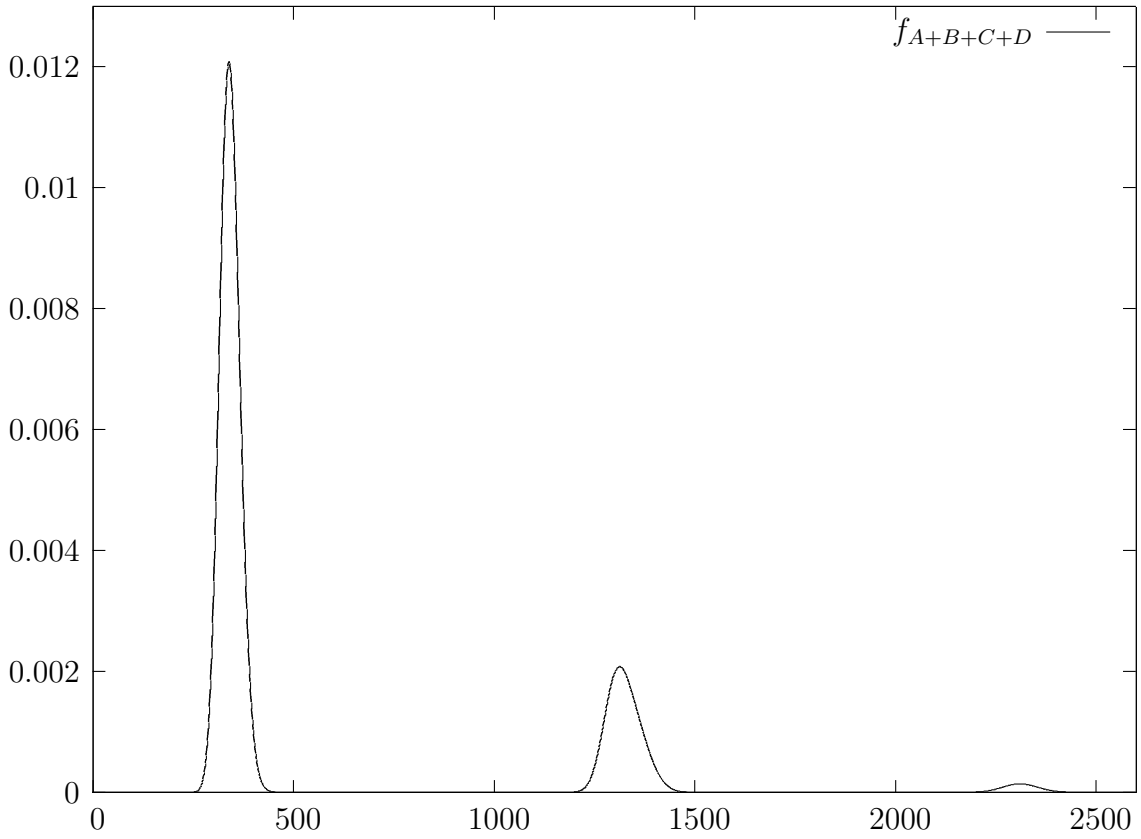

Figure 6.1: The Convolution PDF of all parts

the convolution. The results of applying this method for the example program are shown in Figure 6.1.

Chapter 7

Testing the Models

We have described how to compute estimates of the distributions of the total execution time using several methods. Now, we wish to test the resulting distributions by using them to predict the WCET of the program for a set of simulated values (from the same distributions used to generate our testing data). We will compare the average *excess* time for our prediction and the number of times our deadline is missed to the excess time for the Gumbel estimate. We will also use our estimated distribution to compute a revised Gumbel estimate that makes use of the more detailed information provided by our methods.

7.1 The Test Distributions

The distributions used to generate the training data provided in Appendix B are listed in Figure 7.1. From these test distributions, we simulate a test data set that is used to evaluate the WCET bounds determined several different ways. We simulate 100000 sample values from the total run-time distribution by simulating 100000 ordered 4-tuples with an element from each of the four component distributions. For

the two mixture distributions, the element is determined by simulating a uniform random variate (on the unit interval) and one deviate from each of the two mixture component distributions. We choose the deviate from the first mixture component if the uniform deviate is less than π_1, the second otherwise. Such a method accurately models the mixture distribution as a random (nonuniform) sample from two separate distributions.

Table 7.1: Underlying Test Distributions

Part	**Distribution**	**Parameters**	
A	Mixture (Normal)	$\pi_1 = 0.8$	$\pi_2 = 0.2$
		$\mu_1 = 40$	$\mu_2 = 1000$
		$\sigma_1 = 2$	$\sigma_2 = 20$
B	Mixture (Gamma)	$\pi_1 = 0.85$	$\pi_2 = 0.15$
		$\alpha_1 = 25$	$\alpha_2 = 350$
		$\beta_1 = 2$	$\beta_2 = 3$
C	Gamma	$\alpha = 25$	$\beta = 4$
D	Lognormal	$\mu = 5$	$\sigma = 0.1$

Due to space constraints, the simulated values are not tabulated in this paper, but are available on request from the author.

7.2 WCET Estimates

We use several WCET estimates and compare their performance in terms of the number of times the WCET bound is exceeded by our samples, the average value of

observations that exceed the bound, and the average excess of the bound over the total run-time. We contest that the WCET bound given by [7] is very high and leads to wasted cycles when used for scheduling. However, the percentile method we wished to use for our estimate of the WCET leads to bounds too low for practical use, as a substantial number of observations exceed this bound. We instead recommend a smaller-sample extreme value (EV) estimate similar in character to the Gumbel estimate, except using the exact (approximated) EV distribution of a finite sample size. The results of these various estimates of the WCET are compared in Figure 7.2. The estimation methods for the WCET tabulated in Figure 7.2 are

Gumbel (sample). The Gumbel distribution WCET estimate (as described in [7] with corrections[1]) based on the original sample of total run-times,

Gumbel (distribution). The same as *Gumbel (sample)*, except using estimates of μ and σ from the estimated total run-time distribution given by our method,

Percentiles. The percentiles of the total run-time distribution as estimated by our method,

Linear Combination. The WCET estimate given by a linear combination of the the *Percentile* estimate and the *Gumbel (distribution)* estimate using the coefficients 0.75 and 0.25 respectively,

[1]Some of the formulas given in [7] have errors in them. The most important of which is Equation 3, which should be

$$\theta_i(x) = \frac{G(x) - G(max_i)}{1 - G(max_i)}.$$

A more precise description of the problems of [7] would argue that their Theorem 1 is not valid. In fact, the WCET estimate should be $max_i + \omega_i$ if ω_i is the value of their θ_i that gives $\theta_i(\omega_i) = 1 - \epsilon_i$.

Gumbel (max locmax). The Gumbel WCET estimated found by using a μ and σ estimated from only the portion of the distribution where the largest (in x) local maximum is found in the density,

$\mathbf{F^k(x)}$. The WCET estimate given by the specified percentile of the excess value distribution (see Section 7.3) for sample size k.

7.3 The Exact EV Distribution

The Gumbel distribution used in [7] is the asymptotic distribution of the maximum value from a distribution. It represents the limit as the sample size tends to infinity of the maximal outcome in a random sample from any exponential family distribution (nearly all distributions of interest for our purposes are from this family). In simple terms, it is the probability distribution for the largest value that one would ever expect to see, given a few parameters about an existing sample. Because this distribution is asymptotic, it is very conservative in the bounds it provides. As can be seen from the data in Figure 7.2 for the *Gumbel (sample)* versus the $F^k(x)$ estimates, the average amount by which the WCET from the Gumbel method exceeds the actual run-time is much greater than that from the exact method.

The exact method is based on the same idea as the Gumbel distribution, but makes use of the actual distribution of the individual observations, rather than just the sample and a few sample statistics. Because of this, the real EV distribution based on our estimate of the distribution function $F(x)$ of the total run-time distribution provides a much tighter bound on the WCET. In general, the distribution function (probability that the variate does not exceed a given value) $F_{x_{(k)}}(x)$ of the maximum

value $x_{(k)}$ of a sample of size k is given by

$$F_{x_{(k)}}(x) = (F_X(x))^k \tag{7.1}$$

where $F_X(x)$ is the distribution function underlying the sample observations themselves.

The EV method used here raises the estimated total run-time distribution function $F(x)$ (of the convolution) to various powers of k and locates the p^{th} percentile of this distribution. The percentile is then used as the WCET estimate, providing a much smaller bound than the Gumbel distribution. The method allows adjustment both by changing k and by changing the desired percentile. Also, the underlying accuracy of $F(x)$ can be controlled by changing the number of points at which the CFs are sampled (or at which the FFT is applied in the discrete case).

Table 7.2: WCET Estimates and Results

Estimate Type	Confidence	WCET Bound	Count Over	Average Over	Average Excess
Gumbel (sample)	90%	3193	–	–	2607.37
	95%	3448	–	–	2862.37
	97.5%	3703	–	–	3117.37
	99%	4039	–	–	3453.37
Gumbel (distribution)	90%	3172	–	–	2586.37
	95%	3420	–	–	2834.37
	97.5%	3668	–	–	3082.37
	99%	3994	–	–	3408.37
Percentiles	90%	1333	9769	1511.84	764.84
	95%	1366	4971	1669.77	795.47
	97.5%	1400	2423	1973.97	828.28
	99%	2294	974	2334.02	1708.76
Linear Combination	90%	1793	1511	2309.68	1215.18
	95%	1880	1511	2309.68	1300.87
	97.5%	1967	1511	2309.68	1386.55
	99%	2719	–	–	2133.37
Gumbel (max locmax)	90%	2427	9	2435.47	1841.37
	95%	2451	1	2453.64	1865.37
	97.5%	2475	–	–	1889.37
	99%	2506	–	–	1920.37
$F^k(x)$ $k = 1,000$	90%	2421	11	2433.65	1835.37
	95%	2432	7	2437.48	1846.37
	97.5%	2443	1	2453.64	1857.37
	99%	2455	–	–	1869.37
$F^k(x)$ $k = 10,000$	90%	2455	–	–	1869.37
	95%	2464	–	–	1878.37
	97.5%	2473	–	–	1887.37
	99%	2484	–	–	1898.37
$F^k(x)$ $k = 100,000$	90%	2483	–	–	1897.37
	95%	2492	–	–	1906.37
	97.5%	2499	–	–	1913.37
	99%	2509	–	–	1923.37
$F^k(x)$ $k = 1,000,000$	90%	2509	–	–	1923.37
	95%	2517	–	–	1931.37
	97.5%	2524	–	–	1938.37
	99%	2535	–	–	1949.37

Chapter 8

Conclusion

We have shown that the use of our DFT and CFT methods are an efficient alternative to the traditional methods of computing the convolutions that arise in the estimation of run-time distributions built of several components. Additionally, we have shown that a tighter bound on the WCET can be provided by assuming a less-than-infinite sample size and using distributional estimates of the run-time distribution, along with the theoretical fixed-sample distribution of the maximum observed value (the $F^k(t)$ distribution function).

8.1 Future Work

In the development of WCET estimates, it would be desirable to have estimates of, or bounds on, the error of this estimate. Of course, an exact error estimate or bound requires an assumption for the number of times the task is to be run. Alternatively, if an asymptotic estimate is made, the number of times the task is to be run is assumed to be infinite. This assumption of the number of runs should be consistent with the underlying assumption of the WCET estimate.

In the estimate of the WCET error, it would also be desirable to understand the relationship between the input parameters relating to sampling, bucketization, and distribution choice and the underlying error of the estimate. Also, since there is obviously a relationship of some sort between these items and the error of the estimate, it would be nice to have a method of optimizing the choice of these parameters in order to minimize the resulting error.

In our work, we have assumed that a breakdown of the task into independently distributed components has been chosen appropriately. However, we have made no effort to develop methods for choosing this breakdown. It would be beneficial to have some theoretical support and some well defined algorithms or heuristics for the choice of this breakdown and the evaluation of the independence of the elements.

8.2 Some Observations

During the course of our research, we have made some observations for which we have no theoretical support, but nonetheless would like to present here. First of all, we have noticed that the separation of the task into several independent components has a profound effect on the sampling burden needed for confidence in the resulting distribution. We call this effect the independent sampling effect (ISE). The ISE can be understood through a simple example. Consider a task that can be broken up into three independent components. If we sample each of these components 10 times, this is equivalent (under the assumption of independence) to sampling the total distribution 1000 times.

The obvious advantage of the ISE is the number of samples required for confidence when using independent components is much smaller than the number of samples required for the same level of confidence in an estimate of only the total

run-time distribution. This greatly reduces the sampling burden and allows us to make relatively accurate estimates of the total run-time distribution with relatively few sampling runs of the program. Using the example of the last paragraph, we could run the program ten times, sampling it as three separate components (a total of 30 samples), and have the same level of confidence in the resulting estimate as we would have in an estimate using only the total run-time and a sample size (number of program executions) of 1000 runs. Of course, if it is expensive to run the program, the ISE is of substantial benefit to us.

In addition to the obvious advantage of the ISE, another advantage is the exhaustion effect of the ISE. For instance, in the case of the three components, 10 samples each as given in previous paragraphs, consider that two of the three components had distributions that are mixtures with a high and a low distribution. Consider also that the high elements of the mixtures are much less likely to occur than the low elements, for instance an 80% likelihood of the low element and a 20% likelihood of the high. Consider, as would be expected, that two of the observations from the 10 samples of each of these two components arose from the high distribution, but that none of the total run-time samples (runs) exhibited high elements from both of the two mixture distributions. In this case, the Gumbel method would use a maximum as a parameter of its estimate that is actually not anywhere close to the maximum of the theoretical distribution, since none of the runs exhibited the "high-high" behavior. However, when the components are convolved and the ISE occurs, an "artificial" maximum arises from the pairs of high samples that were encountered. Even though they occurred on different runs, the assumption of independence and the ISE allowed them to be used in the analysis together. In this small-sample case where there are gaping holes in the sample distribution, the ISE provides a great benefit and leads to a much greater accuracy in our estimate of WCET.

One final conclusion related to the ISE is that increasing the number of components into which the total distribution is broken gives much greater benefit than increasing the number of runs of the program used for sampling. This is a very nice result, assuming good methods are available for choosing appropriate breakdowns into independent components. One caveat, however, is that there is always a risk of reducing the sample size (number of sampling runs) to a point that features of the component distributions are missed. For instance, in the example of this chapter, if the sample size is reduced by too much, the likelihood of seeing any "high" elements in the components becomes relatively small. Also, if the "high" elements are much less likely (for instance 1% of the time, rather than 20% as in our example), the sampling burden is much higher on the component level and in turn the number of runs required will be much higher. That being said, the number of runs for the same level of confidence at the total run-time level only would be much higher, again due to the ISE.

Appendix A

Maximum Likelihood Estimation

In this appendix, we derive formulas and methods for the MLEs of the three distributions we use in this thesis. In all sections of this appendix, we assume the following:

1. a set of observations $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$,
2. a random variable X representing the distribution from which the observations are sampled, and
3. a vector of parameters $\boldsymbol{\Psi}$ for MLE fitting.

A.1 Normal Distribution

The MLE vector for the Normal distribution is $\boldsymbol{\Psi} = (\mu, \sigma)$. We wish to compute the estimate of this vector, $\hat{\boldsymbol{\Psi}} = (\hat{\mu}, \hat{\sigma})$. The likelihood function $L(\boldsymbol{\Psi})$ is defined by

$$L(\boldsymbol{\Psi}; \boldsymbol{x}) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}. \tag{A.1}$$

The log-likelihood function $\ell(\boldsymbol{\Psi}; \boldsymbol{x})$ is given by

$$-\ell(\boldsymbol{\Psi}; \boldsymbol{x}) = N \log \sigma\sqrt{2\pi} + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2, \tag{A.2}$$

with partial derivatives (with respect to the parameters) given by

$$\frac{\partial \ell(\boldsymbol{\Psi})}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{N} (x_i - \mu) \tag{A.3}$$

and

$$-\frac{\partial \ell(\boldsymbol{\Psi})}{\partial \sigma} = \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^{N} (x_i - \mu)^2, \tag{A.4}$$

leading to the MLE estimates

$$\hat{\mu} = \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{A.5}$$

and

$$\hat{\sigma}^2 = S^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2. \tag{A.6}$$

A.2 Log-Normal Distribution

The MLE vector for the Log-Normal distribution is $\boldsymbol{\Psi} = (\mu, \sigma)$. We wish to compute the estimate of this vector, $\hat{\boldsymbol{\Psi}} = (\hat{\mu}, \hat{\sigma})$. The likelihood function $L(\boldsymbol{\Psi})$ is defined by

$$L(\boldsymbol{\Psi}; \boldsymbol{x}) = \prod_{i=1}^{N} \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log x - \mu}{\sigma})^2}. \tag{A.7}$$

The log-likelihood function $\ell(\boldsymbol{\Psi}; \boldsymbol{x})$ is given by

$$-\ell(\boldsymbol{\Psi}; \boldsymbol{x}) = N \log(\sigma\sqrt{2\pi}) + \sum_{i=1}^{N} \log x_i + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (\log x_i - \mu)^2, \tag{A.8}$$

with partial derivatives (with respect to the parameters) given by

$$\frac{\partial \ell(\mathbf{\Psi})}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{N} (\log x_i - \mu) \tag{A.9}$$

and

$$-\frac{\partial \ell(\mathbf{\Psi})}{\partial \sigma} = \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^{N} (\log x_i - \mu)^2, \tag{A.10}$$

leading to the MLE estimates

$$\hat{\mu} = \bar{z} = \frac{1}{N} \sum_{i=1}^{N} z_i \tag{A.11}$$

and

$$\hat{\sigma}^2 = S^2_{\log x} = \frac{1}{N} \sum_{i=1}^{N} (z_i - \hat{\mu})^2, \tag{A.12}$$

where

$$z_i = \log x_i.$$

A.3 Gamma Distribution

In this section, we provide a method for computing the maximum likelihood estimators of α and β, the parameters of the two-parameter Gamma distribution specified by the probability density function

$$f(t; \alpha, \beta) = \frac{t^{\alpha-1} e^{-t/\beta}}{\Gamma(\alpha) \beta^{\alpha}} \ .$$

We will compute on the observation vector $\boldsymbol{x}$ three statistics $\bar{x}$, s^2, and $\overline{l(x)}$. We will define these statistics as

$$\bar{x} = \frac{1}{N}\sum_i x_i \ , \tag{A.13}$$

$$s^2 = \frac{1}{N-1}\sum_i (x_i - \bar{x})^2 \ , \tag{A.14}$$

$$\overline{l(x)} = \frac{1}{N}\sum_i \log x_i \ . \tag{A.15}$$

Our goal is to maximize the likelihood function $L(\boldsymbol{\Psi})$ for the parameters $\boldsymbol{\Psi} = \{\alpha, \beta\}$. The likelihood function in this case is

$$L(\boldsymbol{\Psi}) = \prod_i \frac{x_i^{\alpha-1} e^{-x_i/\beta}}{\Gamma(\alpha)\beta^\alpha} = \Big(\Gamma(\alpha)\beta^\alpha\Big)^{-N} \cdot \Big(\prod_i x_i\Big)^{\alpha-1} \cdot \exp\Big(-\frac{1}{\beta}\sum_i x_i\Big) \ .$$

As it is generally simpler to maximize the log-likelihood function, we calculate the log-likelihood function

$$\begin{aligned} \ell(\boldsymbol{\Psi}) = \log L(\boldsymbol{\Psi}) &= -N\big[\log\Gamma(\alpha) + \alpha\log\beta\big] + (\alpha-1)\sum_i \log x_i - \frac{1}{\beta}\sum_i x_i \\ &= -N\big[\log\Gamma(\alpha) + \alpha\log\beta\big] + N\,(\alpha-1)\,\overline{l(x)} - \frac{N\bar{x}}{\beta} \ . \end{aligned}$$

In order to make the optimization simpler, we can instead minimize the negative log-likelihood (NLL) divided by N (since N is a constant). We are then interested in minimizing the function

$$Z = \log\Gamma(\alpha) + \alpha\log\beta - (\alpha-1)\overline{l(x)} + \frac{\bar{x}}{\beta}. \tag{A.16}$$

The vector $\mathbf{\Psi}$ will be optimal if

$$F(\mathbf{\Psi}) = \begin{pmatrix} Z_\alpha(\mathbf{\Psi}) \\ Z_\beta(\mathbf{\Psi}) \end{pmatrix} = \mathbf{0}, \tag{A.17}$$

where the notation $Z_x(\mathbf{\Psi})$ denotes the partial derivative of Z with respect to the variable x evaluated at $\mathbf{\Psi}$.

The partial derivatives of Z are given by

$$Z_\alpha(\mathbf{\Psi}) = \psi(\alpha) + \log\beta - \overline{l(x)} \tag{A.18}$$

and

$$Z_\beta(\mathbf{\Psi}) = \frac{1}{\beta^2}\left(\alpha\beta - \bar{x}\right), \tag{A.19}$$

where $\psi(\alpha)$ is the derivative of $\log\Gamma(\alpha)$.

However, since we are solving the system of equations $F = \mathbf{0}$, we can multiply Z_β by β^2 to eliminate the distributed fraction (β is defined to be nonzero by the Gamma distribution).

Using these changes, we are interested in solving the system

$$\tilde{F}(\mathbf{\Psi}) = \begin{pmatrix} \psi(\alpha) + \log\beta - \overline{l(x)} \\ \alpha\beta - \bar{x} \end{pmatrix} = \mathbf{0}. \tag{A.20}$$

Newton's method for systems states ([4]) that the system can be solved by starting with an initial guess $\mathbf{\Psi}^{(0)}$, and iterating using the equation

$$\mathbf{\Psi}^{(k+1)} = \mathbf{\Psi}^{(k)} + \mathbf{\Delta}^{(k)} \tag{A.21}$$

where

$$\boldsymbol{\Delta}^{(k)} = -\boldsymbol{J}^{-1}(\boldsymbol{\Psi}^{(k)})\tilde{F}(\boldsymbol{\Psi}^{(k)}) \tag{A.22}$$

with $\boldsymbol{J}$ being the Jacobian of the system ($\tilde{F}$).

Let us write $\tilde{F}_1$ and $\tilde{F}_2$ for the first and second elements of $\tilde{F}$ as defined by Equation A.20. Given this notation, and using the partial derivative notation as earlier, we can define the Jacobian of the system as

$$\boldsymbol{J}(\boldsymbol{\Psi}) = \begin{pmatrix} (\tilde{F}_1)_\alpha & (\tilde{F}_1)_\beta \\ (\tilde{F}_2)_\alpha & (\tilde{F}_2)_\beta \end{pmatrix} = \begin{pmatrix} \psi^{(1)}(\alpha) & \frac{1}{\beta} \\ \beta & \alpha \end{pmatrix} \tag{A.23}$$

with $\psi^{(1)}(\alpha)$ the first derivative of $\psi(\alpha)$. The Jacobian has the inverse $\boldsymbol{J}^{-1}(\boldsymbol{\Psi})$ given by

$$\boldsymbol{J}^{-1}(\boldsymbol{\Psi}) = \frac{1}{\alpha\psi^{(1)}(\alpha) - 1} \begin{pmatrix} \alpha & -\frac{1}{\beta} \\ -\beta & \psi^{(1)}(\alpha) \end{pmatrix}. \tag{A.24}$$

Writing T_1, T_2, and T_3 as

$$T_1 = \psi(\alpha) + \log\beta - \overline{l(x)} \tag{A.25}$$

$$T_2 = \alpha\beta - \bar{x} \tag{A.26}$$

$$T_3 = \alpha\psi^{(1)}(\alpha) - 1, \tag{A.27}$$

we can write

$$\boldsymbol{\Delta} = -\boldsymbol{J}^{-1}\tilde{F} = \frac{1}{T_3} \begin{pmatrix} -\alpha T_1 + T_2/\beta \\ \beta T_1 - \psi^{(1)}(\alpha) T_2 \end{pmatrix} \tag{A.28}$$

so that our update equations for $\alpha^{(k+1)}$ and $\beta^{(k+1)}$ are given by

$$\alpha^{(k+1)} = \alpha^{(k)} + \frac{1}{T_3^{(k)}}\left(-\alpha^{(k)}T_1^{(k)} + T_2^{(k)}/\beta^{(k)}\right) \quad \text{(A.29)}$$

$$\beta^{(k+1)} = \beta^{(k)} + \frac{1}{T_3^{(k)}}\left(\beta^{(k)}T_1^{(k)} - \psi^{(1)}(\alpha^{(k)})T_2^{(k)}\right) \quad \text{(A.30)}$$

where the T_j are given by

$$T_1^{(k)} = \psi(\alpha^{(k)}) + \log\beta^{(k)} - \overline{l(x)} \quad \text{(A.31)}$$

$$T_2^{(k)} = \alpha^{(k)}\beta^{(k)} - \bar{x} \quad \text{(A.32)}$$

$$T_3^{(k)} = \alpha^{(k)}\psi^{(1)}(\alpha^{(k)}) - 1. \quad \text{(A.33)}$$

A.3.1 The Process

Armed with the results of this section, we can numerically compute the MLE of the parameters (α and β) by the following procedure.

1. Let $\beta^{(0)} = \frac{s^2}{\bar{x}}$
2. Let $\alpha^{(0)} = \frac{\beta^{(0)}}{\bar{x}}$
3. Let $k = 0$
4. Compute $T_1^{(k)}$, $T_2^{(k)}$, and $T_3^{(k)}$ using Equations A.31, A.32 and A.33
5. Compute $\alpha^{(k+1)}$ and $\beta^{(k+1)}$ using Equations A.29 and A.30.
6. Compute $\delta^{(k+1)} = \left(\left(\alpha^{(k+1)} - \alpha^{(k)}\right)^2 + \left(\beta^{(k+1)} - \beta^{(k)}\right)^2\right)^{1/2}$
7. Increment k
8. If $\delta^{(k)} > \epsilon$ then goto step 4, otherwise – output $\alpha^{(k)}$ and $\beta^{(k)}$ as answer.

A.4 Adjustments for the EM Algorithm

In order to compute the MLE estimates of the parameters for a mixture distribution, we use the EM algorithm. The EM algorithm puts the computation of the MLE estimates for the distribution into a "missing data" context. The missing data are indicators identifying which of the mixture components gave rise to each data point. During each iteration of the EM algorithm, we are compute an estimate of the z_{ij} (the indicator variables). These estimates, denoted $\hat{z}_{ij}$ are then used along with the observations (the x_i) to estimate both the component proportions and the parameters of the component distributions. In this section, we present the adjustments that must be made in light of this new information, to allow the computation of the MLE estimates for the parameters of the mixture.

A.4.1 The Mixture Proportions (π_j)

The mixture proportions can be directly estimated by computing the ratio of the number of observations that were expected to have come from component j to the total number of observations. This estimate is given by

$$\hat{\pi}_j = \frac{1}{N} \sum_{i=1}^{N} \hat{z}_{ij}. \tag{A.34}$$

A.4.2 The Component Parameters

The component parameters must be estimated differently for each component distribution. In order to estimate the component parameters, we must adjust the statistics used by the MLE estimators given by Equations A.5, A.6, A.11, A.12, A.13, A.14 and A.15. These equations must be replaced by the similarly labeled equations

$$\bar{x} = \frac{\sum_{i=1}^{N} \hat{z}_{ij} x_i}{\sum_{i=1}^{N} \hat{z}_{ij}} \qquad \text{(A.5*, A.13*)}$$

and

$$S^2 = \frac{\sum_{i=1}^{N} \hat{z}_{ij} (x_i - \bar{x})^2}{\sum_{i=1}^{N} \hat{z}_{ij}} \qquad \text{(A.6*, A.14*)}$$

for the Normal and Gamma distributions, and

$$\bar{z} = \overline{l(x)} = \frac{\sum_{i=1}^{N} \hat{z}_{ij} \log x_i}{\sum_{i=1}^{N} \hat{z}_{ij}} \qquad \text{(A.11*, A.15*)}$$

for the Log-Normal and Gamma distributions, and

$$S^2_{\log X} = \frac{\sum_{i=1}^{N} \hat{z}_{ij} (\log x_i - \bar{z})^2}{\sum_{i=1}^{N} \hat{z}_{ij}} \qquad \text{(A.12*)}$$

for the Log-Normal distribution.

Appendix B

Training Data and Tabular Results

Table B.1: Example timing data for the example program

Obs.	A	B	C	D	Total
1	981.87	44.57	123.48	144.99	1294.91
2	42.22	55.27	81.29	154.91	333.69
3	40.58	49.82	96.25	155.65	342.3
4	37.63	1031.44	109.84	160.73	1339.64
5	45.34	44.69	79.14	147.14	316.31
6	38.38	48.78	77.55	155.62	320.33
7	1040.32	49.86	102.44	160.86	1353.48
8	999.95	38.42	84.23	136.52	1259.12
9	38.88	42.87	98.95	144.89	325.59
10	36.72	50.18	106.65	141.68	335.23
11	41.14	1078.53	105.73	145.86	1371.26
12	39.51	48.66	106.82	135	329.99
13	38.46	49.6	74.72	143.86	306.64
14	41.86	46.43	124.39	162.19	374.87
15	39.6	64.17	123.74	153.74	381.25
16	43.56	38.98	83.27	128.37	294.18
17	34.62	59.57	111.3	145.36	350.85
18	37.58	56.12	93.11	155.16	341.97
19	40.52	43.22	68.95	146.22	298.91
20	971.61	61.83	95.89	138.51	1267.84
21	40.41	52.37	117.12	141.33	351.23
22	40.95	65.61	81.78	134.89	323.23
23	994.83	34.28	118.03	136.3	1283.44
24	40.83	46.81	120.58	156.48	364.7
25	38.87	66.6	113.64	144.59	363.7
26	38.32	38.85	111.28	163.42	351.87

Table B.1: Example timing data for the example program

Obs.	A	B	C	D	Total
27	37.67	51.44	131.75	171.11	391.97
28	40.11	46.52	117.14	151.18	354.95
29	41.81	49.66	119.37	143.77	354.61
30	43.54	48.26	94.96	169.67	356.43
31	36.92	1049.41	97.22	131.35	1314.9
32	39.01	39.15	90.15	132.28	300.59
33	1007.41	41.17	120.35	165.2	1334.13
34	40.11	47.89	98.6	165.85	352.45
35	981.36	1074.21	138.83	150.78	2345.18
36	42.65	39.19	126.49	123.22	331.55
37	36.81	41.92	89.25	171.79	339.77
38	987.98	58.37	94.22	118.63	1259.2
39	39.45	39.37	100.23	191.85	370.9
40	39.37	49.42	99.82	157.02	345.63
41	39.92	45.98	95.07	127.77	308.74
42	1010.35	55.95	101.87	183.1	1351.27
43	42.58	57.11	118.75	148.33	366.77
44	39.05	58.32	101.4	147.24	346.01
45	36.18	41.16	102.13	158.75	338.22
46	34.21	49.13	177.42	174.79	435.55
47	39.29	55.52	94.46	143.45	332.72
48	39.07	51.26	63.79	132.53	286.65
49	42.24	42.94	82.96	156.57	324.71
50	40.41	69.36	67.53	150.92	328.22
51	39.39	46	115.22	150.76	351.37
52	989.21	62.72	96.97	153.47	1302.37
53	42.21	45.33	108.4	162.4	358.34
54	37.92	38.69	91.42	141.98	310.01
55	37.88	54.11	95.53	170.43	357.95
56	39.04	1102.68	83.56	148.11	1373.39
57	36.56	46.51	101.04	136.64	320.75
58	38.56	51.96	99.64	152.55	342.71
59	40.84	46.06	94.14	153.98	335.02
60	42.09	50.7	113.38	151.12	357.29
61	993.7	50.12	97.88	165.8	1307.5
62	38.83	42.6	108.14	185.25	374.82
63	37.63	56.96	116.86	138.88	350.33
64	1002.91	1002.17	99.88	144.39	2249.35
65	39.87	43.36	130.45	174.64	388.32
66	42.35	42.42	109.53	153.53	347.83
67	42.45	47.68	63.34	131.53	285
68	38.84	1041.85	99.9	150.28	1330.87
69	38.67	1054.31	88.41	152.6	1333.99
70	43.27	68.4	108.88	144.14	364.69

Table B.1: Example timing data for the example program

Obs.	**A**	**B**	**C**	**D**	**Total**
71	1044.11	41.98	93.36	130.72	1310.17
72	36.06	32.64	124.94	138.72	332.36
73	39.4	49.32	73.88	132.73	295.33
74	39.45	1111.46	106.74	142.23	1399.88
75	44.11	42.7	101.41	145.57	333.79
76	40.72	48.82	93.31	141.51	324.36
77	39.6	62.89	100.39	136.06	338.94
78	42.22	25.16	128.75	130.99	327.12
79	39.52	46.79	105.94	149.2	341.45
80	41.36	52.37	111.23	146.78	351.74
81	1014.24	30.97	102.51	151.26	1298.98
82	37.6	42.52	73.78	176.97	330.87
83	46.49	44.55	106.09	118.6	315.73
84	39.71	35.96	130.43	122.39	328.49
85	36.85	57.39	91.42	139.39	325.05
86	38.83	54.51	132.32	149.36	375.02
87	39.45	1034.04	114.3	139.44	1327.23
88	40.96	38.17	74.48	155.89	309.5
89	41.43	56.9	59.25	125.63	283.21
90	38.67	49.13	121.23	144.23	353.26
91	41.11	56.67	128.39	145.78	371.95
92	39.88	49.23	104.18	134.14	327.43
93	43.12	55.61	94.23	140.03	332.99
94	41.47	57.94	111.33	152.39	363.13
95	42.04	53.53	95.52	137.64	328.73
96	37.92	52.05	114.84	186.46	391.27
97	41.45	1048.8	143.85	149.79	1383.89
98	38.88	52.19	104.43	144.52	340.02
99	39.29	45.19	112.2	159.4	356.08
100	35.44	50.82	107.2	144.02	337.48

Table B.2: Unsmoothed Distribution for Part A

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$
34	0.01	35	0.02	36	0.02
37	0.05	38	0.11	39	0.21
40	0.12	41	0.13	42	0.10
43	0.04	44	0.03	45	0.01
46	0.01	972	0.01	981	0.01
982	0.01	988	0.01	989	0.01
994	0.01	995	0.01	1000	0.01
1003	0.01	1007	0.01	1010	0.01
1014	0.01	1040	0.01	1044	0.01

Table B.3: Unsmoothed Distribution for Part B

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$
25	0.01	31	0.01	33	0.01
34	0.01	36	0.01	38	0.02
39	0.06	41	0.02	42	0.03
43	0.07	45	0.05	46	0.04
47	0.04	48	0.03	49	0.08
50	0.06	51	0.04	52	0.05
54	0.02	55	0.02	56	0.04
57	0.05	58	0.03	60	0.01
62	0.01	63	0.02	64	0.01
66	0.01	67	0.01	68	0.01
69	0.01	1002	0.01	1031	0.01
1034	0.01	1042	0.01	1049	0.02
1054	0.01	1074	0.01	1079	0.01
1103	0.01	1111	0.01		

Table B.4: Unsmoothed Distribution for Part C

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$
59	0.01	63	0.01	64	0.01
68	0.01	69	0.01	74	0.03
75	0.01	78	0.01	79	0.01
81	0.01	82	0.01	83	0.02
84	0.02	88	0.01	89	0.01
90	0.01	91	0.02	93	0.03
94	0.04	95	0.02	96	0.04
97	0.02	98	0.01	99	0.02
100	0.06	101	0.03	102	0.03
103	0.01	104	0.02	106	0.03
107	0.04	108	0.02	109	0.01
110	0.02	111	0.04	112	0.01
113	0.01	114	0.02	115	0.02
117	0.03	118	0.01	119	0.02
120	0.01	121	0.02	123	0.01
124	0.02	125	0.01	126	0.01
128	0.01	129	0.01	130	0.02
132	0.02	139	0.01	144	0.01
177	0.01				

Table B.5: Unsmoothed Distribution for Part D

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$
119	0.02	122	0.01	123	0.01
126	0.01	128	0.02	131	0.03
132	0.02	133	0.02	134	0.01
135	0.02	136	0.02	137	0.02
138	0.01	139	0.05	140	0.01
141	0.01	142	0.04	143	0.01
144	0.06	145	0.05	146	0.04
147	0.03	148	0.02	149	0.02
150	0.02	151	0.06	152	0.01
153	0.03	154	0.03	155	0.02
156	0.04	157	0.02	159	0.02
161	0.02	162	0.02	163	0.01
165	0.01	166	0.02	170	0.02
171	0.01	172	0.01	175	0.02
177	0.01	183	0.01	185	0.01
186	0.01	192	0.01		

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
237	0.0002	238	0.0004	239	0.0004
240	0.0011	241	0.0027	242	0.0052
243	0.0041	244	0.0062	245	0.0096
246	0.0135	247	0.0158	248	0.0191
249	0.0243	250	0.0329	251	0.0439
252	0.0497	253	0.0633	254	0.0841
255	0.1141	256	0.1417	257	0.1641
258	0.2114	259	0.2718	260	0.3390
261	0.3747	262	0.4633	263	0.5646
264	0.6717	265	0.7960	266	0.9294
267	1.0989	268	1.2714	269	1.4821
270	1.6902	271	1.9703	272	2.2495
273	2.5639	274	2.9319	275	3.3193
276	3.7113	277	4.1960	278	4.7719
279	5.2993	280	5.9352	281	6.5611
282	7.3236	283	8.1553	284	8.9724
285	9.8698	286	10.7848	287	11.8535
288	12.8981	289	14.0605	290	15.4063
291	16.8229	292	18.1428	293	19.5799
294	21.1170	295	22.6542	296	24.4520
297	26.3358	298	28.1292	299	30.0043
300	32.1011	301	34.1585	302	36.3618
303	38.8238	304	40.9726	305	43.2978
306	45.8496	307	48.4962	308	51.1038
309	53.9138	310	56.8506	311	59.4863
312	62.3378	313	65.3592	314	68.4863
315	71.8616	316	75.2697	317	78.5651
318	81.5531	319	84.7625	320	88.0450
321	91.4246	322	94.7984	323	98.2855
324	101.4492	325	104.3412	326	107.5390
327	110.4036	328	113.2060	329	115.7463
330	117.9942	331	120.0053	332	121.8344
333	123.8301	334	125.5206	335	126.7851
336	127.3886	337	127.7312	338	127.8387
339	127.8949	340	128.0835	341	127.7878
342	126.8418	343	125.9688	344	124.5087
345	123.0813	346	121.6520	347	119.9535
348	117.8609	349	115.7310	350	113.5304
351	111.0966	352	108.8883	353	106.1318
354	103.5156	355	100.4236	356	97.6227
357	94.6687	358	91.8898	359	88.8445
360	85.6816	361	82.7584	362	79.4105
363	76.5277	364	73.4477	365	70.4693

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
366	67.2713	367	64.4909	368	61.2848
369	58.3587	370	55.6777	371	52.9517
372	50.3328	373	47.7679	374	45.1724
375	42.4426	376	40.1944	377	37.7433
378	35.5680	379	33.4140	380	31.4748
381	29.4864	382	27.7476	383	25.8973
384	24.2039	385	22.5704	386	20.9388
387	19.6468	388	18.2725	389	17.0943
390	15.9051	391	14.9498	392	13.7596
393	12.8619	394	11.9711	395	11.0870
396	10.3833	397	9.6908	398	9.0543
399	8.4506	400	7.9508	401	7.4091
402	6.9829	403	6.5763	404	6.1956
405	5.8541	406	5.5203	407	5.1287
408	4.8430	409	4.6002	410	4.3841
411	4.1758	412	3.9745	413	3.6762
414	3.4186	415	3.2159	416	3.0191
417	2.8832	418	2.7374	419	2.5464
420	2.3935	421	2.2150	422	2.0191
423	1.8882	424	1.7743	425	1.6174
426	1.5065	427	1.4219	428	1.2836
429	1.2202	430	1.1090	431	1.0075
432	0.9365	433	0.8560	434	0.7985
435	0.7447	436	0.7017	437	0.6449
438	0.6138	439	0.5543	440	0.5220
441	0.4774	442	0.4431	443	0.4159
444	0.3866	445	0.3538	446	0.3215
447	0.3098	448	0.2913	449	0.2805
450	0.2704	451	0.2545	452	0.2166
453	0.2078	454	0.1842	455	0.1664
456	0.1604	457	0.1539	458	0.1431
459	0.1296	460	0.1104	461	0.0864
462	0.0738	463	0.0597	464	0.0571
465	0.0543	466	0.0459	467	0.0404
468	0.0357	469	0.0284	470	0.0258
471	0.0220	472	0.0162	473	0.0128
474	0.0106	475	0.0086	476	0.0075
477	0.0065	478	0.0043	479	0.0031
480	0.0018	481	0.0009	482	0.0005
483	0.0002	484	0.0001	1175	0.0002
1178	0.0001	1179	0.0003	1180	0.0002
1181	0.0002	1182	0.0002	1183	0.0004
1184	0.0010	1185	0.0007	1186	0.0006

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1187	0.0010	1188	0.0021	1189	0.0027
1190	0.0021	1191	0.0028	1192	0.0042
1193	0.0073	1194	0.0051	1195	0.0065
1196	0.0092	1197	0.0125	1198	0.0149
1199	0.0157	1200	0.0191	1201	0.0230
1202	0.0310	1203	0.0317	1204	0.0368
1205	0.0455	1206	0.0540	1207	0.0615
1208	0.0716	1209	0.0813	1210	0.0931
1211	0.1115	1212	0.1225	1213	0.1415
1214	0.1591	1215	0.1847	1216	0.2102
1217	0.2351	1218	0.2649	1219	0.2907
1220	0.3352	1221	0.3785	1222	0.4146
1223	0.4623	1224	0.5279	1225	0.5746
1226	0.6290	1227	0.7043	1228	0.7702
1229	0.8390	1230	0.9369	1231	1.0176
1232	1.0992	1233	1.2359	1234	1.3501
1235	1.4425	1236	1.5763	1237	1.7000
1238	1.8290	1239	2.0006	1240	2.1810
1241	2.3212	1242	2.4791	1243	2.6983
1244	2.8724	1245	3.0593	1246	3.3337
1247	3.5378	1248	3.7464	1249	4.0060
1250	4.2824	1251	4.5256	1252	4.8227
1253	5.1377	1254	5.3911	1255	5.6947
1256	6.0514	1257	6.3810	1258	6.7250
1259	7.1313	1260	7.4398	1261	7.7499
1262	8.1438	1263	8.5305	1264	8.9335
1265	9.3658	1266	9.8288	1267	10.1653
1268	10.5753	1269	10.9944	1270	11.4251
1271	11.9306	1272	12.3039	1273	12.7136
1274	13.1116	1275	13.5149	1276	14.0145
1277	14.4199	1278	14.8983	1279	15.2419
1280	15.5909	1281	15.9533	1282	16.3755
1283	16.8412	1284	17.1761	1285	17.5463
1286	17.7569	1287	18.0332	1288	18.3675
1289	18.6977	1290	18.9861	1291	19.2023
1292	19.3515	1293	19.4641	1294	19.7032
1295	19.9265	1296	20.1320	1297	20.1877
1298	20.3045	1299	20.3505	1300	20.4329
1301	20.5719	1302	20.6657	1303	20.6820
1304	20.6257	1305	20.6402	1306	20.6920
1307	20.7140	1308	20.7917	1309	20.7521
1310	20.6333	1311	20.6214	1312	20.5181
1313	20.5288	1314	20.4849	1315	20.4818

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1316	20.4148	1317	20.2674	1318	20.2491
1319	20.0621	1320	20.0895	1321	19.9523
1322	19.8909	1323	19.7624	1324	19.5186
1325	19.4881	1326	19.4924	1327	19.3809
1328	19.2281	1329	19.0301	1330	18.7255
1331	18.6713	1332	18.5893	1333	18.4763
1334	18.2507	1335	18.0415	1336	17.7795
1337	17.5316	1338	17.4149	1339	17.1286
1340	16.9036	1341	16.5974	1342	16.3333
1343	16.0724	1344	15.8184	1345	15.5254
1346	15.3252	1347	14.9458	1348	14.6700
1349	14.3735	1350	14.0653	1351	13.8909
1352	13.6097	1353	13.3971	1354	13.0274
1355	12.8640	1356	12.6049	1357	12.3944
1358	12.1839	1359	11.9232	1360	11.5976
1361	11.2897	1362	11.1081	1363	10.8708
1364	10.6985	1365	10.4018	1366	10.1438
1367	9.8625	1368	9.6437	1369	9.4317
1370	9.1920	1371	8.9489	1372	8.7079
1373	8.4752	1374	8.2889	1375	8.0813
1376	7.8925	1377	7.6981	1378	7.5026
1379	7.3834	1380	7.1991	1381	7.0844
1382	6.9077	1383	6.7985	1384	6.6173
1385	6.4824	1386	6.3747	1387	6.2773
1388	6.2220	1389	6.1004	1390	5.9512
1391	5.7910	1392	5.6909	1393	5.5491
1394	5.4813	1395	5.3770	1396	5.2440
1397	5.1077	1398	4.9662	1399	4.8213
1400	4.6776	1401	4.5614	1402	4.4236
1403	4.3312	1404	4.1933	1405	4.0592
1406	3.9127	1407	3.7838	1408	3.6318
1409	3.5057	1410	3.3659	1411	3.2577
1412	3.1579	1413	3.0327	1414	2.8972
1415	2.7807	1416	2.6744	1417	2.5381
1418	2.4672	1419	2.3303	1420	2.2215
1421	2.1189	1422	2.0019	1423	1.8999
1424	1.7984	1425	1.7202	1426	1.6297
1427	1.5646	1428	1.4599	1429	1.3684
1430	1.2758	1431	1.2015	1432	1.1230
1433	1.0750	1434	1.0130	1435	0.9700
1436	0.9172	1437	0.8397	1438	0.7782
1439	0.7182	1440	0.6741	1441	0.6258
1442	0.5973	1443	0.5463	1444	0.5161

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1445	0.4833	1446	0.4397	1447	0.3996
1448	0.3727	1449	0.3470	1450	0.3222
1451	0.3061	1452	0.2785	1453	0.2548
1454	0.2300	1455	0.2181	1456	0.2049
1457	0.1930	1458	0.1788	1459	0.1690
1460	0.1560	1461	0.1424	1462	0.1304
1463	0.1300	1464	0.1278	1465	0.1261
1466	0.1235	1467	0.1067	1468	0.0972
1469	0.0917	1470	0.0865	1471	0.0862
1472	0.0871	1473	0.0815	1474	0.0770
1475	0.0701	1476	0.0608	1477	0.0556
1478	0.0557	1479	0.0461	1480	0.0506
1481	0.0478	1482	0.0412	1483	0.0392
1484	0.0327	1485	0.0283	1486	0.0246
1487	0.0204	1488	0.0208	1489	0.0211
1490	0.0175	1491	0.0161	1492	0.0151
1493	0.0142	1494	0.0124	1495	0.0101
1496	0.0109	1497	0.0107	1498	0.0091
1499	0.0086	1500	0.0073	1501	0.0075
1502	0.0096	1503	0.0073	1504	0.0098
1505	0.0078	1506	0.0054	1507	0.0047
1508	0.0031	1509	0.0033	1510	0.0044
1511	0.0052	1512	0.0058	1513	0.0056
1514	0.0040	1515	0.0032	1516	0.0020
1517	0.0014	1518	0.0016	1519	0.0023
1520	0.0013	1521	0.0013	1522	0.0010
1523	0.0004	1524	0.0003	1525	0.0001
1526	0.0001	2152	0.0002	2155	0.0001
2156	0.0003	2157	0.0002	2159	0.0002
2160	0.0002	2161	0.0007	2162	0.0004
2163	0.0001	2164	0.0006	2165	0.0010
2166	0.0010	2167	0.0009	2168	0.0012
2169	0.0014	2170	0.0018	2171	0.0020
2172	0.0018	2173	0.0019	2174	0.0031
2175	0.0031	2176	0.0037	2177	0.0045
2178	0.0041	2179	0.0053	2180	0.0059
2181	0.0068	2182	0.0061	2183	0.0075
2184	0.0100	2185	0.0090	2186	0.0108
2187	0.0131	2188	0.0131	2189	0.0143
2190	0.0161	2191	0.0162	2192	0.0187
2193	0.0233	2194	0.0224	2195	0.0215
2196	0.0275	2197	0.0302	2198	0.0309
2199	0.0365	2200	0.0351	2201	0.0370

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2202	0.0444	2203	0.0500	2204	0.0512
2205	0.0508	2206	0.0571	2207	0.0594
2208	0.0648	2209	0.0763	2210	0.0767
2211	0.0802	2212	0.0898	2213	0.0946
2214	0.0949	2215	0.1045	2216	0.1169
2217	0.1181	2218	0.1301	2219	0.1391
2220	0.1385	2221	0.1533	2222	0.1640
2223	0.1706	2224	0.1822	2225	0.1924
2226	0.1989	2227	0.2076	2228	0.2259
2229	0.2378	2230	0.2488	2231	0.2638
2232	0.2677	2233	0.2781	2234	0.2986
2235	0.3190	2236	0.3313	2237	0.3399
2238	0.3510	2239	0.3705	2240	0.3868
2241	0.4115	2242	0.4248	2243	0.4351
2244	0.4611	2245	0.4699	2246	0.4914
2247	0.5083	2248	0.5277	2249	0.5560
2250	0.5761	2251	0.6014	2252	0.6059
2253	0.6296	2254	0.6522	2255	0.6731
2256	0.7114	2257	0.7179	2258	0.7456
2259	0.7765	2260	0.7843	2261	0.8166
2262	0.8351	2263	0.8563	2264	0.8861
2265	0.9092	2266	0.9289	2267	0.9453
2268	0.9760	2269	1.0033	2270	1.0299
2271	1.0516	2272	1.0521	2273	1.0720
2274	1.1058	2275	1.1373	2276	1.1698
2277	1.1794	2278	1.1734	2279	1.1929
2280	1.2065	2281	1.2583	2282	1.2821
2283	1.2738	2284	1.2928	2285	1.2732
2286	1.3083	2287	1.3394	2288	1.3536
2289	1.3602	2290	1.3442	2291	1.3627
2292	1.3573	2293	1.3828	2294	1.3907
2295	1.3826	2296	1.3937	2297	1.3794
2298	1.3884	2299	1.3944	2300	1.3892
2301	1.3973	2302	1.3942	2303	1.3765
2304	1.3666	2305	1.3677	2306	1.3717
2307	1.3874	2308	1.3713	2309	1.3415
2310	1.3213	2311	1.3322	2312	1.3457
2313	1.3326	2314	1.3317	2315	1.2898
2316	1.2750	2317	1.2839	2318	1.2793
2319	1.2880	2320	1.2585	2321	1.2489
2322	1.2308	2323	1.2222	2324	1.2277
2325	1.2050	2326	1.2063	2327	1.1825
2328	1.1846	2329	1.1607	2330	1.1534

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2331	1.1564	2332	1.1382	2333	1.1312
2334	1.0970	2335	1.0923	2336	1.0788
2337	1.0842	2338	1.0736	2339	1.0560
2340	1.0281	2341	0.9998	2342	0.9981
2343	0.9887	2344	0.9936	2345	0.9701
2346	0.9374	2347	0.9202	2348	0.9089
2349	0.8994	2350	0.8854	2351	0.8722
2352	0.8543	2353	0.8277	2354	0.8210
2355	0.7984	2356	0.7939	2357	0.7717
2358	0.7615	2359	0.7461	2360	0.7226
2361	0.7091	2362	0.6873	2363	0.6844
2364	0.6588	2365	0.6466	2366	0.6179
2367	0.6054	2368	0.5952	2369	0.5836
2370	0.5637	2371	0.5359	2372	0.5198
2373	0.5024	2374	0.4966	2375	0.4758
2376	0.4643	2377	0.4451	2378	0.4276
2379	0.4121	2380	0.4035	2381	0.3840
2382	0.3769	2383	0.3589	2384	0.3536
2385	0.3342	2386	0.3238	2387	0.3116
2388	0.3030	2389	0.2975	2390	0.2791
2391	0.2745	2392	0.2586	2393	0.2561
2394	0.2421	2395	0.2366	2396	0.2247
2397	0.2170	2398	0.2073	2399	0.2029
2400	0.1957	2401	0.1833	2402	0.1781
2403	0.1667	2404	0.1651	2405	0.1534
2406	0.1542	2407	0.1408	2408	0.1407
2409	0.1283	2410	0.1254	2411	0.1167
2412	0.1129	2413	0.1103	2414	0.1052
2415	0.0978	2416	0.0915	2417	0.0922
2418	0.0816	2419	0.0824	2420	0.0756
2421	0.0762	2422	0.0668	2423	0.0668
2424	0.0603	2425	0.0580	2426	0.0555
2427	0.0529	2428	0.0514	2429	0.0452
2430	0.0444	2431	0.0395	2432	0.0415
2433	0.0354	2434	0.0357	2435	0.0306
2436	0.0324	2437	0.0286	2438	0.0275
2439	0.0245	2440	0.0234	2441	0.0239
2442	0.0203	2443	0.0190	2444	0.0167
2445	0.0173	2446	0.0159	2447	0.0159
2448	0.0116	2449	0.0127	2450	0.0113
2451	0.0108	2452	0.0096	2453	0.0094
2454	0.0083	2455	0.0072	2456	0.0074
2457	0.0067	2458	0.0065	2459	0.0046

Table B.6: Unsmoothed Distribution for Convolution

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2460	0.0056	2461	0.0051	2462	0.0049
2463	0.0034	2464	0.0040	2465	0.0041
2466	0.0040	2467	0.0032	2468	0.0033
2469	0.0028	2470	0.0029	2471	0.0030
2472	0.0024	2473	0.0025	2474	0.0022
2475	0.0022	2476	0.0020	2477	0.0022
2478	0.0013	2479	0.0021	2480	0.0013
2481	0.0014	2482	0.0011	2483	0.0016
2484	0.0007	2485	0.0011	2486	0.0010
2487	0.0009	2488	0.0006	2489	0.0005
2490	0.0007	2491	0.0005	2492	0.0002
2493	0.0003	2494	0.0007	2495	0.0004
2496	0.0001	2497	0.0002	2498	0.0004
2499	0.0003	2500	0.0001	2501	0.0001
2502	0.0002	2503	0.0004	2504	0.0001
2505	0.0002	2506	0.0001	2507	0.0003
2509	0.0002	2510	0.0001	2511	0.0001
2512	0.0001	2513	0.0001	2514	0.0001
2515	0.0001	2516	0.0001	2517	0.0001
2518	0.0001	2520	0.0001	2524	0.0001

Table B.7: Smoothed Distribution for Part A

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$
32	0.0015	33	0.0050	34	0.0100
35	0.0195	36	0.0380	37	0.0755
38	0.1060	39	0.1360	40	0.1355
41	0.1205	42	0.0865	43	0.0590
44	0.0355	45	0.0170	46	0.0095
47	0.0035	48	0.0015	970	0.0015
971	0.0020	972	0.0030	973	0.0020
974	0.0015	979	0.0015	980	0.0035
981	0.0050	982	0.0050	983	0.0035
984	0.0015	986	0.0015	987	0.0035
988	0.0050	989	0.0050	990	0.0035
991	0.0015	992	0.0015	993	0.0035
994	0.0050	995	0.0050	996	0.0035
997	0.0015	998	0.0015	999	0.0020
1000	0.0030	1001	0.0035	1002	0.0035
1003	0.0030	1004	0.0020	1005	0.0030
1006	0.0020	1007	0.0030	1008	0.0035

Table B.7: Smoothed Distribution for Part A

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{A}}(\mathbf{x})$
1009	0.0035	1010	0.0030	1011	0.0020
1012	0.0030	1013	0.0020	1014	0.0030
1015	0.0020	1016	0.0015	1038	0.0015
1039	0.0020	1040	0.0030	1041	0.0020
1042	0.0030	1043	0.0020	1044	0.0030
1045	0.0020	1046	0.0015		

Table B.8: Smoothed Distribution for Part B

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{B}}(\mathbf{x})$
23	0.0015	24	0.0020	25	0.0030
26	0.0020	27	0.0015	29	0.0015
30	0.0020	31	0.0045	32	0.0055
33	0.0065	34	0.0065	35	0.0055
36	0.0075	37	0.0150	38	0.0195
39	0.0250	40	0.0235	41	0.0315
42	0.0270	43	0.0375	44	0.0345
45	0.0395	46	0.0345	47	0.0455
48	0.0480	49	0.0540	50	0.0540
51	0.0460	52	0.0350	53	0.0230
54	0.0235	55	0.0255	56	0.0335
57	0.0320	58	0.0265	59	0.0155
60	0.0090	61	0.0070	62	0.0100
63	0.0100	64	0.0100	65	0.0085
66	0.0080	67	0.0085	68	0.0085
69	0.0065	70	0.0035	71	0.0015
1000	0.0015	1001	0.0020	1002	0.0030
1003	0.0020	1004	0.0015	1029	0.0015
1030	0.0020	1031	0.0030	1032	0.0035
1033	0.0035	1034	0.0030	1035	0.0020
1036	0.0015	1040	0.0015	1041	0.0020
1042	0.0030	1043	0.0020	1044	0.0015
1047	0.0030	1048	0.0040	1049	0.0060
1050	0.0040	1051	0.0030	1052	0.0015
1053	0.0020	1054	0.0030	1055	0.0020
1056	0.0015	1072	0.0015	1073	0.0020
1074	0.0030	1075	0.0020	1076	0.0015
1077	0.0015	1078	0.0020	1079	0.0030
1080	0.0020	1081	0.0015	1101	0.0015
1102	0.0020	1103	0.0030	1104	0.0020
1105	0.0015	1109	0.0015	1110	0.0020
1111	0.0030	1112	0.0020	1113	0.0015

Table B.9: Smoothed Distribution for Part C

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{C}}(\mathbf{x})$
57	0.0015	58	0.0020	59	0.0030
60	0.0020	61	0.0030	62	0.0035
63	0.0050	64	0.0050	65	0.0035
66	0.0030	67	0.0035	68	0.0050
69	0.0050	70	0.0035	71	0.0015
72	0.0045	73	0.0075	74	0.0110
75	0.0090	76	0.0080	77	0.0050
78	0.0050	79	0.0065	80	0.0070
81	0.0095	82	0.0120	83	0.0135
84	0.0115	85	0.0070	86	0.0045
87	0.0035	88	0.0065	89	0.0100
90	0.0105	91	0.0140	92	0.0175
93	0.0230	94	0.0280	95	0.0295
96	0.0275	97	0.0220	98	0.0260
99	0.0275	100	0.0340	101	0.0315
102	0.0290	103	0.0175	104	0.0170
105	0.0175	106	0.0230	107	0.0235
108	0.0235	109	0.0230	110	0.0205
111	0.0210	112	0.0190	113	0.0180
114	0.0135	115	0.0160	116	0.0145
117	0.0170	118	0.0145	119	0.0175
120	0.0125	121	0.0125	122	0.0105
123	0.0115	124	0.0115	125	0.0105
126	0.0095	127	0.0070	128	0.0095
129	0.0090	130	0.0125	131	0.0095
132	0.0090	133	0.0040	134	0.0030
137	0.0015	138	0.0020	139	0.0030
140	0.0020	141	0.0015	142	0.0015
143	0.0020	144	0.0030	145	0.0020
146	0.0015	175	0.0015	176	0.0020
177	0.0030	178	0.0020	179	0.0015

Table B.10: Smoothed Distribution for Part D

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$
117	0.0030	118	0.0040	119	0.0060
120	0.0055	121	0.0065	122	0.0050
123	0.0050	124	0.0050	125	0.0035
126	0.0060	127	0.0060	128	0.0075
129	0.0085	130	0.0120	131	0.0160
132	0.0175	133	0.0195	134	0.0170
135	0.0180	136	0.0170	137	0.0225

Table B.10: Smoothed Distribution for Part D

$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$	$\mathbf{x}$	$\hat{\mathbf{p}}_{\mathbf{D}}(\mathbf{x})$
138	0.0215	139	0.0235	140	0.0225
141	0.0220	142	0.0265	143	0.0320
144	0.0420	145	0.0410	146	0.0400
147	0.0315	148	0.0250	149	0.0275
150	0.0265	151	0.0315	152	0.0285
153	0.0290	154	0.0265	155	0.0275
156	0.0245	157	0.0200	158	0.0140
159	0.0120	160	0.0110	161	0.0145
162	0.0120	163	0.0115	164	0.0100
165	0.0085	166	0.0080	167	0.0055
168	0.0060	169	0.0055	170	0.0095
171	0.0090	172	0.0080	173	0.0065
174	0.0055	175	0.0075	176	0.0060
177	0.0060	178	0.0020	179	0.0015
181	0.0015	182	0.0020	183	0.0045
184	0.0055	185	0.0065	186	0.0050
187	0.0035	188	0.0015	190	0.0015
191	0.0020	192	0.0030	193	0.0020
194	0.0015				

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
234	0.0001	235	0.0001	236	0.0003
237	0.0005	238	0.0009	239	0.0015
240	0.0023	241	0.0034	242	0.0047
243	0.0064	244	0.0086	245	0.0112
246	0.0145	247	0.0187	248	0.0238
249	0.0304	250	0.0387	251	0.0493
252	0.0627	253	0.0795	254	0.1006
255	0.1264	256	0.1579	257	0.1956
258	0.2404	259	0.2932	260	0.3550
261	0.4273	262	0.5114	263	0.6087
264	0.7209	265	0.8493	266	0.9954
267	1.1609	268	1.3478	269	1.5582
270	1.7942	271	2.0581	272	2.3521
273	2.6784	274	3.0396	275	3.4386
276	3.8781	277	4.3609	278	4.8897
279	5.4670	280	6.0951	281	6.7757
282	7.5106	283	8.3009	284	9.1484
285	10.0560	286	11.0274	287	12.0672
288	13.1790	289	14.3644	290	15.6229
291	16.9533	292	18.3550	293	19.8291
294	21.3783	295	23.0056	296	24.7129
297	26.5008	298	28.3693	299	30.3178
300	32.3457	301	34.4522	302	36.6361
303	38.8969	304	41.2348	305	43.6515
306	46.1479	307	48.7230	308	51.3740
309	54.0972	310	56.8913	311	59.7585
312	62.7028	313	65.7256	314	68.8219
315	71.9789	316	75.1793	317	78.4076
318	81.6537	319	84.9130	320	88.1817
321	91.4514	322	94.7061	323	97.9233
324	101.0769	325	104.1409	326	107.0909
327	109.9045	328	112.5624	329	115.0489
330	117.3515	331	119.4581	332	121.3529
333	123.0150	334	124.4212	335	125.5531
336	126.4035	337	126.9781	338	127.2892
339	127.3487	340	127.1615	341	126.7276
342	126.0460	343	125.1220	344	123.9677
345	122.6008	346	121.0391	347	119.2991
348	117.3947	349	115.3377	350	113.1388
351	110.8066	352	108.3498	353	105.7775
354	103.1020	355	100.3375	356	97.5004
357	94.6040	358	91.6598	359	88.6763
360	85.6617	361	82.6255	362	79.5775

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
363	76.5281	364	73.4856	365	70.4585
366	67.4553	367	64.4869	368	61.5644
369	58.6984	370	55.8952	371	53.1564
372	50.4806	373	47.8661	374	45.3149
375	42.8327	376	40.4288	377	38.1121
378	35.8894	379	33.7623	380	31.7294
381	29.7859	382	27.9279	383	26.1527
384	24.4609	385	22.8553	386	21.3390
387	19.9138	388	18.5781	389	17.3277
390	16.1562	391	15.0582	392	14.0291
393	13.0669	394	12.1706	395	11.3395
396	10.5719	397	9.8650	398	9.2150
399	8.6178	400	8.0693	401	7.5652
402	7.1013	403	6.6729	404	6.2759
405	5.9068	406	5.5634	407	5.2446
408	4.9488	409	4.6738	410	4.4158
411	4.1705	412	3.9348	413	3.7073
414	3.4890	415	3.2814	416	3.0855
417	2.9005	418	2.7242	419	2.5542
420	2.3889	421	2.2282	422	2.0736
423	1.9264	424	1.7879	425	1.6586
426	1.5380	427	1.4252	428	1.3193
429	1.2198	430	1.1266	431	1.0398
432	0.9599	433	0.8872	434	0.8215
435	0.7621	436	0.7081	437	0.6583
438	0.6119	439	0.5685	440	0.5277
441	0.4895	442	0.4539	443	0.4211
444	0.3911	445	0.3639	446	0.3396
447	0.3179	448	0.2982	449	0.2797
450	0.2619	451	0.2441	452	0.2264
453	0.2092	454	0.1928	455	0.1776
456	0.1635	457	0.1499	458	0.1364
459	0.1226	460	0.1087	461	0.0951
462	0.0825	463	0.0714	464	0.0620
465	0.0541	466	0.0474	467	0.0415
468	0.0361	469	0.0311	470	0.0265
471	0.0223	472	0.0185	473	0.0152
474	0.0124	475	0.0100	476	0.0081
477	0.0064	478	0.0049	479	0.0037
480	0.0027	481	0.0018	482	0.0012
483	0.0007	484	0.0004	485	0.0002
486	0.0001	1175	0.0001	1176	0.0001
1177	0.0001	1178	0.0001	1179	0.0002

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1180	0.0002	1181	0.0003	1182	0.0004
1183	0.0005	1184	0.0007	1185	0.0009
1186	0.0012	1187	0.0015	1188	0.0019
1189	0.0025	1190	0.0031	1191	0.0038
1192	0.0047	1193	0.0058	1194	0.0071
1195	0.0086	1196	0.0104	1197	0.0126
1198	0.0152	1199	0.0181	1200	0.0215
1201	0.0255	1202	0.0301	1203	0.0353
1204	0.0414	1205	0.0483	1206	0.0562
1207	0.0651	1208	0.0753	1209	0.0867
1210	0.0995	1211	0.1139	1212	0.1302
1213	0.1484	1214	0.1686	1215	0.1912
1216	0.2162	1217	0.2438	1218	0.2744
1219	0.3081	1220	0.3453	1221	0.3862
1222	0.4309	1223	0.4795	1224	0.5322
1225	0.5890	1226	0.6505	1227	0.7169
1228	0.7887	1229	0.8666	1230	0.9509
1231	1.0419	1232	1.1398	1233	1.2443
1234	1.3555	1235	1.4736	1236	1.5990
1237	1.7325	1238	1.8743	1239	2.0247
1240	2.1838	1241	2.3517	1242	2.5286
1243	2.7151	1244	2.9114	1245	3.1179
1246	3.3345	1247	3.5614	1248	3.7986
1249	4.0462	1250	4.3044	1251	4.5731
1252	4.8525	1253	5.1426	1254	5.4437
1255	5.7559	1256	6.0788	1257	6.4112
1258	6.7519	1259	7.0999	1260	7.4553
1261	7.8193	1262	8.1931	1263	8.5773
1264	8.9712	1265	9.3729	1266	9.7805
1267	10.1925	1268	10.6082	1269	11.0268
1270	11.4478	1271	11.8700	1272	12.2926
1273	12.7153	1274	13.1380	1275	13.5601
1276	13.9802	1277	14.3960	1278	14.8057
1279	15.2080	1280	15.6030	1281	15.9907
1282	16.3696	1283	16.7372	1284	17.0905
1285	17.4276	1286	17.7479	1287	18.0519
1288	18.3397	1289	18.6103	1290	18.8622
1291	19.0946	1292	19.3083	1293	19.5050
1294	19.6861	1295	19.8517	1296	20.0012
1297	20.1336	1298	20.2493	1299	20.3492
1300	20.4344	1301	20.5056	1302	20.5631
1303	20.6080	1304	20.6416	1305	20.6658
1306	20.6813	1307	20.6876	1308	20.6832

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1309	20.6674	1310	20.6407	1311	20.6056
1312	20.5647	1313	20.5197	1314	20.4702
1315	20.4148	1316	20.3523	1317	20.2819
1318	20.2042	1319	20.1199	1320	20.0299
1321	19.9346	1322	19.8351	1323	19.7324
1324	19.6269	1325	19.5176	1326	19.4022
1327	19.2779	1328	19.1437	1329	19.0004
1330	18.8504	1331	18.6951	1332	18.5338
1333	18.3641	1334	18.1828	1335	17.9885
1336	17.7813	1337	17.5625	1338	17.3337
1339	17.0958	1340	16.8495	1341	16.5956
1342	16.3353	1343	16.0696	1344	15.7994
1345	15.5248	1346	15.2467	1347	14.9666
1348	14.6870	1349	14.4106	1350	14.1395
1351	13.8744	1352	13.6153	1353	13.3614
1354	13.1117	1355	12.8652	1356	12.6199
1357	12.3741	1358	12.1259	1359	11.8754
1360	11.6238	1361	11.3732	1362	11.1250
1363	10.8794	1364	10.6353	1365	10.3916
1366	10.1480	1367	9.9050	1368	9.6639
1369	9.4257	1370	9.1913	1371	8.9615
1372	8.7371	1373	8.5193	1374	8.3091
1375	8.1073	1376	7.9146	1377	7.7310
1378	7.5565	1379	7.3903	1380	7.2314
1381	7.0790	1382	6.9323	1383	6.7917
1384	6.6575	1385	6.5302	1386	6.4088
1387	6.2916	1388	6.1758	1389	6.0593
1390	5.9411	1391	5.8216	1392	5.7020
1393	5.5828	1394	5.4636	1395	5.3428
1396	5.2191	1397	5.0919	1398	4.9617
1399	4.8300	1400	4.6984	1401	4.5679
1402	4.4384	1403	4.3092	1404	4.1790
1405	4.0474	1406	3.9144	1407	3.7812
1408	3.6489	1409	3.5190	1410	3.3920
1411	3.2680	1412	3.1463	1413	3.0265
1414	2.9082	1415	2.7911	1416	2.6756
1417	2.5616	1418	2.4491	1419	2.3382
1420	2.2291	1421	2.1222	1422	2.0182
1423	1.9178	1424	1.8211	1425	1.7279
1426	1.6375	1427	1.5494	1428	1.4631
1429	1.3794	1430	1.2990	1431	1.2230
1432	1.1520	1433	1.0856	1434	1.0229
1435	0.9627	1436	0.9040	1437	0.8467

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
1438	0.7910	1439	0.7378	1440	0.6877
1441	0.6411	1442	0.5976	1443	0.5569
1444	0.5183	1445	0.4815	1446	0.4466
1447	0.4137	1448	0.3831	1449	0.3548
1450	0.3287	1451	0.3044	1452	0.2819
1453	0.2611	1454	0.2419	1455	0.2246
1456	0.2088	1457	0.1946	1458	0.1815
1459	0.1696	1460	0.1588	1461	0.1492
1462	0.1410	1463	0.1338	1464	0.1273
1465	0.1212	1466	0.1150	1467	0.1087
1468	0.1026	1469	0.0969	1470	0.0918
1471	0.0872	1472	0.0829	1473	0.0786
1474	0.0740	1475	0.0691	1476	0.0643
1477	0.0597	1478	0.0555	1479	0.0517
1480	0.0481	1481	0.0446	1482	0.0411
1483	0.0374	1484	0.0337	1485	0.0301
1486	0.0269	1487	0.0241	1488	0.0218
1489	0.0199	1490	0.0182	1491	0.0167
1492	0.0153	1493	0.0140	1494	0.0129
1495	0.0118	1496	0.0109	1497	0.0102
1498	0.0096	1499	0.0091	1500	0.0087
1501	0.0084	1502	0.0081	1503	0.0078
1504	0.0073	1505	0.0068	1506	0.0061
1507	0.0055	1508	0.0050	1509	0.0047
1510	0.0046	1511	0.0045	1512	0.0044
1513	0.0042	1514	0.0038	1515	0.0033
1516	0.0028	1517	0.0024	1518	0.0020
1519	0.0017	1520	0.0014	1521	0.0012
1522	0.0009	1523	0.0007	1524	0.0005
1525	0.0003	1526	0.0002	1527	0.0001
1528	0.0001	2152	0.0001	2153	0.0001
2154	0.0001	2155	0.0001	2156	0.0001
2157	0.0002	2158	0.0002	2159	0.0003
2160	0.0003	2161	0.0004	2162	0.0005
2163	0.0006	2164	0.0007	2165	0.0008
2166	0.0009	2167	0.0011	2168	0.0013
2169	0.0015	2170	0.0017	2171	0.0019
2172	0.0022	2173	0.0025	2174	0.0029
2175	0.0033	2176	0.0037	2177	0.0042
2178	0.0047	2179	0.0053	2180	0.0059
2181	0.0066	2182	0.0073	2183	0.0082
2184	0.0091	2185	0.0101	2186	0.0112
2187	0.0123	2188	0.0135	2189	0.0148

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2190	0.0162	2191	0.0177	2192	0.0194
2193	0.0211	2194	0.0230	2195	0.0250
2196	0.0272	2197	0.0295	2198	0.0320
2199	0.0347	2200	0.0376	2201	0.0406
2202	0.0438	2203	0.0471	2204	0.0506
2205	0.0543	2206	0.0583	2207	0.0626
2208	0.0673	2209	0.0723	2210	0.0775
2211	0.0829	2212	0.0884	2213	0.0943
2214	0.1005	2215	0.1071	2216	0.1141
2217	0.1215	2218	0.1291	2219	0.1371
2220	0.1454	2221	0.1540	2222	0.1630
2223	0.1724	2224	0.1820	2225	0.1921
2226	0.2026	2227	0.2135	2228	0.2249
2229	0.2365	2230	0.2485	2231	0.2607
2232	0.2733	2233	0.2864	2234	0.2999
2235	0.3139	2236	0.3282	2237	0.3429
2238	0.3580	2239	0.3737	2240	0.3898
2241	0.4064	2242	0.4232	2243	0.4403
2244	0.4576	2245	0.4754	2246	0.4938
2247	0.5130	2248	0.5327	2249	0.5528
2250	0.5731	2251	0.5934	2252	0.6139
2253	0.6348	2254	0.6563	2255	0.6783
2256	0.7008	2257	0.7235	2258	0.7462
2259	0.7689	2260	0.7915	2261	0.8142
2262	0.8370	2263	0.8599	2264	0.8830
2265	0.9062	2266	0.9294	2267	0.9527
2268	0.9757	2269	0.9983	2270	1.0205
2271	1.0422	2272	1.0639	2273	1.0856
2274	1.1072	2275	1.1282	2276	1.1484
2277	1.1677	2278	1.1863	2279	1.2045
2280	1.2224	2281	1.2399	2282	1.2564
2283	1.2718	2284	1.2860	2285	1.2996
2286	1.3125	2287	1.3248	2288	1.3359
2289	1.3457	2290	1.3541	2291	1.3613
2292	1.3677	2293	1.3734	2294	1.3782
2295	1.3821	2296	1.3850	2297	1.3871
2298	1.3882	2299	1.3886	2300	1.3880
2301	1.3864	2302	1.3839	2303	1.3808
2304	1.3772	2305	1.3733	2306	1.3689
2307	1.3636	2308	1.3576	2309	1.3509
2310	1.3439	2311	1.3368	2312	1.3295
2313	1.3215	2314	1.3128	2315	1.3037
2316	1.2946	2317	1.2856	2318	1.2768

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2319	1.2680	2320	1.2586	2321	1.2489
2322	1.2388	2323	1.2286	2324	1.2186
2325	1.2086	2326	1.1986	2327	1.1884
2328	1.1781	2329	1.1677	2330	1.1569
2331	1.1457	2332	1.1340	2333	1.1220
2334	1.1099	2335	1.0978	2336	1.0857
2337	1.0732	2338	1.0601	2339	1.0462
2340	1.0319	2341	1.0174	2342	1.0032
2343	0.9890	2344	0.9746	2345	0.9597
2346	0.9443	2347	0.9287	2348	0.9131
2349	0.8976	2350	0.8823	2351	0.8668
2352	0.8512	2353	0.8355	2354	0.8198
2355	0.8042	2356	0.7888	2357	0.7733
2358	0.7577	2359	0.7418	2360	0.7256
2361	0.7092	2362	0.6927	2363	0.6761
2364	0.6594	2365	0.6427	2366	0.6260
2367	0.6093	2368	0.5926	2369	0.5758
2370	0.5588	2371	0.5418	2372	0.5249
2373	0.5085	2374	0.4924	2375	0.4767
2376	0.4613	2377	0.4460	2378	0.4310
2379	0.4163	2380	0.4022	2381	0.3885
2382	0.3753	2383	0.3624	2384	0.3500
2385	0.3379	2386	0.3262	2387	0.3149
2388	0.3040	2389	0.2934	2390	0.2831
2391	0.2731	2392	0.2633	2393	0.2538
2394	0.2445	2395	0.2354	2396	0.2266
2397	0.2181	2398	0.2097	2399	0.2016
2400	0.1936	2401	0.1858	2402	0.1782
2403	0.1708	2404	0.1638	2405	0.1570
2406	0.1504	2407	0.1439	2408	0.1376
2409	0.1314	2410	0.1254	2411	0.1197
2412	0.1143	2413	0.1090	2414	0.1040
2415	0.0991	2416	0.0943	2417	0.0898
2418	0.0854	2419	0.0813	2420	0.0772
2421	0.0733	2422	0.0695	2423	0.0659
2424	0.0623	2425	0.0590	2426	0.0558
2427	0.0528	2428	0.0499	2429	0.0471
2430	0.0444	2431	0.0419	2432	0.0394
2433	0.0371	2434	0.0350	2435	0.0329
2436	0.0310	2437	0.0291	2438	0.0274
2439	0.0256	2440	0.0240	2441	0.0224
2442	0.0209	2443	0.0195	2444	0.0182
2445	0.0169	2446	0.0158	2447	0.0146

Table B.11: Convolution After Smoothing

$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$	$\mathbf{z}$	$\hat{\mathbf{p}}_{\mathbf{Z}}(\mathbf{z})$
($Z = A + B + C + D$, represented in 0.0001's)					
2448	0.0136	2449	0.0126	2450	0.0116
2451	0.0108	2452	0.0100	2453	0.0092
2454	0.0085	2455	0.0079	2456	0.0073
2457	0.0067	2458	0.0062	2459	0.0058
2460	0.0053	2461	0.0050	2462	0.0046
2463	0.0044	2464	0.0041	2465	0.0039
2466	0.0037	2467	0.0035	2468	0.0033
2469	0.0031	2470	0.0029	2471	0.0028
2472	0.0026	2473	0.0025	2474	0.0023
2475	0.0022	2476	0.0020	2477	0.0019
2478	0.0018	2479	0.0017	2480	0.0015
2481	0.0014	2482	0.0013	2483	0.0012
2484	0.0011	2485	0.0010	2486	0.0009
2487	0.0008	2488	0.0007	2489	0.0006
2490	0.0006	2491	0.0005	2492	0.0005
2493	0.0004	2494	0.0004	2495	0.0003
2496	0.0003	2497	0.0003	2498	0.0003
2499	0.0002	2500	0.0002	2501	0.0002
2502	0.0002	2503	0.0002	2504	0.0002
2505	0.0002	2506	0.0002	2507	0.0002
2508	0.0001	2509	0.0001	2510	0.0001
2511	0.0001	2512	0.0001	2513	0.0001
2514	0.0001	2515	0.0001	2516	0.0001
2517	0.0001	2518	0.0001	2519	0.0001

References

[1] Joseph Abate, Gagan L. Choudhury, and Ward Whitt. An introduction to numerical transform inversion and its application to probability models. In W. Grassman, editor, *Computational Probability*. Kluwer Academic Publishers, Boston, 2000.

[2] Johann Blieberger. Data-flow frameworks for worst-case execution time analysis. *Real-Time Syst.*, 22(3):183–227, 2002.

[3] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. Source-level execution time estimation of c programs. In *Proceedings of the ninth international symposium on Hardware/software codesign*, pages 98–103. ACM Press, 2001.

[4] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. PWS Publishing Company, Boston, 5th edition, 1993.

[5] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, 1998.

[6] Paul L. DeVries. *A First Course in Computational Physics*. John Wiley and Sons, New York, 1994.

[7] S. Edgar and A. Burns. Statistical analysis of WCET for scheduling. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE, 3-6 Dec. 2001*, pages 215–224. IEEE Computer Society, 2001.

[8] A. W. F. Edwards. *Likelihood*. The Johns Hopkins University Press, Baltimore, 1992.

[9] Andreas Ermedahl, Friedhelm Stappert, and Jakob Engblom. Clustered calculation of worst-case execution times. In *Proceedings of the international conference*

on Compilers, architectures and synthesis for embedded systems, pages 51–62. ACM Press, 2003.

[10] R. Ernst and W. Ye. Embedded program timing analysis based on path clustering and architecture classification. In *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 598–604. IEEE Computer Society, 1997.

[11] N. Gehani and K. Ramamritham. Real-time concurrent c: a language for programming dynamic real-time systems. *The Journal of Real-Time Systems*, 1(3):377–405, 1991.

[12] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, 3rd edition, 2001.

[13] A. Hergenhan and W. Rosenstiel. Static timing analysis of embedded software on advanced processor architectures. In *Proceedings of the conference on Design, automation and test in Europe*, pages 552–559. ACM Press, 2000.

[14] Francis B. Hildebrand. *Introduction to Numerical Analysis*. Dover Publications, New York, 2nd edition, 1987.

[15] Robert V. Hogg and Allen T. Craig. *Introduction to Mathematical Statistics*. Prentice-Hall, Englewood Cliffs, NJ, 5th edition, 1995.

[16] P. Holgate. The lognormal characteristic function. *Commun. Statist.-Theory Meth.*, 18(12):4539–4548, 1989.

[17] Stuart A. Klugman, Harry H. Panjer, and Gordon E. Willmot. *Loss Models - From Data to Decisions*. John Wiley & Sons, Inc., New York, 1998.

[18] Roy P. Leipnik. On lognormal random variables: I – the characteristic function. *J. Austral. Math. Soc. Ser. B*, 32(1):327–347, 1991.

[19] Yau-Tsun Steven Li, Sharad Malik, and Andrew Wolfe. Performance estimation of embedded software with instruction cache modeling. *ACM Trans. Des. Autom. Electron. Syst.*, 4(3):257–279, 1999.

[20] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, Inc., New York, 1997.

[21] Geoffrey J. McLachlan and David Peel. *Finite Mixture Models*. John Wiley & Sons, Inc., New York, 2000.

[22] A. K. Mok, P. Amerasinghe, M. Chen, and K. Tantisirivat. Evaluating tight execution time bounds of programs by annotations. In *Proceedings of the IEEE Workshop on Real-Time Operating Systems and Software*, pages 74–80. IEEE Computer Society, 1989.

[23] Kelvin D. Nilsen and Bernt Rygg. Worst-case execution time analysis on modern processors. In *Proceedings of the ACM SIGPLAN 1995 workshop on Languages, compilers, & tools for real-time systems*, pages 20–30. ACM Press, 1995.

[24] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[25] B. W. Silverman. *Density Smoothing for Statistics and Data Analysis*. Number 26 in Monographs on Statistics and Applied Probability. Chapman & Hall / CRC, Boca Raton, FL, 1986.

[26] D. S. Sivia. *Data Analysis - A Bayesian Tutorial*. Oxford University Press, New York, 1996.

[27] Alan Stuart and J. Keith Ord. *Kendall's Advanced Theory of Statistics - Volume 1 - Distribution Theory*. Oxford University Press, New York, 6th edition, 1994.

[28] Alan Stuart and J. Keith Ord. *Kendall's Advanced Theory of Statistics - Volume 2a - Classical Inference and the Linear Model*. Oxford University Press, New York, 6th edition, 1999.

[29] George B. Thomas and Ross L. Finney. *Calculus and Analytic Geometry*. Addison-Wesley, Reading, MS, 8th edition, 1992.

[30] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Number 60 in Monographs on Statistics and Applied Probability. Chapman & Hall / CRC, Boca Raton, FL, 1995.

[31] David V. Wedder. *Advanced Calculus*. Dover Publications, New York, 2nd edition, 1989.

Vita

Kelly P. Leahy

Date of Birth September 19, 1976

Place of Birth St. Louis, MO U.S.A

Degrees B.S. Cum Laude, Actuarial Science, May 1994

May 2005