A probability programming language: Development and applications

Andrew Gordon Glen
College of William & Mary - Arts & Sciences

A Probability Programming Language: Development and Applications

A Dissertation

Presented to

The Faculty of the Department of Applied Science

The College of William & Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

Andrew G. Glen

1998

APPROVAL SHEET

This Dissertation is submitted in partial fulfillment of

the requirements for the Degree of

Doctor of Philosophy

[illegible]

Andrew G. Glen

APPROVED, January 1998

Lawrence Leemis

Lawrence M. Leemis,
Dissertation Advisor

[illegible]

John H. Drew

[illegible]

Sidney H. Lawrence

[illegible]

Rex K. Kincaid

Donald R Barr

Donald R. Barr,
Outside Examiner

Contents

Acknowledgement

I must gratefully acknowledge the assistance of many people who have helped in very many ways in the production of this dissertation and research. Professors Hank Krieger, Marina Kondratovitch gave help in specific parts of this research. A special note of personal thanks goes to Professors Donald Barr, John Drew, and especially Larry Leemis, who were tremendously patient, insightful, and supportive of the research. I thank for their patience and support my wife, Lisa Glen, and my children Andrea, Rebecca, and Mary. I also thank and praise almighty God, for it is only by His divine will that we are privileged to understand what we have learned in this process.

List of Figures

List of Tables

Abstract

A probability programming language is developed and presented: applications illustrate its use. Algorithms and generalized theorems used in probability are encapsulated into a programming environment with the computer algebra system Maple to provide the applied community with automated probability capabilities. Algorithms of procedures are presented and explained, including detailed presentations on three of the most significant procedures. Applications that encompass a wide range of applied topics including goodness-of-fit testing, probabilistic modeling. central limit theorem augmentation. generation of mathematical resources, and estimation are presented.

A Probability Programming Language:

Development and Applications

Chapter 1

Introduction

1.1 General

Probability theory, as it exists today, is a vast collection of axioms and theorems that, in essence, provides the scientific community many contributions, including:

- the naming and description of random variables that occur frequently in applications,
- the theoretical results associated with these random variables, and,
- the applied results associated with these random variables for statistical applications.

No one volume categorizes its work in exactly these three ways, but the literature's comprehensive works accomplish these goals. Whether voluminous, such as the work of Johnson, Kotz, and Balakrishnan (1995), or succinct, such as that of Evans, Hastings, and Peacock (1993), one finds all three of these areas presented in chapters that are organized on the first contribution above, naming and describing the random variables. Works such as Hogg and Craig (1995), Port (1994), and David (1981) organize their efforts according to the second contribution, covering theoretical results that apply to random variables. Then there are the works such as Law and Kelton (1991), Lehmann (1986), and D'Agostino and Stephens (1986) who concentrate on the statistical applications of random variables, and tailor their explanations of probability theory to the portions of the field that have application in statistical analysis.

In all these works, as well as countless others, one stark omission is apparent. There is no mention of an ability to automate the naming, processing, or application of random variables. This omission is even more profound when one considers the tedious nature of the mathematics involved in the execution of many of these results for all but the simplest of examples. In practice, the level of tedium makes the actual execution untenable for many random variables. Automation of certain types of these procedures could eradicate this tedium. There is an abundance of statistical software packages that give the scientific community powerful tools to apply statistical procedures. But to date, there is no package that attempts to automate the more theoretical side of the probabilist's work. Even the simplest of tasks, plotting a fully specified probability density function, is not provided in many statistical packages. In order to plot new, ad-hoc densities or CDFs, one is often required to write an appropriate program.

A conceptual probability software package is now presented that begins to fill this existing gap. Before outlining this work's approach, let us present some examples to illustrate what should be included in such a probability software package.

Consider the following independent random variables: $W \sim$ gamma(λ, κ), $Z \sim$ N(μ, σ), $Y \sim$ Weibull(λ, κ), X and $R \sim$ arctan(ϕ, α), and D, T, U, and V as specified in the questions below. See Appendix A for more information on the arctan distribution.

- What is the distribution of $V = W + X + Y$?
- What is the distribution of $T = X \cdot \ln(W^2) + e^{YZ}$?
- What is the distribution of a random distance D, which is the sum of the product of random rates $R_1, R_2, \ldots, R_n$ and random times $T_1, T_2, \ldots, T_n$, i.e., $D = R_1 \cdot T_1 + R_2 \cdot T_2 + \cdots + R_n \cdot T_n$?
- What is the distribution of the system lifetime U in a reliability block diagram containing two parallel blocks of two subsystems that consist of two components in series, i.e., $U = \max\{\min\{V, W\}, \min\{Y, Z\}\}$?
- What is the *exact* upper tail probability for the statistic 6.124 associated with the distribution of the 4th order statistic out of a sample of 12 iid observations that have the same distribution as U?
- How might one use the previous result to improve goodness-of-fit testing?

- How could one employ the PDFs of order statistics of a population, instead of the PDF of the population itself, to develop an alternate approach to maximum likelihood estimation?
- What is the distribution of the maximum likelihood estimator of the inverse Gaussian random variable's first parameter μ when a sample size of n is specified?
- Most importantly, if one could find these answers, what is their utility to the statistical and applied science community?

There is no implication that the previously cited authors are remiss in neglecting the automation of probability software. In fact it is only with the advent and maturing of computer algebra systems, such as Maple and Mathematica, that there now exists the ability to automate probabilistic modeling and research. This doctoral research and dissertation will take advantage of this relatively new technology by developing and presenting a software "engine" that contributes to the fields of probabilistic modeling and statistical applications. This research has concentrated on procedures in the symbolic language Maple V.

The specific contributions to the applied probability and statistics community of this research and dissertation include the following:

1. Detailed algorithms that comprise the conceptual software.
2. Generalized versions of theorems that comprise the software. [Note that while the theorems themselves may exist in more general forms, their general forms

appear difficult to implement in an automated environment.]

3. An algorithm that produces the exact CDF of the Kolmogorov–Smirnov test statistic.

4. Detailed explanations and examples of the software's capabilities.

5. Application examples that contribute, on their own, to various areas within probability and statistics.

6. Explorative examples of probabilistic quests that appear to be difficult to carry out without automation.

7. Extensions of testing statistical hypotheses, to include specific contributions in the areas of outlier detection, goodness-of-fit, and parameter estimation.

8. Demonstrations in the general area of probabilistic model design, to include specific contributions in the areas of survival distributions, reliability block diagrams, exact solutions to central limit theorem (CLT) applications, and estimation.

1.2 Literature review

While the current literature will be reviewed throughout the dissertation, there are a number of works that should be mentioned for their general applicability to this research. These include the works of Johnson, Kotz, and Balakrishnan (1995), Leemis (1995), Port (1984), Rohatgi (1976), and others that provide the foundation for the theory behind the algorithms and implementation. The review of the literature has discovered no publication on implementing probabilistic procedures in computer algebra languages, nor on the benefits of such a paradigm. Databases searched include FirstSearch, INNOPAC, INSPEC, DTIC, NTIC, Science Citation Index, Library of Congress, Swem Library at The College of William & Mary, and the USMA library at West Point, NY. Search strings included the following individual subject areas and pairs of subject areas, where appropriate: distributions, goodness-of-fit, life testing, Maple, modeling, order statistics, probability, reliability, and symbolic algebra. While there are many listings under these terms and pairings of these terms, no work was found about combining probabilistic results with computer algebra implementation. The negative result of this search indicates that there is a lack of archival material on the subject.

1.3 Outline of the dissertation

This dissertation is presented according to the following outline. In Chapter 2 the development, abilities, and examples of use of the software language are presented. Chapter 3 contains the development of the procedure that accommodates transformations of random variables to include a re-stated, general, implementable theorem for such work. In Chapter 4 a procedure for finding the distribution of the product of two independent continuous random variables is presented. In Chapter 5 a procedure that returns the distribution of the Kolmogorov–Smirnov goodness of fit statistic, given a specified sample size, is presented. Chapter 6 contains an application in which a new goodness-of-fit test procedure is presented and tested using the software. Chapter 7 is a collection of examples of explorations in the fields of probability and statistics that are now possible due to the software. Finally, in Chapter 8, conclusions and suggestions of further work are given. In the appendices are listed the algorithms for the software, as well as documentation of the early work in creating new probabilistic models.

1.4 Notation and nomenclature

This section reviews certain notation and nomenclature used here. Use is made of the following acronyms and functional notation for density representations:

- probability density function (PDF) $f_X(x)$,
- cumulative distribution function (CDF) $F_X(x) = \int_{-\infty}^{x} f_X(s)\,ds$,
- survivor function (SF) $S_X(x) = 1 - F_X(x)$,
- hazard function (HF) $h_X(x) = \frac{f_X(x)}{S_X(x)}$,
- cumulative hazard function (CHF) $H_X(x) = \int_{-\infty}^{x} h_X(s)\,ds$, and
- inverse distribution function (IDF) $F_X^{-1}(x)$.

Throughout the dissertation, the proposed software is referred to as "a probability programming language" (APPL) for brevity. The terms "piecewise" and "segmented" are used to refer to PDFs (and other functions) that can be constructed by piecing together various standard functions, such as polynomials, logarithms, exponentials, and trigonometric functions, e.g., the triangular(1, 2, 3) distribution which has two segments or two pieces, each of which are linear functions. The common abbreviation "N(μ, σ)" is used to refer to the normal distribution. Note that the second parameter is the standard deviation, not the variance. Also, "U(a, b)" is used to represent the uniform distribution with parameters a and b. Subscripts in parentheses represent order statistics, e.g. the r^{th} order statistic associated with a random sample $X_1, X_2, \ldots, X_n$ is denoted by $X_{(r)}$. The abbreviation "iid" is used to denote independent and identically distributed random variables. The terms "fully-specified," "semi-specified," and "unspecified" are used to describe the degree to which parameters are specified as constants or fixed parameters in a distribution. For example, the exponential(1) distribution is a fully specified distribution. The Weibull(1, κ) and the N(0, σ) are both semi-specified distributions. The triangular(a, b, c) and exponential(λ) distributions are both unspecified. Typewriter font is used to represent Maple language statements. For example "`> X := UniformRV(0, 1);`" is a Maple assignment statement. Note that the symbol "`>`" represents the Maple input prompt and is not typed.

Chapter 2

Software Development

The notion of probability software is different from the notion of applied statistical software. Probability theory is rife with theorems and calculations that require symbolic, algebraic manipulations. Applied statistical calculations are usually numeric manipulations of data based on known formulas associated with distributions of common random variables. This section contains a discussion on several algorithms that contribute to the development of APPL. Availability of computer algebra systems such as Maple and Mathmatica facilitate the development of software that will derive functions, as opposed to computing numbers.

Probability software must, at the most basic level, be a means of producing distributions of random variables. At the heart of the software must reside an "engine" that can compute new, useful representations of distributions.

The derivation of exact distribution functions of complex random variables is often untenable. In such cases, one had to be content with approximations and summary statistics of the unknown distributions. regardless of whether those approximations and summaries were adequate. For example. one often approximates distributions using Monte Carlo simulation or by invoking the central limit theorem. Statistics such as the sample mean and variance of the approximated distribution are then reported. If what was really needed was a certain percentile of the approximated distribution. often times the entire simulation would need to be remodeled. re-validated. re-verified. and re-run to obtain the needed information. A result such as a fully-specified PDF would eradicate the need for such redundant efforts. One also would have analytical results to represent characteristics of certain complex random variables whose fully-specified functions are untenable. For example, renewal theory and compound Poisson process theory have results that derive the mean and variance of complex distributions. but fall short of actually determining the entire representation of complex distributions via a PDF. CDF. or some other form of the distribution. The proposed probabilistic software is designed to make a breakthrough into the area of completely describing complex distributions with PDFs. CDFs. and the like. thereby providing increased modeling capability to the analyst.

At the most general level. one could attempt to find distributions of intricate transformations of multivariate. dependent random variables. The software described here is limited to univariate. continuous. independent random variables. and the complex transformations and combinations that can result between independent random variables. A set of algorithms that derives functional representations of distributions and uses these functions in a manner that is typically needed in applications is presented. Specifically, algorithms have been developed that will conduct the following operations:

- supply a common data structure for the distributions of continuous, univariate, random variables—including distribution functions that may be defined piecewise, e.g. the triangular distribution,
- convert any functional representation of a random variable into another functional representation using the common data structure, i.e. allowing conversion amongst the PDF, CDF, SF, HF, CHF, and IDF,
- verify that the area under a computed PDF is one,
- provide straightforward instantiation of well-known distributions, such as the exponential, normal, uniform, and Weibull distributions, with either numeric or symbolic parameters,
- determine the distribution of a simple transformation of a continuous random variable, $Y = g(X)$—including piecewise, continuous transformations,
- determine common summary characteristics of random variables, such as the mean, variance, other moments, and so forth,
- calculate the PDF of sums of independent random variables, i.e. $Y = X + Z$,
- calculate the PDF of products of independent random variables, i.e. $Y = XZ$,
- calculate the PDF of the minimum and maximum of independent random variables, i.e. $Y = \min\{X, Z\}$ and $Y = \max\{X, Z\}$.

- calculate the PDF of the r^{th} order statistic from a sample of n iid random variables,
- calculate probabilities associated with a random variable,
- generate random variates associated with a random variable,
- plot any of the six functional forms of any distribution, e.g. the HF or CDF,
- provide basic statistical abilities, such as maximum likelihood estimation, for distributions defined on a single segment of support,
- compliment the structured programming language that hosts the software (in this case Maple) so that all of the above mentioned procedures may be used in mathematical and computer programming in that language.

2.1 The common data structure

Implicit in a probability software language is a common, succinct, intuitive, and manipulatable data structure for describing the distribution of a random variable. This implies there should be one data structure that applies to the CDF, PDF, SF, HF, CHF, and IDF. The common data structure used in this software is referred to as the "list-of-lists." Specifically, any functional representation of a random variable is presented in a list that contains three sub-lists, each with a specific purpose. The first sub-list contains the ordered functions that define the segments of the distribution. The PDF representation of the triangular distribution, for example, would have the two linear functions that comprise the two segments of its PDF for its first sub-list. Likewise, the CDF representation of the triangular distribution would have the two quadratic functions that comprise the two segments of its CDF for its first sub-list. The second sub-list is an ordered list of real numbers that delineate the end points of the segments for the functions in the first sub-list. The end point of each segment is automatically the start point of the succeeding segment. The third sub-list indicates what distribution form the functions in the first sub-list represent. The first element of the third sub-list is either the string `Continuous` for continuous distributions or `Discrete` for discrete distributions. The second element of the third sub-list shows which of the 6 functional forms is used in the first sub-list. The string `PDF`, for example, indicates the list-of-lists is currently a PDF list-of-lists. Likewise, `CDF` indicates that a CDF is being represented.

Examples:

- The following Maple statement assigns the variable `X` to a list-of-lists that represents the PDF of a U(0, 1) random variable:

  ```
  > X := [[x -> 1], [0, 1], ['Continuous', 'PDF']];
  ```

- The triangular distribution has a PDF with two pieces to its distribution. The following statement defines a triangular(0, 1, 2) random variable `X` as a list-of-lists:

  ```
  > X := [[x -> x, x -> 2 - x], [0, 1, 2], ['Continuous', 'PDF']];
  ```

- An exponential random variable `X` with a mean of 2 can be defined in terms of its hazard function with the statement:

```
> X := [[x -> 1 / 2], [0, infinity], ['Continuous', 'HF']];
```

- Unspecified parameters can be represented symbolically. A N(θ, 1) random variable X can be defined with the statement:

```
> X := [[x -> exp(-(x - theta) ^ 2) / sqrt(2 * Pi)],
       [-infinity, infinity], ['Continuous', 'PDF']];
```

- The parameter space can be specified by using the Maple **assume** function. Consider the random variable T with HF

$$h_T(t) = \begin{cases} \lambda & 0 < t < 1 \\ \lambda t & t \geq 1 \end{cases}$$

for $\lambda > 0$. The random variable T can be defined by the statements:

```
> assume(lambda > 0);
> T := [[t -> lambda, t -> lambda * t], [0, 1, infinity],
       ['Continuous', 'HF']];
```

- The syntax allows for the endpoints of the segments associated with the support of the random variable to be specified symbolically. A U(a, b) random variable X is defined by:

```
> X := [[x -> 1 / (b - a)], [a, b], ['Continuous', 'PDF']];
```

- No error checking is performed when a distribution is defined. This means that the statement shown below will create a list-of-three lists that is not a legitimate

PDF.

```
> X := [[x -> 6], [0, 5], ['Continuous', 'PDF']];
```

Some error checking will be performed by the procedure `VerifyPDF`, which is presented in a subsequent section.

2.2 Common continuous, univariate distributions

Syntax: The command

> `> X :=` *RandomVariableName*`RV(`*ParameterSequence*`);`

assigns to the variable `X` a list-of-lists representation of the specified random variable. The arguments in *ParameterSequence* may be real, integer, or string (for symbolic parameters).

Purpose: Included in the prototype software is the ability to instantiate common distributions. While the list-of-lists is a functional form that lends itself to the mathematics of the software, it is not an instantly recognizable form for representing a distribution. Here is provided a number of simple procedures that take relatively common definitions of distributions and convert them to a list-of-lists format. The included distributions are well-know ones, such as the normal, Weibull, exponential, and gamma. A complete list of the distributions provided in APPL, to include their parameters, is presented in Appendix B.

Special Issues: The suffix `RV` is added to each name to make it readily identifiable as a distribution assignment, as well as to deconflict Maple specific names such as

normal and **gamma**. The first letter of each word is capitalized, which is the case for all procedures in APPL. Also, there is no space between words in a procedure call, e.g., an inverse Gaussian random variable may be defined by the command `InverseGaussianRV(`*Parameter1, Parameter2*`)`. Usually the format is returned as a PDF, but in the case of the IDB distribution, a CDF is returned. The CDF of the IDB distribution, it turns out, is easier for Maple to manipulate (e.g., integrate, differentiate) than the PDF. Certain assumptions are made about unspecified parameters. For example, an assignment of an unspecified exponential random variable (see the second example below), will result in the assumption that $\lambda > 0$. This assumption, as with all other distributions' assumptions, are only applied to unspecified parameters. The assumptions allow Maple to carry out certain types of symbolic integration, such as verifying the area under the density is in fact one, for a PDF (see Section 2.4).

Examples:

- The exponential(1) distribution may be created with the following statement:

  ```
  > X := ExponentialRV(1);
  ```

- The exponential(λ) random variable X, where $\lambda > 0$, distribution may be created as follows:

  ```
  > X := ExponentialRV(lambda);
  ```

- These procedure also allow a modeler to reparameterize a distribution. The exponential ($\frac{1}{\theta}$) distribution where $\theta > 0$, for example, may be created as follows:

  ```
  > assume(theta > 0);
  > X := ExponentialRV(1 / theta);
  ```

- The semi-specified Weibull(λ, 1), where $\lambda > 0$, distribution may be created as follows:

  ```
  > X := WeibullRV(lambda, 1);
  ```

  Note that this is a special case where the Weibull distribution is equivalent to an exponential distribution.

- The standard normal distribution may be created as follows:

  ```
  > X := NormalRV(0, 1);
  ```

All distributions presently included in APPL and their parameterizations are listed in Appendix B.

2.3 The six representations of distributions

Syntax: The command

DesiredForm(*RandomVariable* [, *Statistic*]);

returns the list-of-lists format of the desired functional representation of the distribution, where *DesiredForm* is one of the following: `PDF`, `CDF`, `SF`, `HF`, `CHF`, or `IDF`. The single argument *RandomVariable* must be in the list-of-lists format. The optional argument , *Statistic* may be a constant or a string.

Purpose: The 6 × 6 distribution conversion ability, a variation of the matrix outlined by Leemis (1995, p. 55), is provided so that the functional form of a distribution can be converted to and from its six well-known forms, the PDF, CDF, SF, IDF, HF, and

CHF. This set of procedures will take one form of the distribution as an argument and return the desired form of the distribution in the appropriate list-of-lists format. For the one-parameter call, the functional representation will be returned. For the two-parameter call, the actual value of the function at that point will be returned.

Special Issues: The procedures are fairly robust against non-specified parameters for the distributions that will be converted (see the fourth example below).

Examples:

- To obtain the CDF form of a standard normal random variable:

  ```
  > X := NormalRV(0, 1);
  > X := CDF(X);
  ```

  or, equivalently, in a single line,

  ```
  > X := CDF(NormalRV(0, 1));
  ```

  Since the CDF for a standard normal random variable is not closed form, APPL returns the following:

  $$X := [[x \to \frac{1}{2}\operatorname{erf}(\frac{1}{2}x\sqrt{2}) + \frac{1}{2}], [-\infty, \infty], [\mathit{Continuous}, \mathit{CDF}]]$$

- If $X \sim \mathrm{N}(0, 1)$, then the following statements can be used to find $P(X < 1.96) \cong 0.975$.

  ```
  > X := NormalRV(0, 1);
  > prob := CDF(X, 1.96);
  ```

- Should the hazard function of an exponential distribution be entered, its associated PDF may be determined as follows:

```
> X := [[x -> 1], [0, infinity], ['Continuous', 'HF']];
> X := PDF(X);
```

- For the case of unspecified parameters, the following statements convert an unspecified Weibull PDF to an unspecified Weibull SF:

```
> X := WeibullRV(lambda, kappa);
> X := SF(X);
```

  which returns:

$$X := [[x \rightarrow e^{(-x^{\kappa^{\sim}}\lambda^{\sim\kappa^{\sim}})}], [0, \infty], [Continuous, SF]]$$

  Note that the tildes after the parameters indicate that assumptions have been made concerning the parameters (i.e., $\lambda > 0$ and $\kappa > 0$) in the WeibullRV procedure.

- Finding a quantile of a distribution requires the IDF procedure. If $X \sim$ Weibull(1, 2), then the 0.975 quantile of the distribution can be found with the statement

```
> quant := IDF(WeibullRV(1, 2), 0.975);
```

- The procedures can be nested so that if the random variable X has been defined in terms of its PDF, then the statement

```
> X := PDF(CDF(HF(SF(CHF(IDF(X))))));
```

does nothing to the list-of-lists representation for `X`, assuming that all transformations can be performed analytically.

Algorithm: The conversions are shown in a 6×6 matrix in Appendix C. Each element of the matrix takes the 'row' and converts it to the type specified in the 'column' heading. Thus the first row, second element of the matrix shows a call to the `CDF` procedure using the PDF representation of a random variable as an argument which returns the CDF representation of a random variable.

2.4 VerifyPDF

Syntax: The command

`VerifyPDF(`*RandomVariable*`);`

returns true or false, depending on whether or not the PDF integrates to one. The single argument *RandomVariable* must be in the list-of-lists format described previously. In addition, the procedure prints

'The area under the PDF is ',

along with the area, and "true" if the area is 1.0 or "false" if the area is not 1.0.

Purpose: The purpose of this procedure is to help determine if a random variable in the list-of-lists format is in fact a viable representation of a continuous distribution. Specifically, the procedure converts the distribution to the PDF form and carries out the definite integration of the PDF to see if the area under the PDF is 1. If so, it displays

The area under the PDF is . 1

and returns "true": otherwise it returns the computed area and the string "false" if the area is more than 0.0000001 away from 1. This procedure is primarily an indicator tool to check if the list-of-lists format of a random variable has been input correctly.

Special issues: The procedure only integrates the area under each segment of the PDF of the argument *RandomVariable*. It does not check for negative functional values of $f(x)$. The third example below shows the continuous function

$$f(x) = 3\,|x| - 1 \qquad -1 < x < 1$$

integrates to one, yet is not a PDF since $f(0) = -1$.

For many well-known distributions, the procedure will carry out the symbolic integration and verify that the area under the PDF is one, as illustrated in the second example below. Not all of the distributions described in Section 2.2 have this symbolic capability, but most do. For example, the unspecified log normal distribution will integrate to one, but the unspecified inverse Gaussian distribution will not; see the fourth example below.

Examples:

- The following Maple statements create an exponential random variable `X` with a mean of 1, verify that the area under $f(x)$ is one, and return *true* from `VerifyPDF`:

```
> X := ExponentialRV(1);
> VerifyPDF(X);
```

- Since assumptions are made internally in ExponentialRV about the parameter space, the following two statements will also return *true*:

  ```
  > X := ExponentialRV(lambda);
  > VerifyPDF(X);
  ```

- The following code defines a function $f(x)$ such that $\int_{-1}^{1} f(x)dx = 1$ and $f(0) = -1$, so that VerifyPDF returns *true* even though this is not a legitimate PDF:

  ```
  > X := [[x -> 3 * abs(x) - 1], [-1, 1], ['Continuous', 'PDF']];
  > VerifyPDF(X);
  ```

- Maple is not able to conduct the integration for more complex distributions. In this example, X is assigned the unspecified inverse Gaussian distribution, and an attempt to integrate the area under the density is unsuccessful.

  ```
  > X := InverseGaussianRV(p1, p2);
  > VerifyPDF(X);
  ```

  These statements return an error message indicating that the function does not evaluate to numeric. The assumption is made that future releases of Maple will be able to correctly integrate this PDF.

Algorithm: The algorithm first checks to see whether the distribution of interest is continuous. Next, it checks to see if the distribution of the random variable is represented by a PDF. If not, it converts a local distribution to a PDF form using the PDF procedure, which was introduced in Section 2.3. At this point, the area under the PDF is calculated and printed. The returned value from the procedure is "*true*" if the area is within 0.0000001 of 1. and "*false*" otherwise. The algorithm is given in Appendix D.

2.5 MedianRV

Syntax: The command

MedianRV(*Random Variable*);

returns the median of a specified distribution.

Purpose: This procedure returns the median of a random variable.

Special Issues: It is fairly robust for use with distributions that have unspecified parameters.

Examples:

- For the fully-specified Weibull distribution, the following statements will assign the median of the distribution to the variable m.

```
> X := WeibullRV(1, 2);
> m := MedianRV(X);
```

- The following statements determine the median of an exponential random variable with unspecified parameters:

```
> X := ExponentialRV(lambda);
> m := MedianRV(X);
```

  which results with the value $\frac{\ln(2)}{\lambda}$.

Algorithm: The algorithm is a special case of the two-parameter `IDF` procedure call, where the second parameter is $\frac{1}{2}$.

2.6 DisplayRV

Syntax: The command

`DisplayRV(`*RandomVariable*`);`

displays the list-of-lists format of the distribution in standard mathematical notation, using the Maple `piecewise` procedure.

Purpose: The purpose of this procedure is to make the list-of-lists representation of a distribution more readable. A long list-of-lists with several segments is not easy to understand. This procedure converts a list-of-lists formatted distribution into the Maple-syntaxed "piecewise" function. Such versions of segmented functions are displayed in a more readable manner in Maple. It also states whether the current representation is a PDF, CDF, etc. There is no computation in this procedure. The procedure attempts to make the list-of-lists format more readable.

Special Issues: None.

Example:

- The piecewise triangular distribution could be displayed as follows:

  ```
  > DisplayRV(TriangularRV(1, 2, 3));
  ```

  which displays the following on a Maple worksheet:

This random variable is currently represented as follows:

'Continuous', 'PDF'

$$\begin{cases} 0 & x < 1 \\ x - 1 & x < 2 \\ 3 - x & x < 3 \end{cases}$$

Algorithm: This algorithm is a set of commands that creates a sequence of conditions and functions in a manner that is usable by the `piecewise` command.

2.7 PlotDist

Syntax: The command

`PlotDist(`*RandomVariable, LowerLimit, UpperLimit*`);`

plots the current list-of-lists defined distribution between *LowerLimit* and *UpperLimit* on a coordinate axis.

Purpose: To give a graphical representation of any list-of-lists represented distribution. The arguments *LowerLimit* and *UpperLimit* define the minimum and maximum values desired on the horizontal axis.

Special Issues: A distribution function must be fully-specified for a plot to be generated. The procedure is especially useful for plotting distributions that have more than one segment.

Examples:

- The following statements will generate the plot of the PDF for the triangular(1, 2, 3) distribution:

```
> X := TriangularRV(1, 2, 3);
> PlotDist(X, 1, 3);
```

- To plot the HF of the exponential(1) distribution for $0 < t < 10$, enter the statements:

```
> X := ExponentialRV(1);
> PlotDist(HF(X), 0, 10);
```

- To see a progression of the five PDFs of the order statistics (the procedure is introduced in Section 2.10) for an exponential(1) distribution, one could enter the following statements:

```
> X := ExponentialRV(1);
> n := 5;
> For i from 1 to n do
    PlotDist(OrderStat(X, n, i), 0, 10);
  od;
```

  The result is five PDFs plotted sequentially. This sequence could be of use to an instructor explaining the progressive nature of order statistics to first-year probability students.

- Unspecified distributions produce "empty plot" warnings:

```
> X := ExponentialRV(lambda);
> PlotDist(X, 0, 10);
```

Algorithm: The algorithm is a nested set of `Plot` commands that combine to form a single plot. This is standard Maple programming for plotting multiple functions on a single set of axes. Since $-\infty$ and ∞ are common endpoints of random variables, it is necessary to specify the lower and upper endpoints of the horizontal axis.

2.8 ExpectationRV

Syntax: The command

ExpectationRV(*Random Variable*, *Function*);

returns the expected value of a function of a random variable.

Purpose: To find the expected value of a function of a random variable.

Special Issues: Procedures `MeanRV` and `VarianceRV` are the special cases of the `ExpectationRV` procedure, evident by their names.

Examples:

- In order to find the expected value of a standard normal random variable, type:

```
> X := NormalRV(0, 1);
> meanX := ExpectationRV(X, x -> x);
```

- Unspecified distributions may also be used. Here is the mean of the exponential(λ) random variable is calculated with the statements:

```
> X := ExponentialRV(lambda);
> meanX := ExpectationRV(X, x -> x);
```

Algorithm: The algorithm is a straightforward implementation of the following result. Let the continuous random variable X have PDF $f_X(x)$. Let $g(X)$ be a continuous function of the X. The expected value of $g(X)$, when it exists, is given by

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) \cdot f_X(x) dx\,.$$

The algorithm is in Appendix D.

2.9 Transform

Syntax: The command

```
Transform(RandomVariable, Transformation);
```

returns the PDF of the transformed random variable in the list-of-lists format.

Purpose: To determine the PDF of the transformation of a random variable of the form $Y = g(X)$. As is the case for the random variable X, the transformation function $g(X)$ may be defined in a piecewise fashion (see chapter 3).

Special Issues: The transformation function must also be defined in an altered list-of-lists format. For this function, the modeler must break the transformation into piecewise monotone segments. Details on why this must be the case, in addition to other implementation issues are given in Chapter 3.

Examples:

- Let $X \sim$ U(0, 1) and $Y = g(X) = 4X$. The following statements will generate the PDF of Y:

```
> X := [[x -> 1], [0, 1], ['Continuous', 'PDF']];
> g := [[x -> 4 * x], [-infinity, infinity]];
> Y := Transform(X, g);
```

- The following statements determine the distribution of the square of an inverse Gaussian random variable with $\lambda = 1$ and $\mu = 2$:

```
> X := InverseGaussianRV(1, 2);
> g := [[x -> x ^ 2], [0, infinity]];
> Y := Transform(X, g);
```

- An example of finding the negative of a random variable is included in Section 2.12 on the command SumRV, used in finding differences of random variables.

- An example of finding the reciprocal of a random variable is included in Section 2.11 on the command ProductRV, used in finding ratios of random variables.

- An example of dividing a random variable by a constant is included in Section 2.15 on the command MLE, used to the find distributions of certain estimators.

- A number of other illustrative examples are given in Chapter 3.

Algorithm: The theorem which provides the basis for the algorithm and the details associated with the algorithm are found in Chapter 3. The algorithm is in Appendix E.

2.10 OrderStat

Syntax: The command

`OrderStat(`*RandomVariable*`, `*n*`, `*r*`);`

returns the PDF of the r^{th} of n order statistics drawn from a population having the same distribution as *RandomVariable*.

Purpose: This procedure is designed to return the marginal distribution of specified order statistics. The procedures arguments are defined as follows: the population distribution is represented by the list-of-lists format, the integer sample size n, and the integer r to denote the r^{th} order statistic. The procedure returns the marginal PDF for the r^{th} order statistic in the list-of-lists format. The procedure is a direct implementation of the widely-published theorem on the distribution of the order statistics (e.g., Larsen and Marx, 1986, p. 145).

Special Issues: This procedure is robust for unspecified parameters in the population distribution. It is also fairly robust at returning the appropriate PDF when either n or r is unspecified. It is also robust when dealing with more than one segment in a PDF. This procedure was a cornerstone procedure that allowed the goodness-of-fit contributions discussed in Chapter 6 in this dissertation.

Examples:

- The PDF of the third order statistic from a sample of five items distributed according to the standard normal distribution is found by the commands:

```
> X := NormalRV(0, 1);
> Y := OrderStat(X, 5, 3);
```

- The minimum of 6 iid exponential(1) random variables, which in turn is exponentially distributed with $\lambda = 6$, is found by the commands:

```
> X := ExponentialRV(1);
> Y := OrderStat(X, 6, 1);
```

- In this example, an unspecified exponential distribution is entered as an argument.

```
> X := ExponentialRV(lambda);
> Y := OrderStat(X, 3, 2);
```

The result is the unspecified order statistic distribution of the sample median:

$$Y := [[x \rightarrow -6\,(e^{(-\lambda\,x)} - 1)\,\lambda\, e^{(-2\,\lambda\,x)}], [0, \infty], [\text{`}Continuous\text{`}, \text{`}PDF\text{`}]]$$

- In this example, n, r, and the distribution are unspecified.

```
> X := ExponentialRV(lambda);
> Y := OrderStat(X, n, r);
```

The result is the general form of the r^{th} order statistic from an exponential(λ) population:

$$Y := \left[\left[x \rightarrow \frac{\Gamma(n+1)\,(-e^{(-\lambda\,x)} + 1)^{(r-1)}\,\lambda\, e^{(-\lambda\,x\,(n-r+1))}}{\Gamma(r)\,\Gamma(n-r+1)}\right], [0, \infty], [\text{`}Continuous\text{`}, \text{`}PDF\text{`}]\right]$$

- As seen in the example in Section 2.7 for `PlotDist`, the `OrderStat` command may be embedded in loops and other programming functions in Maple. This ability is essential in the goodness-of-fit tests presented in Chapter 6 of this dissertation.

Algorithm: The algorithm is a straightforward implementation of the following result (found in Larsen and Marx, 1986, p. 145):

$$f_{X_{(r)}}(x) = \frac{n!}{(r-1)!(n-r)!} F_X(x)^{r-1} \cdot (1 - F_X(x))^{n-r} \cdot f_X(x).$$

As seen in Appendix D, the algorithm computes the new distribution segment by segment.

2.11 ProductRV and ProductIID

Syntax: The command

`ProductRV(`*RandomVariable1*`,` *RandomVariable2*`);`

returns the PDF of the product of the two random variables in the argument list.

Purpose: This procedure computes the PDF of products of random variables, i.e., $Z = XY$. The arguments can be any list-of-lists representation of a distribution. This procedure is another cornerstone procedure for the software package and is more fully explained in Chapter 4. The similar command ProductIID(*RandomVariable*, *n*) will compute the PDF of the n-fold product of iid random variables.

Special Issues: The two random variables in the argument list are assumed to be independent, but not necessarily identically distributed in ProductRV. Special care is used to allow for the multiplication of segmented random variables. The distribution of a product of random variables is frequently segmented, as in the case of a triangular random variable multiplied by another triangular random variable. Distributions do not have to fully-specified as seen in the fifth example. Also, the algorithm may be used in conjunction with Transform to compute the PDF of ratios of random variables, as seen in the fifth example.

Examples:

- The distribution of the product of a standard normal and a U(0, 1) random variable is found with the following commands:

```
> X := NormalRV(0, 1);
> Y := UniformRV(0, 1);
> Z := ProductRV(X, Y);
```

- The PDF of the product of two independent exponential(1) random variables is found with the following commands:

```
> X := ExponentialRV(1);
> Z := ProductRV(X, X);
```

- The PDF of the product of six independent standard normal random variables is found with:

```
> X := NormalRV(0, 1);
> Z := ProductIID(X, 6);
```

- The PDF of the product of two unspecified exponential(λ) distributions is a PDF in terms of the 'BesselK' function:

  ```
  > X := ExponentialRV(lambda);
  > Y := ProductRV(X, X);
  ```

  The procedure returns the following list-of-lists:

  $$Y := [[v \to 2\,\lambda^{2}\,\text{BesselK}(0,\,2\,\lambda\,\sqrt{v})],\,[0,\,\infty],\,[\text{'}Continuous\text{'},\,\text{'}PDF\text{'}]]$$

- For ratios of random variables, employ the transformation ability of the software as in the following example, which could be used to calculate the distribution of a random rate, given independent distributions for distance and time:

  ```
  > D := UniformRV(0, 10);
  > T := ExponentialRV(1);
  > R := ProductRV(D, Transform(T, [[x -> 1 / x], [0, infinity]]);
  ```

  Note, in this example, the call to `Transform` finds the distribution of $1/T$, so the PDF of the random ratio $R = D/T$ is computed with the `ProductRV` command.

- Chapter 4 contains other illustrative examples of this procedure's capabilities.

Algorithm: The algorithm is presented in Appendix F, and is explained in detail in Chapter 4.

2.12 SumRV and SumIID

Syntax: The command

`SumRV(`*RandomVariable1*`,` *RandomVariable2*`);`

returns the PDF of the sum of *RandomVariable1* and *RandomVariable2*.

Purpose: This procedure returns the list-of-lists formatted PDF of the convolution of two independent random variables. For example, it will produce the PDF of $Z = X + Y$, where X and Y are independent random variables. The similar command `SumIID(`*RandomVariable*`,` *n*`)` will compute the PDF for the n-fold convolution of iid random variables.

Special Issues: The random variables entered as arguments are assumed to be independent, but not necessarily having the same distribution in `SumRV`. The ability to compute differences of random variables is inherent in the software by employing the transformation ability, as in the fourth example below.

Examples:

- The sum of a standard normal random variable and a $U(0, 1)$ random variable has a PDF found as follows:

```
> X := NormalRV(0, 1);
> Y := UniformRV(0, 1);
> Z := SumRV(X, Y);
```

- The PDF of the sum of two independent unit exponential random variables, which is an Erlang PDF, is found as follows:

```
> X := ExponentialRV(1);
> Z := SumRV(X, X);
```

- The PDF of the sum of six standard normal random variables, which is N(0, 6) is found as follows:

```
> X := NormalRV(0, 1);
> Z := SumIID(X, 6);
```

- In this example one finds the PDF for the difference between a uniform and exponential random variable:

```
> X := UniformRV(0, 10);
> Y := ExponentialRV(1);
> D := SumRV(X, Transform(Y, [[y -> -y], [-infinity, infinity]]));
```

  Note, in this example, the call to `Transform` finds the negative distribution of the Y random variable, so the PDF of the random difference $D = X - Y$ is computed.

Algorithm: The algorithm for this procedure relies heavily on the `Transform` and `ProductRV` procedures. Specifically, to compute the convolution distribution of $Z = X + Y$, it carries out the transformation $Z = \ln(e^X e^Y)$ using the `Transform` and `ProductRV` procedures. A separate algorithm for the convolution of two random variables has been implemented by Berger (1995), but to date the author hasn't been able to get that procedure to be compatible in all cases of segmented random variables. The algorithm for this procedure is listed in Appendix D.

2.13 MinimumRV

Syntax: The command

`MinimumRV(`*RandomVariable1*, *RandomVariable2*`);`

returns the PDF of the minimum of the two random variables listed as arguments.

Purpose: `MinimumRV` is a procedure that produces the distribution of the random variable $Z = \min\{X, Y\}$, where X and Y are independent, continuous random variables. The procedure takes the PDFs of X and Y as arguments and returns the PDF of Z, all in the usual list-of-lists format.

Special Issues: The two random variables in the argument list are assumed to be independent, but not necessarily identically distributed. The procedure is robust on unspecified parameters for the distributions (see the third example below). The procedure is able to handle segmented random variables, such as in the first example below where two distributions with only one segment each in their PDFs have a minimum with two segments.

Examples:

- The minimum of a standard normal random variable and a U(0, 1) random variable is found as follows:

```
> X := NormalRV(0, 1);
> Y := UniformRV(0, 1);
> Z := MinimumRV(X, Y);
```

- The PDF of the minimum of two independent unit exponential random variables, which is also an exponential distribution with $\lambda = 2$, is determined as follows:

```
> X := ExponentialRV(1);
> Z := MinimumRV(X, X);
```

- The minimum of two unspecified iid Weibull random variables is found as follows:

```
> X := WeibullRV(lambda, kappa);
> Y := MinimumRV(X, X);
```

This call to `MinimumRV` returns `Y` as:

$$Y := \left[\left[x \to 2\,\lambda^{\sim -\kappa^{\sim}}\, x^{(\kappa^{\sim}-1)}\,\kappa^{\sim}\, e^{\left(-2\,\lambda^{\sim -\kappa^{\sim}}\, x^{\kappa^{\sim}}\right)}\right], [0, \infty], [\text{`}Continuous\text{`}, \text{`}PDF\text{`}]\right]$$

Algorithm: The procedure uses the CDF technique of finding the PDF of the minimum of two independent random variables. Careful consideration is given to segmented random variables, as the CDF technique requires the segments to be aligned properly. The algorithm is in Appendix D.

2.14 MaximumRV

Syntax: The command

`MaximumRV(`*RandomVariable1*`,` *RandomVariable2*`);`

returns the PDF of the maximum of the two random variables listed as arguments.

Purpose: `MaximumRV` is a procedure that produces the distribution of the random variable $Z = \max\{X, Y\}$ where X and Y are independent, continuous random variables. The procedure takes as arguments the PDFs of X and Y, and returns the PDF of Z, all in the list-of-lists format.

Special Issues: The two random variables in the argument list are assumed to be independent, but not necessarily identically distributed. Notice there are no procedures for the maximum or minimum of n iid random variables. Such a determination is already possible with the procedure `OrderStat`.

Examples:

- The maximum of a standard normal random variable and a $U(0, 1)$ random variable is found as follows:

```
> X := NormalRV(0, 1);
> Y := UniformRV(0, 1);
> Z := MaximumRV(X, Y);
```

- The maximum of two independent unit exponential random variables is found as follows:

```
> X := ExponentialRV(1);
> Z := MaximumRV(X, X);
```

  Note that this could represent the system lifetime of a parallel arrangement of a two-component system.

Algorithm: This procedure relies on the `MinimumRV` and `Transform` procedures to determine the distribution of the maximum. Specifically, it maximizes $Z = \max\{X, Y\}$ by performing the transformation $Z = -\min\{-X, -Y\}$. The algorithm is found in Appendix D.

2.15 Maximum likelihood estimation

Syntax: The command

`MLE(`*RandomVariable*`, `*SampleList*`, `*ParameterList*`);`

returns the MLE of the parameters listed in *ParameterList* as a result of the sample observations listed in *SampleList* for sampling from a population with a distribution represented by *RandomVariable*.

Purpose: The purpose of this procedure is to find real, and symbolic, where possible, estimates of parameters for distributions. The argument *SampleList* can either be a fully-specified list of real numbers or the unspecified list of strings. The argument *ParameterList* must be a list of the unknown parameters to be estimated. The argument *RandomVariable* must be a list-of-lists specified distribution of a random variable.

Special Issues: One is not limited to performing ML estimates on known random variables. In the explorations envisioned for this software, any list-of-lists represented distribution may be used in the estimation procedure. Clearly one is limited by Maple's solving capability, especially in the case of unspecified samples. An advantage of this approach to ML estimation is that sometimes we can actually find the distribution of the estimator. For instance, in the third example below, the procedure `MLE` returns a function of independent, univariate random variables, in this case $\bar{X}$ of the inverse Gaussian distribution. After the distribution of the estimator, called `XB` in the example, is found, one can find its mean, variance, quantiles, and so forth, using APPL.

Examples:

- For the semi-specified inverse Gaussian distribution where σ is known (or assumed) but μ is unknown, one finds the estimate of μ as follows:

  ```
  > X := InverseGaussianRV(1, mu);
  > muhat := MLE(X, [1, 2, 3, 4, 5], [mu]);
  ```

  The call returns the estimated value for μ of 3.

- Should both parameters need to be estimated, then the parameter list would include both of the unknown parameters as follows:

  ```
  > X := InverseGaussianRV(lambda, mu);
  > paramhat := MLE(X, [1, 2, 3, 4, 5], [lambda, mu]);
  ```

  The call returns the estimated values in the form of a list assigned to `paramhat`, where `paramhat` is now the list `[300/37, 3]` corresponding to the MLEs $\hat{\lambda} = \frac{300}{37}$ and $\hat{\mu} = 3$.

- For the case of a general random sample $x_1, x_2, \ldots, x_n$, i.e., the sample is unspecified, but n is fixed, one would type the following:

  ```
  > X := NormalRV(mu, 1);
  > Y := MLE(X, [x1, x2, x3, x4, x5], [mu]);
  ```

  This code returns the MLE $\bar{X}$ for $n = 5$. Should one want to find the distribution of this estimator, the following commands would be used:

  ```
  > XB := Transform(SumIID(X, 5), [[x -> x / 5], [-infinity, infinity]]);
  ```

Algorithm: The algorithm, found in Appendix D, is a straightforward implementation of maximum likelihood estimation using the log-likelihood function.

Chapter 3

Transformations of Univariate Random Variables

3.1 Introduction

As mentioned in Chapter 2, the procedure `Transform` is a key procedure in APPL. Its uses are many, and `Transform` is often embedded in other procedures, as in `MaximumRV` for example. A generalized version of the univariate change-of-variable technique for transforming continuous random variables is presented here. Extending a theorem from Casella and Berger (1990) for many–to–1 transformations, to include a more general set of univariate transformations. Specifically, transformations can range from 1–to–1 to many–to–1 on various subsets of the support of the random variable of interest.

In its simplest application, the change-of-variable technique is used to determine the distribution of a continuous random variable Y given the distribution of a continuous random variable X and a 1-to-1 transformation from the support of X to the support of Y. As the conditions on the transformation $Y = g(X)$ become more general, the technique requires more detail in its description and is less likely to appear in introductory probability and statistics texts. Casella and Berger (1990, p. 51) discuss transforming random variables using the change-of-variable technique when the entire transformation is many–to–1, except for a finite number of points, that is, the cardinality of the set $g^{-1}(y)$ is the same for almost all y in the support of Y. Hogg and Craig (1995, p. 190) extend this many–to–1 technique to n-dimensional random variables. In this chapter, a more general univariate case is considered in which the transformations are "piecewise many–to–1," where "many" may vary based on the subinterval of the support of Y under consideration. What follows is a theorem for this case and an algorithm (in Appendix E) for a computer algebra system implementation of the result. Although the theorem is a straightforward generalization of Casella and Berger's, there are a number of details that have to be addressed in order to produce an algorithm for finding the PDF of Y. The resulting computer algebra system implementation of the theorem relieves analysts, researchers, and students from arduous computations.

Consider the following example. Let $f_X(x) = \frac{1}{3}$ for $-1 < x < 2$, be the PDF for the random variable X. Consider the transformation $Y = g(X) = X^2$. This transformation is a 2–to–1 transformation on the interval $X \in (-1, 1)$ (except at $X = 0$) and it is 1–to–1 on the interval $X \in [1, 2)$; see Figure 3.1 on page 54. Some introductory probability texts use this as an example (e.g., Larsen and Marx, 1986, p. 137), but fail to state a general theorem that treats such piecewise many-to-1 transformations. The difficulty lies in identifying an appropriate partition of the support of X and a corresponding partition for the support of Y, and then determining which of these subsets of the support of X correspond to each of the subsets of the support of Y. A further complication is encountered when the transformation itself is either discontinuous or non-differentiable at certain points. For example, consider the random variable X, where $f_X(x) = (x+1)/18$ for $-1 < x < 5$, and the transformation (see Figure 3.3 on page 57):

$$Y = g(X) = \begin{cases} X^2 & -1 < X < \frac{3}{2} \\ X & \frac{3}{2} < X < 5 \end{cases}.$$

In this example, the transformation is discontinuous as well as "piecewise many-to-1." This theorem and resulting implementation in a computer algebra system will determine the PDF of Y for such transformations.

In Section 3.2, the theorem, modeled after Casella and Berger's, is presented. This theorem has been presented more generally in earlier papers. Barr and Zehna (1971, p. 225) consider multivariate many-to-one transformations. Rohatgi (1976, pp. 73–74) and Port (1994, p. 462) consider the piecewise many-to-one case in the univariate and multivariate settings, respectively. This theorem is strictly univariate, but permits implementation in a computer algebra system. It determines the distribution of $Y = g(X)$ for any univariate random variable X of the continuous type with few restrictions on the transformation $g(X)$. Note that this theorem will be stated in a somewhat elaborate form for the purpose of facilitating its implementation. Section 3.3 discusses an algorithmic implementation of the theorem using the computer algebra system capabilities of the Maple V software package. Section 3.4 illustrates the usefulness of the algorithm.

3.2 Theorem

Before the theorem is presented, an overview of the rationale for the notation is in order. Assume that the support of X, denoted by $\mathcal{X}$, consists of a finite union of open intervals. The points $x_1 < x_2 < \cdots < x_{n+1}$ generate n consecutive subintervals and are determined as follows. The first subinterval of $\mathcal{X}$ begins with x_1 and the last subinterval of $\mathcal{X}$ ends with x_{n+1}. The remaining x_i's correspond to other endpoints of the intervals of $\mathcal{X}$ or to the locations in $\mathcal{X}$ where the function g is discontinuous or non-differentiable, or g' is zero. Let $g_i(x)$ denote the restriction of $g(x)$ to (x_i, x_{i+1}); by design, g_i is monotone and therefore invertible. Let $X^* = \{x_1, x_2, \ldots, x_{n+1}\}$; note that since X is continuous, $P(X \in X^*) = 0$.

Let Y^* be a set of points on the y-axis, to partition the support of Y into subintervals. The range of each g_i, denoted by (m_i, M_i), either contains or is disjoint from each Y subinterval. The set Y^* is designed so that the final PDF of Y may be specified by designating its value on each Y subinterval. The set Y^* is defined using one-sided limits because the transformations $g_1, g_2, \ldots, g_n$ are not necessarily defined at the points in X^*. The set I_j consists of integers i with the property that the range of g_i is either equal to or properly contains the jth Y subinterval. The PDF for each Y subinterval depends on the PDF of X and the associated transformations g_i.

Theorem. Let X be a random variable of the continuous type with PDF $f_X(x)$ and with support $\mathcal{X}$, where $\mathcal{X}$ consists of a finite union of open intervals. Let $g(x)$ be a real-valued function whose domain includes $\mathcal{X}$. Let $-\infty \le x_1 < x_2 < \cdots < x_n < x_{n+1} \le +\infty$ be a sequence of extended real numbers which satisfy the following conditions:

1. The sequence includes the endpoints of the intervals whose union is $\mathcal{X}$.
2. $f_X(x)$ is continuous on each open interval $A_i = (x_i, x_{i+1})$ for $i = 1, 2, \ldots, n$.
3. If $f_X(x)$ is not identically zero on A_i, then the function $g_i(x)$, which is the restriction of $g(x)$ to A_i, is monotone on A_i and has a nonzero derivative at each point in A_i, for $i = 1, 2, \ldots, n$.

Let $X^* = \{x_1, x_2, \ldots, x_{n+1}\}$.

Let $\alpha = \{i | f_X(x) \text{ is not identically zero on } A_i\}$.

Let $m_i = \min\left\{\lim_{x \downarrow x_i} g(x), \lim_{x \uparrow x_{i+1}} g(x)\right\}$ for $i = 1, 2, \ldots, n$.

Let $M_i = \max\left\{\lim_{x \downarrow x_i} g(x), \lim_{x \uparrow x_{i+1}} g(x)\right\}$ for $i = 1, 2, \ldots, n$.

Let $Y^* = \cup_{i \in \alpha} \{m_i, M_i\}$.

Let $m = \| Y^* \| - 1$, where $\| \cdot \|$ denotes cardinality.

Order the elements of y_j of Y^* so that $y_1 < y_2 < \cdots < y_{m+1}$.

Let $I_j = \{i | m_i \le y_j \text{ and } y_{j+1} \le M_i\}$, for $j = 1, 2, \ldots, m$.

Then for $y \in (y_j, y_{j+1})$.

$$f_Y(y) = \sum_{i \in I_j} f_X\left(g_i^{-1}(y)\right) \left|\frac{dg_i^{-1}(y)}{dy}\right|$$

for $j = 1, 2, \ldots, m$.

Proof. Without loss of generality, consider the jth Y subinterval (y_j, y_{j+1}). Also suppose that a and b are any points that lie inside (y_j, y_{j+1}) such that $a < b$. Furthermore, let $M_i = \max\{g_i^{-1}(a), g_i^{-1}(b)\}$ and $m_i = \min\{g_i^{-1}(a), g_i^{-1}(b)\}$ for $i \in I_j$. As Hogg and Craig (1995, p. 190) point out,

$$P(a < Y < b) = \sum_{i \in I_j} P\left(m_i < X < M_i\right).$$

Since Y is a continuous random variable,

$$\begin{aligned} P(a < Y < b) &= \sum_{i \in I_j} \int_{m_i}^{M_i} f_X(x)\, dx \\ &= \sum_{i \in I_j} \int_a^b f_X\left(g_i^{-1}(y)\right) \left|\frac{dg_i^{-1}(y)}{dy}\right| dy \\ &= \int_a^b \sum_{i \in I_j} f_X\left(g_i^{-1}(y)\right) \left|\frac{dg_i^{-1}(y)}{dy}\right| dy \end{aligned}$$

where one performs the change-of-variable $y = g_i(x)$, or equivalently $x = g_i^{-1}(y)$, and has used the absolute value to treat both increasing and decreasing transformations (see Freund 1992, p. 268). Referring to Theorem 7.1 in Freund (1992), one sees that the integrand

$$\sum_{i \in I_j} f_X\left(g_i^{-1}(y)\right) \left|\frac{dg_i^{-1}(y)}{dy}\right|$$

is the PDF for Y on the subinterval (y_j, y_{j+1}). □

3.3 Implementation

It could be quite tedious to implement this theorem by hand for large m and n. With the onset of computer algebra systems such as Maple V, however, one may implement algorithms that correspond to theorems such as this with relative ease. This algorithmic implementation is included in the Appendix E and parallels the theorem. Two main implementation issues emerged. First, Maple may produce several candidates for g_i^{-1}, e.g., when $g_i(x) = x^2$, Maple returns $g_i^{-1}(y) = -\sqrt{y}$ and $g_i^{-1}(y) = \sqrt{y}$. The correct inverse is selected by requiring that $g_i^{-1}(g_i(c_i)) = c_i$, where c_i is any point in the ith X subinterval. When the ith X subinterval has finite endpoints, c_i can be chosen to be the midpoint of the subinterval. In the cases where $x_1 = -\infty$ or $x_{n+1} = \infty$ [e.g., $X \sim N(\mu, \sigma^2)$], c_i must be selected more carefully. The algorithm for determining c_i is:

1. If $x_1 = -\infty$ and $x_2 = +\infty$, then $c_1 = 0$.
2. If $x_1 = -\infty$ and $x_2 \neq +\infty$, then $c_1 = x_2 - 1$.
3. If $x_n \neq -\infty$ and $x_{n+1} = +\infty$, then $c_n = x_n + 1$.
4. For all other cases, $c_i = \dfrac{x_i + x_{i+1}}{2}$.

The second implementation issue involves redundancies in lists. Maple doesn't recognize that the integer 3 and the floating-point value 3.0, for example, are redundant in a list. The `ReduceList` procedure finds all adjacent points that differ by a small prescribed δ and discards redundant floating-point type elements.

The data structure for g is a "list of two lists" that is similar in nature to that of f. Here one characterizes g by listing its monotone components and the endpoints of the corresponding domains. Example 3.4.1 gives a detailed illustration of the data structures for f and g.

The following are some additional implementation issues that arose during the coding of the algorithm:

- The user must supply $x_1, x_2, \ldots, x_{n+1}$. This ordered list consists of the endpoints of the open intervals which constitute $\mathcal{X}$ and all locations in $\mathcal{X}$ where $g(x)$ is discontinuous or non-differentiable, or $g'(x)$ is zero. A preprocessor to determine elements of X^* corresponding to $g'(x) = 0$ could be added to the algorithm if desired. Thus if the support of X is $(-1, 2)$, and $Y = g(X) = X^2$, this preprocessor would include $x = 0$ in X^* yielding $X^* = \{-1, 0, 2\}$.

- The partition points $x_1, x_2, \ldots, x_{n+1}$ must be chosen so that $f_X(x)$ is not defined piecewise on any X subinterval (e.g., if X has a triangular distribution, the mode must be a partition point).

- Adding extra x-values in $\mathcal{X}$ to X^* that are not maxima, minima or saddle points of $g(x)$ or discontinuities of $f_X(x)$ or $g(x)$ will not affect the correctness of the algorithm's implementation of the theorem. although the implementation will be slower.

- Many transformations such as $g(x) = x + e^x$ do not have an inverse that can be expressed in a closed-form. Numerical methods could be used in conjunction with the algorithm to find individual values of $f_Y(y)$. This has not been implemented in the algorithm. If $g(x) = x + e^x$, for example, then

$$\left|\frac{dg^{-1}(y)}{dy}\right| = \left.\frac{1}{1+e^x}\right|_{x=g^{-1}(y)}.$$

- Periodic transformations can present a problem in the implementation of the theorem. Maple uses the usual principal inverses for $\sin x$, $\cos x$, and $\tan x$, but the restricted ranges of these principal inverses are often inappropriate. When the algorithm tries to identify which inverse is appropriate, it is only given one choice, the principal inverse, but the domain of X might not coincide with the range of the principal inverse. Example 3.4.4 illustrates the problem and a circumvention.

- No error-trapping has been done to insure that $f_X(x)$ is a legitimate PDF, i.e.

$$\int_{-\infty}^{\infty} f_X(x)dx = 1, f_X(x) \geq 0 \ \forall \ x.$$

  Also, there is limited error-trapping on $g(x)$ in that the procedure gives an error message when the inverse cannot be found.

- Parameters are allowed in $f_X(x)$ but not in $g(x)$.

3.4 Examples

This section contains some illustrative examples of using the algorithm described in the previous section to determine the distribution of $Y = g(X)$. Before using the program, its performance on several common transformations was tested. Given $X \sim N(0,1)$ and $Y = g(X) = X^2$, the program returned the PDF for a χ_1^2 random variable. Given $X \sim N(\mu, \sigma^2)$ and $Y = g(X) = (X - \mu)/\sigma$, the program returned the PDF for a standard normal random variable. Now consider more complex examples that illustrate the theorem's implementation.

Example 3.4.1 Consider the first example from the introduction: given the random variable $X \sim U(-1, 2)$, find the distribution of $Y = g(X) = X^2$. Given X^*, the algorithm determines the relevant partitions of the supports of X and Y. Then it determines which X subintervals map onto which Y subintervals. The subintervals are apparent in Figure 3.1. Note the set X^* is displayed on the horizontal axis, Y^* on the vertical axis, both marked with the $\times$ symbol. The transformation is partitioned into monotone segments (with identical or disjoint ranges) delineated by the $+$ symbol. The assignment of data structures for f and g and the call to `Transform` are as follows:

```
> X := [ [x -> 1 / 3], [-1, 2], ['Continuous','PDF'] ];
```

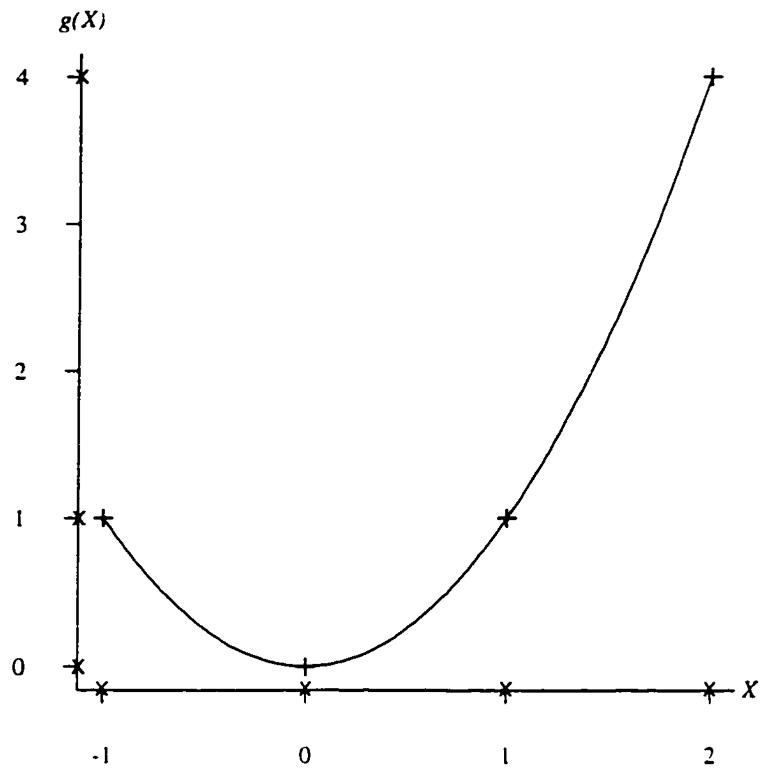


Figure 3.1: The transformation $Y = g(X) = X^2$ for $-1 < X < 2$.

```
> g := [ [x -> x ^ 2, x -> x ^ 2], [- infinity, 0, infinity] ];
> Y := Transform(X, g);
```

The program determines that the transformation is 2–to–1 on $-1 < x < 1$ (excluding $x = 0$) and 1–to–1 on $1 \le x < 2$. Since $Y = X^2$ has two inverses, the program determines which inverse to apply to which X subinterval. The resulting PDF for Y is

$$f_Y(y) = \begin{cases} \frac{1}{3\sqrt{y}} & 0 < y < 1 \\ \frac{1}{6\sqrt{y}} & 1 < y < 4 \end{cases} .$$

The corresponding returned value of Y is a "list-of-lists" represented distribution that specifies the PDF of Y:

```
[[y -> 1 / (3 * sqrt(y)), y -> 1 / (6 * sqrt(y))],
      [0, 1, 4], ['Continuous','PDF']]
\vspace*{-0.25in}
```

Example 3.4.2 Consider the random variable $X \sim U(0, 7)$. Find the PDF of $Y = g(X) = ||X - 3| - 1|$. A graphical representation of this

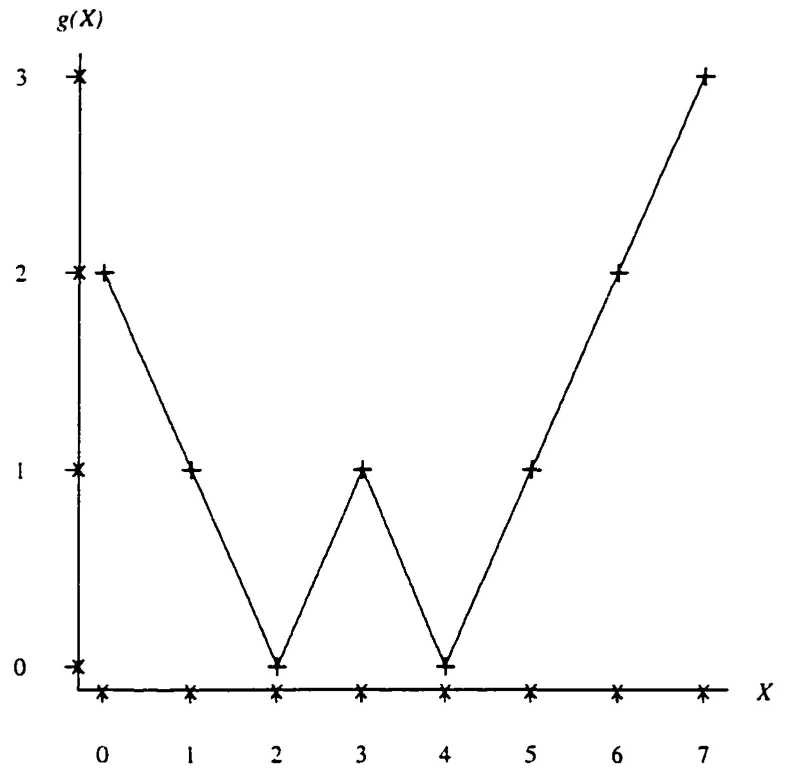


Figure 3.2: The transformation $Y = g(X) = ||X - 3| - 1|$ for $0 < X < 7$.

transformation is shown in Figure 3.2, with X^*, Y^*, and the monotone segments of g marked as before. This transformation is more complex than

Example 3.4.1 in that it is 4–to–1 for $0 < y < 1$, 2–to–1 for $1 < y < 2$, and 1–to–1 for $2 < y < 3$. The program yields the correct PDF for Y:

$$f_Y(y) = \begin{cases} \frac{4}{7} & 0 < y < 1 \\ \frac{2}{7} & 1 < y < 2 \\ \frac{1}{7} & 2 < y < 3 \end{cases}.$$

Example 3.4.3 Consider the second example from the introduction: if the random variable X has PDF $f_X(x) = (x+1)/18$ for $-1 < x < 5$, find the distribution of

$$Y = g(X) = \begin{cases} X^2 & -1 < X < \frac{3}{2} \\ X & \frac{3}{2} < X < 5 \end{cases}.$$

The appropriate partition for the transformation is depicted in Figure 3.3. The program determines the following PDF for Y:

$$f_Y(y) = \begin{cases} \frac{1}{18\sqrt{y}} & 0 < y < 1 \\ \frac{\sqrt{y}+1}{36\sqrt{y}} & 1 < y < 1.5 \\ \frac{(2y+3)\sqrt{y}+1}{36\sqrt{y}} & 1.5 < y < 2.25 \\ \frac{y+1}{18} & 2.25 < y < 5 \end{cases}.$$

Example 3.4.4 As a final example, consider the problem that Casella and Berger (1990) discussed, without providing $f_Y(y)$, as a prelude to

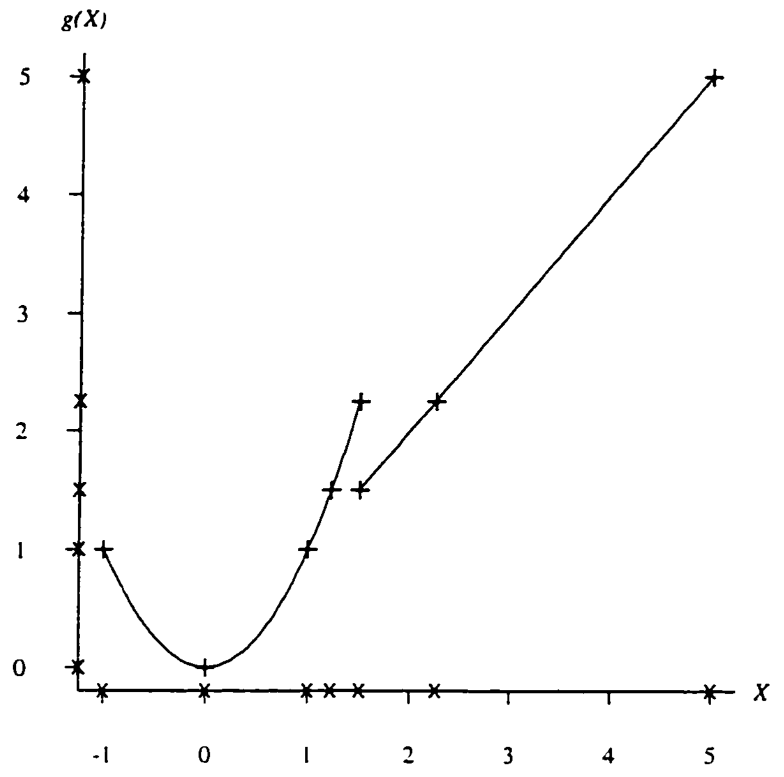


Figure 3.3: The transformation $Y = g(X)$ has a discontinuity and is variously 1–to–1 and 2–to–1 on different subsets of the support of X.

their theorem. Letting X be a uniform random variable on $(0, 2\pi)$, find the distribution of $Y = g(X) = \sin^2(X)$. As Figure 3.4 shows, this transformation is 4–to–1 for the single Y subinterval $(0, 1)$. Furthermore, since $n = 4$, $X^* = \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi\}$. The PDF of Y is

$$f_Y(y) = \frac{1}{\pi\sqrt{y - y^2}} \qquad 0 < y < 1$$

which is commonly known as the arcsin distribution (see Johnson, Kotz, and Balakrishnan (1995, p. 212). To overcome the principal inverse diffi-

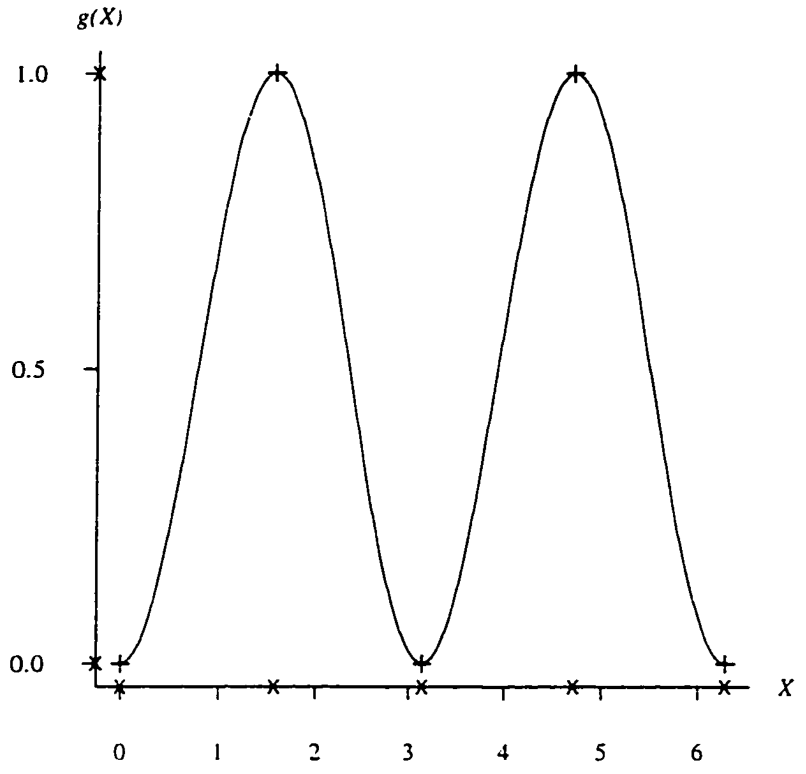


Figure 3.4: The transformation $Y = g(X) = \sin^2(X)$ for $0 < X < 2\pi$.

culty alluded to in the previous section, the following equivalent situation has been devised: consider the random variable $X \sim U(-\frac{\pi}{2}, \frac{\pi}{2})$ and let $Y = g(X) = \sin^2(X)$. In this case the domain of X will be the same as the range of g^{-1} which results from using the standard arcsin function.

This solution yielded a random variable with a notable feature. The distribution's HF, $h_Y(y) = \dfrac{f_Y(y)}{1 - F_Y(y)}$, might be of interest to a reliability engineer. For this distribution,

$$h_Y(y) = \frac{2}{\sqrt{y - y^2}\,[\pi - 2\arcsin(2y - 1)]} \qquad 0 < y < 1.$$

By plotting this HF, one sees it has the rare property of a closed-form, "bathtub"-shaped HF. Furthermore, one can now apply the transformation $W = g(Y) = \lambda Y$ in order to derive a random variable W that is a one-parameter random variable with a closed-form, bathtub-shaped HF. Although the arcsin distribution has been discussed in the literature, there appears to be no mention of the fact that it has a bathtub-shaped HF. Most distributions with bathtub-shaped hazard functions must be analyzed numerically because their hazard functions are not closed form. One of the useful features of the `Transform` procedure is that it can help the practitioner gain insight into what transformations could result in a useful model. In this example, it becomes apparent that the transformation "crowds" the uniformly distributed X random variable into a distribution for Y that is "heavy" on either end of its support. The result is a random variable with a bathtub-shaped hazard function. The program enables the model designer to confirm, or gain insight into the univariate transformations that could result in a useful model.

3.5 Conclusion

The tool provided by the `Transform` procedure will be useful to the practitioner as well as the academic. The academic might use this tool to find new distributions. Let X be a beta random variable with specified parameters, for example, and let

$g(X) = X^2$. Applying g several times in succession may yield a distribution that is useful in modeling. Alternatively, students might be asked to combine a set of ten transformations and ten distributions in order to create 100 new distributions, picking out the interesting properties of the more notable ones.

The practitioner, on the other hand, may use the **`Transform`** procedure for the specific purpose of iterative probabilistic model design, illustrated in Example 3.4.4. Examples provided in Chapter 7 use **`Transform`** extensively.

Chapter 4

Products o: Random Variables

4.1 Introduction

Another keystone procedure is that of computing the probability density function of the product of two independent random variables. The procedure ProductRV is very useful for that purpose and is also embedded in the SumRV procedure. Rohatgi's well-known result (1976, p. 141) for determining the distribution of the product of two random variables is straightforward to derive, but difficult to implement. Let X and Y be continuous random variables with joint PDF $f_{X,Y}(x,y)$. The PDF of $V = XY$ is

$$f_V(v) = \int_{-\infty}^{\infty} f_{X,Y}\left(x, \frac{v}{x}\right) \frac{1}{|x|}\, dx\,.$$

The implementation of this result, however, is not straightforward for general X and Y. Difficulties occur as a result of both the myriad variations to the limits of integration and the propensity of the PDF of V to be defined in a piece-wise manner.

The cases when X and Y are independent and may have probability density functions defined in a piece-wise fashion are considered in this chapter. Appendix F contains an algorithm that handles these difficulties and is genesis for the procedure ProductRV.

4.2 Theorem

A simple theorem is presented in this section which illustrates some of the issues associated with a general algorithm for determining the PDF of the product of two independent random variables. For simplicity, assume that the random variable X has support on the interval (a, b) and the random variable Y has support on the interval (c, d). Also, the product space of the two random variables is assumed to fall entirely in the first quadrant.

Theorem. Let X be a random variable of the continuous type with PDF $f(x)$ which is defined and positive on the interval (a, b), where $0 < a < b < \infty$. Similarly, let Y be a random variable of the continuous type with PDF $g(y)$ which is defined and positive on the interval (c, d), where $0 < c < d < \infty$. The PDF of $V = XY$ is

$$h(v) = \begin{cases} \int_a^{v/c} g\left(\frac{v}{x}\right) f(x) \frac{1}{x}\, dx & ac < v < ad \\ \int_{v/d}^{v/c} g\left(\frac{v}{x}\right) f(x) \frac{1}{x}\, dx & ad < v < bc \\ \int_{v/d}^{b} g\left(\frac{v}{x}\right) f(x) \frac{1}{x}\, dx & bc < v < bd \end{cases}$$

when $ad < bc$,

$$h(v) = \begin{cases} \int_a^{v/c} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ac < v < ad \\ \int_{v/d}^{b} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ad < v < bd \end{cases}$$

when $ad = bc$, and

$$h(v) = \begin{cases} \int_a^{v/c} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ac < v < bc \\ \int_a^{b} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & bc < v < ad \\ \int_{v/d}^{b} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ad < v < bd \end{cases}$$

when $ad > bc$.

Proof. Only the case of $ad < bc$ is considered. The other cases are proven analogously. Using the transformation technique (Hogg and Craig, 1995, page 173), the dummy transformation $Z = X$ and the transformation $V = XY$ constitute a 1–1 mapping from $\mathcal{A} = \{(x, y)|a < x < b, c < y < d\}$ to $\mathcal{B} = \{(z, v)|a < z < b, cz < v < dz\}$. Let u denote the transformation and w the inverse transformation. The transformation, inverse, and Jacobian are:

$$\begin{aligned} z &= u_1(x, y) = x & \qquad x &= w_1(z, v) = z \\ v &= u_2(x, y) = xy & \qquad y &= w_2(z, v) = v/z \end{aligned}$$

$$J = \begin{vmatrix} 1 & 0 \\ -v/z^2 & 1/z \end{vmatrix} = 1/z.$$

The joint PDF of Z and V is

$$f_{Z,V}(z,v) = f(w_1(z,v))g(w_2(z,v))\,|J| \qquad (z,v) \in \mathcal{B},$$

or

$$f_{Z,V}(z,v) = f(z)g(v/z)\frac{1}{z} \qquad (z,v) \in \mathcal{B}.$$

Integrating with respect to z over the appropriate intervals and replacing z with x in the final result yields

$$h(v) = \begin{cases} \int_a^{v/c} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ac < v < ad \\ \int_{v/d}^{v/c} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & ad < v < bc \\ \int_{v/d}^{b} g\left(\frac{v}{x}\right) f(x)\frac{1}{x}\,dx & bc < v < bd \end{cases}$$

as desired. $\square$

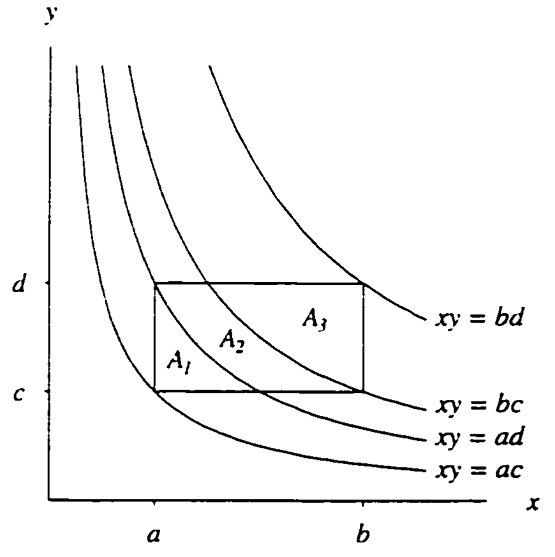


Figure 4.1: The support of X and Y when $ad < bc$.

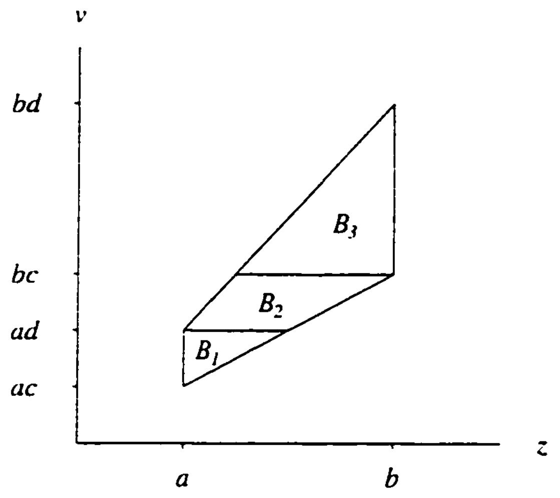


Figure 4.2: The mapping of $Z = X$ and $V = XY$ when $ad < bc$.

The geometry associated with the transformation is shown in Figures 4.1 and 4.2. Note that the transformation maps A_i onto B_i for $i = 1, 2, 3$. Although the transformation technique has been used to prove this theorem, the CDF technique could also have been used.

4.3 Implementation

The theorem in the previous section illustrates the importance of considering the magnitudes of the product of the coordinates of the southeast and northwest corners of the product space (e.g., (b, c) and (a, d)) when it lies entirely in the first quadrant. In order to apply the theorem to any continuous random variables X and Y, three generalizations need to be addressed.

1. Analogous theorems must be written for the cases when the (a, b) by (c, d) rectangle lies wholly in one of the other three quadrants.

2. Instead of having probability density functions which are specified by a single standard function over their entire support, the random variables X and Y may be defined in a piece-wise manner over several intervals, forming many segments to the PDF (e.g. the triangular distribution).

3. The cases when 0 and $\pm\infty$ belong to the endpoints of the intervals which constitute the support of X and Y must be considered.

In fact, these generalizations result in 24 different cases that must be considered in order to correctly compute the limits of integration of the theorem. The paragraphs below address these cases. For quadrants II, III, and IV, the limits of integration must be set appropriately based on the geometry of the transformation.

For random variables that are defined piece-wise over various intervals, let n be the number of intervals for X and let m be the number of intervals for Y. There are mn rectangular "product spaces" and the contribution of each to the value of the PDF of $V = XY$ must be computed. Furthermore, each "product space" can contribute differently to the PDF of V on up to three segments of the support of V. As a result, the PDF of V tends to become complicated very quickly, with an upper limit of $3mn$ segments to its PDF. For example, the product of two U$(0, 1)$ random variables yields a random variable V with only one segment (see Example 4.4.1). But with only a slight change, e.g., $X \sim$ U$(1, 2)$ and $Y \sim$ U$(3, 4)$, yields a $V = XY$

defined differently on three segments (see Example 4.4.2).

The case where the support of a random variable contains 0 (e.g., $U(-1, 2)$) poses special difficulty since some of the rectangular product spaces will not lie wholly in any one quadrant and cannot be handled by the previously developed techniques. One solution to this difficulty is to add 0 as one of the endpoints of the intervals for X and Y whenever this case occurs, producing redundant segments, i.e., two segments on either side of zero with the same formula for the PDF.

The algorithm consists of a set-up portion, followed by nested loops that determine the contribution to the PDF of $V = XY$ separately for each of the four quadrants. Appendix F contains the set-up portion and the algorithm associated with the first quadrant. The algorithm for the other quadrants is similar.

The set-up phase begins by setting n and m to the number of intervals that form the support of X and Y. Next, 0 is added as an interval delimiter for X and/or Y if the random variable can assume both positive and negative values, and 0 is not already an interval delimiter. Finally, the endpoints of the intervals which form the support of V are determined by taking all products of the endpoints of the X intervals times the endpoints of the Y intervals.

A nested set of loops follows that treats all pairings of X and Y intervals. As shown in Figure 4.1, the coordinates (a, c) are assigned to the southwest corner of the current rectangle of interest, and the coordinates (b, d) are assigned to the northeast corner of the current rectangle of interest. A test to determine which quadrant contains the current rectangle is made at this point. Adding 0 as an interval delimiter in the set-up phase assures that the current rectangle will be completely contained in just one of the quadrants. Once the quadrant is determined, tests on c and d determine which integrals should be computed and the appropriate limits of integration. Finally, the insertion of 0 sometimes leads to a PDF for V with the same formula on both sides of 0. If this occurs, the program simplifies the PDF by removing 0 as an interval endpoint if the function is defined at 0.

4.4 Examples

This section contains applications of using the Maple procedure `ProductRV`.

Example 4.4.1 Consider the random variable $X \sim$ U(0, 1) and the random variable $Y \sim$ U(0, 1). Find the distribution of $V = XY$.

This is a simple application of the algorithm. The following Maple code defines the random variables X and Y and returns the PDF of their product. Note, the procedure `Uniform` returns the PDF in a Maple list-of-lists data structure outlined in Chapter 2.

```
X := UniformRV(0, 1);
Y := UniformRV(0, 1);
V := ProductRV(X, Y);
PDF(V);
```

The resulting PDF for $V = XY$ is

$$h(v) = -\ln v \qquad 0 < v < 1,$$

which is readily verified by hand.

Example 4.4.2 Consider the random variable $X \sim U(1,2)$ and the random variable $Y \sim U(3,4)$. Find the distribution of $V = XY$.

This is a straightforward application of the algorithm in that all segments of X and Y are in the first quadrant. In this example, the PDF is less simple to calculate by hand. The program yields the following PDF for $V = XY$:

$$h(v) = \begin{cases} \ln v - \ln 3 & 3 < v < 4 \\ \ln 4 - \ln 3 & 4 < v < 6 \\ 3\ln 2 - \ln v & 6 < v < 8 \end{cases}.$$

Note that while the PDF of both X and Y are defined on single segments that have positive interval limits, the PDF of V is defined on three segments.

Example 4.4.3 Consider the random variable $X \sim U(-1,2)$ and the random variable $Y \sim U(-3,4)$. Find the PDF of $V = XY$. This example will test whether the algorithm handles the case of 0 in the support of X and Y correctly. The program yields the correct PDF for V:

$$h(v) = \begin{cases} \frac{1}{21}\ln\left(-\frac{6}{v}\right) & -6 < v < -4 \\ \frac{1}{21}\ln\left(\frac{24}{v^2}\right) & -4 < v < 0 \\ \frac{1}{21}\ln\left(\frac{24}{v^2}\right) & 0 < v < 3 \\ \frac{1}{21}\ln\left(\frac{8}{v}\right) & 3 < v < 8 \end{cases}.$$

Maple returns a mathematically equivalent PDF with a slightly different functional form. Maple's calculation for the first segment of V yields $\frac{1}{21}\ln(2) + \frac{1}{21}\ln(3) - \frac{1}{21}\ln(v) + \frac{1}{21}I\pi$, which reduces to the above first segment, remembering that since v is negative in this segment $\ln(v) = \ln(-1) + \ln(-v) = I\pi + \ln(-v)$, so the imaginary portion cancels.

Example 4.4.4 Consider the random variable $X \sim$ triangular(1, 2, 3) and the random variable $Y \sim$ triangular(1, 2, 4). Find the PDF of $V = XY$. This non-uniform example illustrates the case of several rectangular product spaces. The Maple code in this case is:

```
X := TriangularRV(1, 2, 3);
Y := TriangularRV(1, 2, 4);
V := ProductRV(X, Y);
PDF(V);
```

The resulting PDF for V has six segments:

$$h(v) = \begin{cases} -4v/3 + 2/3 \ln v + 2v/3 \ln v + 4/3 & 1 < v < 2 \\ -8 + 14/3 \ln 2 + 7v/3 \ln 2 + 10v/3 - 4 \ln v & \\ \quad -5v/3 \ln v & 2 < v < 3 \\ -4 + 14/3 \ln 2 + 7v/3 \ln 2 + 2v - 2 \ln v & \\ \quad -v \ln v - 2 \ln 3 - 2v/3 \ln 3 & 3 < v < 4 \\ 44/3 - 14 \ln 2 - 7v/3 \ln 2 - 8v/3 - 2 \ln 3 & \\ \quad +22/3 \ln v - 2v/3 \ln 3 + 4v/3 \ln v & 4 < v < 6 \\ 8/3 - 8 \ln 2 - 4v/3 \ln 2 - 2v/3 + 4/3 \ln v & \\ \quad +v/3 \ln v + 4 \ln 3 + v/3 \ln 3 & 6 < v < 8 \\ -8 + 8 \ln 2 + 2v/3 \ln 2 + 2v/3 + 4 \ln 3 & \\ \quad -4 \ln v + v/3 \ln 3 - v/3 \ln v & 8 < v < 12 \end{cases}.$$

Example 4.4.5 This example illustrates the fact that the product of two lognormal random variables has the lognormal distribution. Consider the random variable $X \sim \text{LogN}(\mu_1, \sigma_1^2)$ and the random variable $Y \sim \text{LogN}(\mu_2, \sigma_2^2)$. Find the PDF of $V = XY$. Here one sets $\mu_1 = \mu_2 = 0$, $\sigma_1^2 = 1$ and $\sigma_2^2 = 4$. The program yields the following PDF:

$$h(v) = \frac{e^{-(\ln v)^2/10}}{v\sqrt{10\pi}} \qquad 0 < v < \infty .$$

This can be identified as a $\text{LogN}(\mu = 0, \sigma^2 = 5)$ random variable.

Example 4.4.6 Consider the random variable $X \sim \text{N}(0, 1)$ and the random variable $Y \sim \text{N}(0, 1)$. Find the PDF of $V = XY$. This will test the case where the support of X and Y include the endpoints $\pm\infty$.

The program yields the following PDF for V:

$$h(v) = \begin{cases} \dfrac{\text{BesselK}(0,\ v \cdot \text{signum}(v))}{\pi} & -\infty < v < 0 \\ \dfrac{\text{BesselK}(0,\ v \cdot \text{signum}(v))}{\pi} & 0 < v < \infty \end{cases},$$

which relies on Maple's `BesselK` and `signum` functions. The Maple output of this function's plot is given in Figure 4.3.

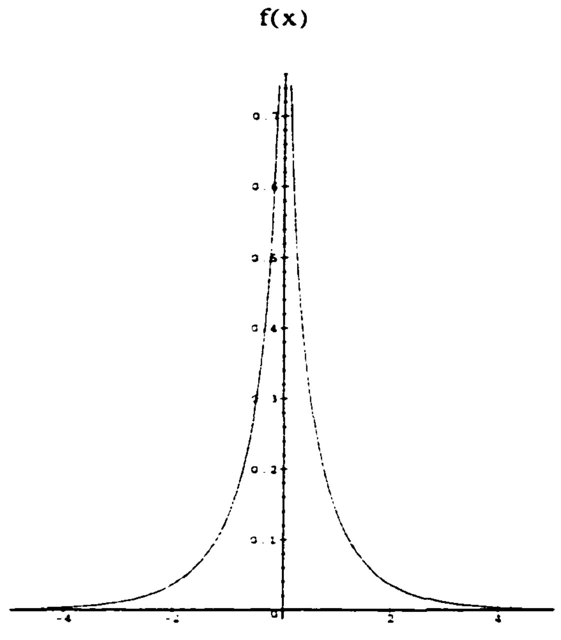


Figure 4.3: The PDF of $V = XY$ for $X \sim$ N(0, 1) and $Y \sim$ N(0, 1).

Example 4.4.7 Consider the independent random variables $U_1 \sim$ U(0, 1) and $U_2 \sim$ U(0, 1). The Box-Muller algorithm for generating a single standard normal deviate V can be coded in one line (Devroye, 1996) as

$$V \leftarrow \sqrt{-2\ln U_1}\cos(2\pi U_2),$$

where U_1 and U_2 are independent random numbers. Using the `Transform` and `ProductRV` procedures together, one can determine the PDF of V. Due to the principle inverse difficulty with trigonometric functions, however, the transformation must be rewritten as

$$V \leftarrow \sqrt{-2\ln U_1}\cos(\pi U_2)$$

before using `Transform`.

The program yields the following PDF for V:

$$h(v) = \begin{cases} \dfrac{v}{\pi}\displaystyle\int_{-1}^{0} \dfrac{e^{-v^2/(2x^2)}}{\sqrt{1-x^2}\,x^2}\,dx & -\infty < v < 0 \\ \dfrac{v}{\pi}\displaystyle\int_{0}^{1} \dfrac{e^{-v^2/(2x^2)}}{\sqrt{1-x^2}\,x^2}\,dx & 0 < v < \infty \end{cases}.$$

While this form in not easily recognizable as the PDF for the normal distribution, it is mathematically equivalent to the more standard

$$h(v) = \frac{1}{\sqrt{2\pi}}e^{-v^2/2} \qquad -\infty < v < \infty.$$

One anticipates that future generations of computer algebra systems will be able to simplify these integrals.

Example 4.4.8 Devroye (1996) gives

$$X \leftarrow m + (a + (b - a)U_1 - m)\max\{U_2, U_3\}$$

as a one-line algorithm for generating a triangular(a, m, b) variate, where U_1, U_2, and U_3 are independent and identically distributed U(0, 1) random variables and the triangular distribution has a minimum of a, a mode of m, and a maximum of b. One may now use this relationship to generate the PDF of a random variable with the triangular distribution. Using the Maple procedure **MaximumRV**, which returns the PDF of the maximum of two independent random variables and **Transform** for the linear transformations, one can derive the PDF with the following Maple commands.

```
a  := 1;
m  := 2;
b  := 3;
U1 := UniformRV(0, 1);
U2 := UniformRV(0, 1);
U3 := UniformRV(0, 1);
T1 := Transform(U1, [[x -> a + (b - a) * x - m],
      [- infinity, infinity]]);
T2 := MaximumRV(U2, U3);
T3 := ProductRV(T1, T2);
X  := Transform(T3, [[x -> m + x],[- infinity, infinity]]);
```

The resulting PDF for X is:

$$f(x) = \begin{cases} x - 1 & 1 < x < 2 \\ 3 - x & 2 < v < 3 \end{cases}.$$

4.5 Conclusion

An algorithm for calculating the PDF of the product of two independent random variables X and Y (which may be defined in a piece-wise manner) has been developed and implemented. The APPL procedure `ProductRV` is one of many procedures capable of automating complicated probability calculations associated with random variables. Potential application areas for calculations of this type include applied statistics, biostatistics, goodness-of-fit, probabilistic modeling, renewal theory, reliability, simulation, and time series analysis.

Chapter 5

Computing the CDF of the Kolmogorov–Smirnov Test Statistic

5.1 Introduction

Another probabilistic problem that is well suited for computer algebra systems is finding the distribution functions of the Kolmogorov–Smirnov test statistic D_n in the all-parameters-known case. Birnbaum (1952) gives an n-fold integral for the CDF of the test statistic which yields a function defined in a piecewise fashion, where each piece is a polynomial of degree n. Unfortunately, it is difficult to determine the appropriate limits of integration for computing these polynomials. The algorithm developed here (see Appendix G) performs the required integrations in a manner that avoids calculating the same integrals repeatedly, resulting in shorter computation time. It can be used to compute the entire CDF or just a portion of the CDF, which is more efficient for finding a critical value or a p-value associated with a hypothesis test. This approach yields exact critical values and significance levels.

The Kolmogorov–Smirnov (KS) test statistic D_n, is defined by

$$D_n = \sup_x |F(x) - F_n(x)|,$$

where n is the sample size, $F(x)$ is a hypothesized CDF with fixed parameters, and $F_n(x)$, also called the empirical distribution function, is a step-function that increases by $1/n$ at each data value. This statistic has been used for goodness-of-fit testing for continuous populations for decades. The KS test's appeal is its straightforward computation of the test statistic and the distribution-free characteristic of D_n. Its drawback is that its cumulative distribution function under the null hypothesis is difficult to determine, leaving one to calculate critical values with various approximation methods. Consider the distribution of the KS statistic in the case when all parameters are known.

Birnbaum (1952) gives the CDF of $D_n - \frac{1}{2n}$ as

$$P\left(D_n < \frac{1}{2n} + v\right) = n! \int_{\frac{1}{2n}-v}^{\frac{1}{2n}+v} \int_{\frac{3}{2n}-v}^{\frac{3}{2n}+v} \cdots \int_{\frac{2n-1}{2n}-v}^{\frac{2n-1}{2n}+v} g(u_1, u_2, \ldots, u_n)\, du_n \ldots du_2\, du_1$$

for $0 \le v \le \frac{2n-1}{2n}$, where

$$g(u_1, u_2, \ldots, u_n) = 1$$

for $0 \leq u_1 \leq u_2 \leq \cdots \leq u_n \leq 1$, and is 0 otherwise. Equivalently,

$$P\left(D_n < \frac{1}{2n} + v\right) = n! \int_{\max(0,\frac{1}{2n}-v)}^{\min(1,\frac{1}{2n}+v)} \int_{\max(u_1,\frac{3}{2n}-v)}^{\min(1,\frac{3}{2n}+v)} \cdots \int_{\max(u_{n-1},\frac{2n-1}{2n}-v)}^{\min(1,\frac{2n-1}{2n}+v)} 1\, du_n \ldots du_2\, du_1.$$

Birnbaum's rather harmless-looking integral is tedious to compute by hand, even for small values of n, due to the complexity of the region where g is nonzero. Evaluating his expression requires calculating many n-fold integrals whose limits are determined by considering a carefully chosen partition of the support of D_n. The difficult part of the process is to set the appropriate limits on these integrals. The final expression for the KS CDF is a piecewise polynomial function in which each polynomial has degree n.

5.2 Literature review

The literature available on the KS statistic is extensive. Stephens' article (Chapter 4 of D'Agostino and Stephens, 1986) contains comprehensive coverage on the use of the KS statistic, as well as other statistics based on the empirical distribution function. He calculates the power of these goodness-of-fit tests. Johnson, Kotz, and Balakrishnan (1995, p. 640) consider this source (D'Agostino, 1986) to be so complete that they have deleted KS discussions in their second edition and refer the reader to that compendium instead. For computing critical points of the KS distribution, one finds five-digit accuracy as early as 1956 (Miller, 1956). Miller relies on asymptotic results that converge fairly quickly to produce these estimates. Birnbaum's article

(1952) also explains how various sets of recursive formulas can be used to calculate certain critical points to reasonable levels of accuracy. Law and Kelton (1991, p. 387) indicate that critical points require computer techniques, and are only easily calculated for $n \le 50$. In the literature there appears to be no source that produces exact distribution functions for any distribution where $n > 3$. Birnbaum (1952, p. 441) gives the CDF of $D_n - \frac{1}{2n}$ for $n = 2$ and $n = 3$. Knuth (1981) provides a functional form for the CDFs for the two statistics D_n^+ and D_n^-, but does not provide the CDF for $D_n = \max\{D_n^+, D_n^-\}$, a harder problem given the dependence between the two random variables. As a consequence of the apparent complexity of the required integration and lack of literature on exact distributions, it was apparent early on that a computational algebra system, such as Maple, would be necessary to compute the polynomials needed for the distribution of the KS test statistic.

5.3 Computing the distribution of D_n

When $0 < v < \frac{1}{2n}$, Birnbaum's integral is easy to compute since none of the intervals of integration overlap. Additionally, these intervals are all wholly contained within the interval from 0 to 1. For example, u_1 varies over the interval from $\frac{1}{2n} - v$ to $\frac{1}{2n} + v$, and u_2 varies over the interval from $\frac{3}{2n} - v$ to $\frac{3}{2n} + v$. The interval for u_1 begins to the right of 0 and ends before the interval for u_2 begins. Thus when $0 < v < \frac{1}{2n}$, the limits of integration of Birnbaum's integral guarantee that $0 \le u_1 \le u_2 \le \cdots \le u_n \le 1$. Because the u_i's are constrained in this way, one may replace Birnbaum's integrand $g(u_1, u_2, \ldots, u_n)$ with 1, and thus computation of the KS CDF requires only a single

n-fold integral:

$$P\left(D_n < \frac{1}{2n} + v\right) = n! \int_{\frac{1}{2n}-v}^{\frac{1}{2n}+v} \int_{\frac{3}{2n}-v}^{\frac{3}{2n}+v} \cdots \int_{\frac{2n-1}{2n}-v}^{\frac{2n-1}{2n}+v} 1\, du_n \ldots du_2\, du_1 = n!(2v)^n$$

for $0 < v < \frac{1}{2n}$.

When v has a fixed value greater than $\frac{1}{2n}$, it is also desirable to replace Birnbaum's integrand by 1. In order to justify this replacement, one must only allow integration over that region of n-dimensional space for which $0 \le u_1 \le u_2 \le \cdots \le u_n \le 1$. Since the intervals of integration specified in Birnbaum's integral for different u_i's can overlap, the smallest allowable value for any u_i is influenced by all u_j's with subscripts less than i that can take on values in u_i's interval of integration. This overlapping requires partitioning the interval from $\frac{1}{2n} - v$ to $\frac{2n-1}{2n} + v$ into subintervals (which are henceforth refered to as u-subintervals), with the first u-subinterval starting at $\frac{1}{2n} - v$ and a new u-subinterval beginning whenever the left endpoint of one of Birnbaum's intervals of integration is encountered. When any u_i lies in a u-subinterval that consists entirely of negative values, Birnbaum's integrand is zero. For this reason, only u-subintervals that have a positive right endpoint contribute to the KS CDF.

Of course, the number of u-subintervals that have a positive right endpoint depends on the value of v. For small values of v (i.e., $0 < v < \frac{1}{2n}$), $n + 1$ u-subintervals have a positive right endpoint since all n of Birnbaum's lower limits of integration are positive. For large values of v (i.e., $\frac{2n-3}{2n} < v < \frac{2n-1}{2n}$), only two u-subintervals have a positive right endpoint since only u_n's lower limit of integration is positive.

Intermediate values of v produce all possibilities from 2 to $n+1$ u-subintervals that have a positive right endpoint.

Because different values of v can yield different numbers of u-subintervals that have a positive right endpoint, the interval $0 < v < \frac{2n-1}{2n}$ must be subdivided at the following values of v: $\frac{1}{2n}, \frac{3}{2n}, \frac{5}{2n}, \ldots, \frac{2n-3}{2n}$. When the values of v remain within one of the resulting subintervals for v, the number of u-subintervals that have a positive right endpoint will remain fixed.

Another complication arises because it is necessary to know, for a fixed value of v and for the u-subintervals produced by this value, which variables of integration u_i can take on values in each of the u-subintervals of $[0,1]$. The previous subdivision of the values of v is not fine enough to allow unambiguous specification of the range for each u_i. When $i < j$, u_i, and u_j have overlapping intervals of integration if the upper integration limit for u_i exceeds the lower integration limit for u_j, i.e., $\frac{2i-1}{2n} + v \geq \frac{2j-1}{2n} - v$. As a result, as v increases from 0 to $\frac{2n-1}{2n}$, new overlaps take place when v equals $\frac{1}{2n}, \frac{2}{2n}, \frac{3}{2n}, \ldots, \frac{n-1}{2n}$. The interval $0 < v < \frac{2n-1}{2n}$ must be divided into subintervals at these values of v as well as at the values of v listed previously. Henceforth these are refered to as v-subintervals.

Indicator matrices will be used to summarize the interrelationships between the possible values for the variables of integration $u_1, u_2, \ldots, u_n$. For a fixed n and for values of v in the kth v-subinterval of $\left(0, \frac{2n-1}{2n}\right)$, the indicator matrix A_k will show, by the presence of a 1 or a 0 in row i and column j, whether or not u_i can take on values in the jth u-subinterval. Finally, by defining paths through these indicator matrices, one produces a complete set of n-fold integrals which satisfy the requirement $0 \leq u_1 \leq u_2 \leq \cdots \leq u_n \leq 1$, and which produce the KS CDF when summed.

The algorithm for computing the cumulative distribution function of D_n is divided into four phases. In Phase 1, an appropriate partition of the support of $D_n - \frac{1}{2n}$ is determined. In Phase 2, matrices A_k are defined that are instrumental in determining the limits of integration in the n-fold integrals. In Phase 3, these integrals and their associated limits are computed. To take advantage of the similarities in the limits of integration of these n-fold integrals, they are grouped for efficiency and all evaluated to the same level of integration before proceeding to the next level. Finally, Phase 4 consists of translating the support of $D_n - \frac{1}{2n}$ to the support of D_n. Thus the input to the algorithm is a positive integer n and the output is the piecewise CDF of D_n.

5.3.1 Phase 1: Partition the support of $D_n - \frac{1}{2n}$

The endpoints of the segments that define the support of the KS test statistic can be determined from the limits of integration given by Birnbaum. Using Birnbaum's formula, the *baseline* lower and upper limits of integration associated with $v = 0$ are

$$\left\{ \frac{1}{2n}, \frac{3}{2n}, \frac{5}{2n}, \ldots, \frac{2n-1}{2n} \right\}.$$

As v increases, the support of $D_n - \frac{1}{2n}$ is broken into disjoint v-subintervals. The endpoints of these support v-subintervals consist of $v = 0$ and the values of v for which the endpoints of the n intervals of integration either:

- equal 0 or 1, *or*
- coincide.

The 0 or 1 cases occur at v values in the set

$$\left\{\frac{1}{2n}, \frac{3}{2n}, \frac{5}{2n}, \ldots, \frac{2n-1}{2n}\right\}.$$

The set of values of v for which the endpoints of different intervals of integration coincide are

$$\left\{\frac{1}{2n}, \frac{2}{2n}, \frac{3}{2n}, \ldots, \frac{n-1}{2n}\right\}.$$

Thus the union of these two sets and 0 comprise the endpoints of the v-subintervals of the support of $D_n - \frac{1}{2n}$.

The first phase of the algorithm given in Appendix G computes the above endpoints $v_0 = 0, v_1, v_2, \ldots, v_m$, where m is the number of v-subintervals on which the CDF of $D_n - \frac{1}{2n}$ is defined. For any n,

$$m = \left\lceil \frac{3n}{2} \right\rceil - 1.$$

5.3.2 Phase 2: Define the A matrices

At this point in the algorithm, $v_0, v_1, \ldots, v_m$ are the endpoints of the m v-subintervals on which the CDF of $D_n - \frac{1}{2n}$ is defined.

Two book-keeping steps are needed in this phase of the algorithm. They are:

1. Define $c_1, c_2, \ldots, c_m$ as the midpoints of the v-subintervals of support for $D_n - \frac{1}{2n}$.

2. Define $x_1, x_2, \ldots, x_n$ as midpoints of the intervals of integration for $u_1, u_2, \ldots, u_n$ in Birnbaum's n-fold integral. Thus

$$x_i = \frac{2i-1}{2n} \qquad i = 1, 2, \ldots, n.$$

Part of the algorithm involves defining $n \times n$ indicator matrices $A_1, A_2, \ldots, A_m$ corresponding to the v-subintervals which form the support of $D_n - \frac{1}{2n}$. The rows of an A matrix refer to the variables of integration $u_1, u_2, \ldots, u_n$. The columns of an A matrix refer to the u-subintervals, with the jth column corresponding to the interval from $x_j - v$ to $x_{j+1} - v$, except for the nth column, which corresponds to the interval from $x_n - v$ to $x_n + v$. If the (i, j) element of an A matrix equals 1, then the range of u_i includes at least part of the jth u-subinterval.

Since the (n, n) element of each A matrix is the only element in the nth row that is 1, the integrals that need to be computed for each segment of support of $D_n - \frac{1}{2n}$ can be visualized as a path in A, consisting of a sequence of moves from the (n, n) position of A to a nonzero element in the first row of A. All moves in the A matrix from row i to row $i - 1$ (for $i = n, n-1, \ldots, 2$) require that the following conditions are met:

1. The move must be from one of the 1's in row i to one of the 1's in row $i - 1$.

2. The move is either directly vertical or vertical and one or more spaces to the left.

That is, if the move begins at a 1 element in column j of row i, it must end at a 1 element in column 1 through j of row $i-1$.

Example 5.3.1 For $n = 3$ and $k = 3$, the CDF is defined on the v-subinterval $\frac{2}{6} < v < \frac{3}{6}$ and has the following A_3 matrix:

$$A_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

There are five different paths from the (3, 3) element to the top row of the matrix. In the five matrices below, these five paths to the top row of the A matrix are displayed by using boldface on the path. Next to each matrix is the corresponding triple integral that the path represents.

$$\begin{bmatrix} \mathbf{1} & 1 & 1 \\ 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} \end{bmatrix} \qquad \int_0^{\frac{3}{6}-v} \int_{\frac{3}{6}-v}^{\frac{5}{6}-v} \int_{\frac{5}{6}-v}^{1} 1 \, du_3 \, du_2 \, du_1$$

$$\begin{bmatrix} 1 & \mathbf{1} & 1 \\ 0 & \mathbf{1} & 1 \\ 0 & 0 & \mathbf{1} \end{bmatrix} \qquad \int_{\frac{3}{6}-v}^{\frac{5}{6}-v} \int_{u_1}^{\frac{5}{6}-v} \int_{\frac{5}{6}-v}^{1} 1 \, du_3 \, du_2 \, du_1$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \int_0^{\frac{3}{6}-v} \int_{\frac{5}{6}-v}^{\frac{3}{6}+v} \int_{u_2}^{1} 1 \, du_3 \, du_2 \, du_1$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \int_{\frac{3}{6}-v}^{\frac{5}{6}-v} \int_{\frac{5}{6}-v}^{\frac{3}{6}+v} \int_{u_2}^{1} 1 \, du_3 \, du_2 \, du_1$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \int_{\frac{5}{6}-v}^{\frac{1}{6}+v} \int_{u_1}^{\frac{3}{6}+v} \int_{u_2}^{1} 1 \, du_3 \, du_2 \, du_1$$

When these five integrals are summed, Birnbaum's expression reduces to

$$P\left(D_3 < \frac{1}{6} + v\right) = -4v^3 + \frac{11}{3}v - \frac{11}{27}, \qquad \frac{2}{6} < v < \frac{3}{6}.$$

Note that the inside integral

$$\int_{u_2}^{1} 1 \, du_3$$

on the third, fourth, and fifth triple integral is identical. The number of identical integrals of this type grows rapidly as n increases. The algorithm developed here avoids recalculation of duplicate integrals. Continuing in this fashion for the other A matrices (i.e., A_1, A_2, and A_4), yields the

CDF

$$P\left(D_3 < \frac{1}{6} + v\right) = \begin{cases} 48v^3 & 0 < v < \frac{1}{6} \\ -12v^3 + 8v^2 + v - \frac{1}{9} & \frac{1}{6} < v < \frac{2}{6} \\ -4v^3 + \frac{11}{3}v - \frac{11}{27} & \frac{2}{6} < v < \frac{3}{6} \\ 2v^3 - 5v^2 + \frac{25}{6}v - \frac{17}{108} & \frac{3}{6} < v < \frac{5}{6}. \end{cases}$$

Note that Birnbaum (1952) contains a sign error in the fourth v-subinterval of support.

Table 5.1 shows how rapidly the number of n-fold integrations required to compute the CDF of $D_n - \frac{1}{2n}$ grows with n. Note that the five triple integrals in the previous example corresponds to $n = 3$ and the third v-subinterval ($k = 3$), shown in boldface in the table. Note that the fifth column in Table 5.1 is the product of the first and fourth columns. The fourth and fifth columns did not match any well-known integer sequences in the literature (connect to

`http://www.research.att.com/~njas/sequences/index.html`).

Table 5.1: Computational requirements for computing the D_n CDF for small n.

n	m	Number of n-fold integrals required for each v-subinterval	Total number of n-fold integrals required	Total number of single integrals required
1	1	1	1	1
2	2	1, 2	3	6
3	4	1, 4, **5**, 3	13	39
4	5	1, 8, 13, 9, 4	35	140
5	7	1, 16, 34, 27, 28, 14, 5	125	625
6	8	1, 32, 89, 81, 89, 48, 20, 6	366	2196

Example 5.3.2 The method for determining the integers in the third column of the previous table is somewhat analogous to determining the entries in a row of Pascal's triangle by using the entries of the previous row. Beginning at each nonzero A matrix element in rows 2 through n, there are a certain number of paths whose moves to the top row of the matrix satisfy the two conditions stated previously. Assigned to such elements is a number p (for paths) that denotes the number of paths from that element to the top of the matrix. Thus if $p_{2,2} = 2$ (as is the case for the (2, 2) element of the A_3 matrix for $n = 3$ and $k = 3$ in Example 3.1) then no matter how one reached this element of the A matrix, there are only two possible paths remaining to the top. The $p_{i,j}$ values are assigned as follows. Only A matrix elements with value 1 are assigned a $p_{i,j}$ value. If the jth entry in the first row of A is nonzero, assign $p_{1,j} = 1$. For the second and subsequent rows $(i = 2, 3, \ldots, n)$, $p_{i,j} = \sum_{q=1}^{j} p_{i-1,q}$. Upon completion, $p_{n,n}$ represents the total number of paths from the lower right corner of the matrix to the top row of the matrix. Consider, for example, the case of $n = 5$ and $k = 4$. The A_4 matrix with superscripts denoting the $p_{i,j}$ values is

$$A_4 = \begin{bmatrix} 0 & 1^1 & 1^1 & 1^1 & 0 \\ 0 & 1^1 & 1^2 & 1^3 & 1^3 \\ 0 & 0 & 1^3 & 1^6 & 1^9 \\ 0 & 0 & 0 & 1^9 & 1^{18} \\ 0 & 0 & 0 & 0 & 1^{27} \end{bmatrix}.$$

Here one sees that $p_{5,5} = 27$, meaning there are 27 possible paths to the top row of A_4, consistent with the $n = 5$ and $k = 4$ element in Table 5.1. The general algorithm for computing the number of paths for any A-matrix (provided as input to the algorithm) is as follows:

```
$TotSum \leftarrow 0$
$p \leftarrow A$
For $i \leftarrow 2$ to $n$
    For $j \leftarrow i$ to $n$
        If $p_{i,j-1} = 0$
            $p_{i,j} \leftarrow p_{i-1,j-1} + p_{i-1,j}$
        else
            $p_{i,j} \leftarrow p_{i,j-1} + p_{i-1,j}$
$TotSum \leftarrow TotSum + p_{n,n}$
```

5.3.3 Phase 3: Set limits on the appropriate integrals

The indicator matrix A shows, by the presence of a 1 in row i and column j, for $j < n$, that u_i can assume values from the maximum of $x_j - v$ and 0 to the minimum of $x_{j+1} - v$ and $x_i + v$. The presence of a 1 in row i and column n, means that u_i assumes values from $x_n - v$ to the minimum of $x_i + v$ and 1. Each path in A, as described in Section 3.2, represents a set of allowable intervals of integration for the variables u_1 to u_n in a particular n-fold integral. For a particular path, the limits of integration for each u_i are those given above that correspond to entry $a_{i,j}$ of A in row i that lies on the path, with one exception: if both $a_{i,j}$ and $a_{i-1,j}$ are on the same path, the lower limit of integration for u_i must be u_{i-1}. This exception occurs because the path allows u_i and u_{i-1} to assume values in the same u-subinterval. In such a case, the lower limit of integration for u_i must be u_{i-1} in order to reflect the requirement $u_{i-1} \leq u_i$.

For each path through the matrix A, and for each nonzero entry $a_{i,j}$ on that path, a single integration with respect to u_i must be performed. If $a_{i-1,j}$ is on the same path as $a_{i,j}$, then the lower limit of integration will be the variable u_{i-1}. If $a_{i-1,j}$ is not on the same path as $a_{i,j}$, then the lower limit of integration will be a fixed number: the maximum of $x_j - v$ and 0. Thus for each path which passes through $a_{i,j}$, either a variable or fixed lower limit of integration might be appropriate.

Example 5.3.3 Consider again the case of $n = 5$ and the fourth v-subinterval ($k = 4$). The A_4 matrix shown below has its 1's replaced with $\mathcal{F}$, $\mathcal{V}$, or $\mathcal{B}$, indicating whether a fixed-limit integral, a variable-limit integral. or both need to be computed for each entry.

$$A_4 = \begin{bmatrix} 0 & \mathcal{F} & \mathcal{F} & \mathcal{F} & 0 \\ 0 & \mathcal{V} & \mathcal{B} & \mathcal{B} & \mathcal{F} \\ 0 & 0 & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ 0 & 0 & 0 & \mathcal{B} & \mathcal{B} \\ 0 & 0 & 0 & 0 & \mathcal{B} \end{bmatrix}.$$

In general. when the A matrix contains a zero, neither the fixed nor variable lower limits need to be computed. Now consider the elements of the A matrix that contain a 1. The positions associated with the first 1 in each column of A require only a fixed lower limit to be calculated. The positions below the first 1 in the first nonzero column require only a variable lower limit to be calculated. All other positions in the A matrix associated with 1 elements require both a fixed a variable lower limit to be calculated. Table 5.2 shows the computational efficiency of performing the integrations for various values of n by using the algorithm. The number of single integrals (algorithm) for any n is bounded above by

$$\frac{n(n+1)}{2} \cdot 2 \cdot \left(\left\lceil \frac{3n}{2} \right\rceil - 2 \right) \cong \frac{3}{2} n^3,$$

where the first factor corresponds to the maximum number of 1's in any A matrix, the second factor accounts for computing both the fixed and variable matrices (described subsequently), and the last factor is one less than m since no integration is required

Table 5.2: Computational efficiency associated with using the F and V arrays.

n	m	Total number of single integrals required (Table 5.1)	Total number of single integrals required (algorithm)
1	1	1	0
2	2	6	4
3	4	39	23
4	5	140	48
5	7	625	108
6	8	2196	170
10	14	442540	800
15	22	318612735	2793
20	29	232093052440	6400
30	44	1366173823664868000	21550
50	74	6105637995599418238684313815 0	99500

for the first v-subinterval. Due to savings associated with 0 columns in the A matrix and the fact that both fixed and variable matrix elements do not need to be calculated in all cases, this order n^3 expression is closely approximated by $0.80n^3$ for $n = 20$, 30, and 50.

Table 5.3 on page 101 lists the coefficients of the polynomials that define the CDF of $D_n - \frac{1}{2n}$ as computed by the algorithm in the Appendix which has been implemented in Maple for $n = 1, 2, \ldots, 6$. The ability to store fractions exactly is of particular benefit for these calculations. Rather than detail every aspect of the logic of Phase 3 of the algorithm, illustrated below is the evolution of the $n \times n$ matrices F (for fixed limits) and V (for variable limits) in a row-by-row fashion for a particular combination of n and k.

Example 5.3.4 Consider again the case of $n = 3$ and $k = 3$, which corresponds to the v-subinterval $\frac{2}{6} < v < \frac{3}{6}$. The x_i values associated with $n = 3$ are $x_1 = \frac{1}{6}$, $x_2 = \frac{3}{6}$, and $x_3 = \frac{5}{6}$. The v_i values associated with $n = 3$ are $v_0 = 0$, $v_1 = \frac{1}{6}$, $v_2 = \frac{2}{6}$, $v_3 = \frac{3}{6}$, and $v_4 = \frac{5}{6}$. The centers of the v-subintervals are $c_1 = \frac{1}{12}$, $c_2 = \frac{3}{12}$, $c_3 = \frac{5}{12}$, and $c_4 = \frac{8}{12}$. The A_3 matrix is

$$A_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

The F and V matrices are computed simultaneously, row by row, starting with the last row and ending with the first. Thus all of the elements of the third row of F and of V are computed before computing the elements of the second row is begun. In general, integrals which are calculated in a particular row become integrand candidates for the integration to be carried out in the row immediately above. The F and V matrices are designed to store the intermediate results of integration of the n-fold integrals so that the necessary inner integration operations are performed only once. Entries in the ith rows of F and V each result from $n + 1 - i$ single integrations, one integration corresponding to the ith row and one integration for every lower row. The F matrix stores the results of all integration in which the last integral had a fixed lower limit and the V matrix stores the results of all integration in which the last integral had a variable lower limit. In this example, for the inner-most integration variable u_3, there is only one possible fixed integration to compute and one possible variable integration to compute on the third u-subinterval. These computations are seen in the (3, 3) element of the respective matrices:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ 0 & F_{2,2} & F_{2,3} \\ 0 & 0 & \int_{\frac{5}{6}-v}^{1} 1 du_3 \end{bmatrix}, \qquad V = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ 0 & V_{2,2} & V_{2,3} \\ 0 & 0 & \int_{u_2}^{1} 1 du_3 \end{bmatrix}.$$

The second variable, u_2, has four possible combinations of limits and integrands. Thus there are four storage elements for u_2, found in the second rows of the F and V matrices. Elements (2, 2) and (2, 3) of the F matrix store integration results that have all fixed lower-limits of integration, but the integrands are fixed and variable, respectively. Similarly, elements (2, 2) and (2, 3) of the V matrix are for the cases when the lower-limit of integration contains the u_1 variable, while the integrands are fixed and variable, respectively. $F_{2,2}$ is computed to cover the case in which u_2 varies over the second u-subinterval and u_3 varies over the third u-subinterval. $F_{2,3}$ covers the case in which both u_2 and u_3 vary over the third u-subinterval, so u_3 must have u_2 as its variable lower limit; hence the integrand is $V_{3,3}$, not $F_{3,3}$. Similar cases are covered by $V_{2,2}$ and $V_{2,3}$, except that variable lower limits are used in anticipation of the fact that u_1 and u_2 can both vary over the same u-subinterval, making it necessary to have u_1 as the (variable) lower limit for u_2. The integrations in the second row of each matrix are shown below (note, the integration for the (3, 3) element, previously discussed, has been carried out).

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ 0 & \int_{\frac{1}{2}-v}^{\frac{5}{6}-v} F_{3,3}\,du_2 & \int_{\frac{5}{6}-v}^{\frac{1}{2}+v} V_{3,3}\,du_2 \\ 0 & 0 & \frac{1}{6}+v \end{bmatrix} .$$

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ 0 & \int_{u_1}^{\frac{5}{6}-v} F_{3,3}\,du_2 & \int_{u_1}^{\frac{1}{2}+v} V_{3,3}\,du_2 \\ 0 & 0 & 1-u_2 \end{bmatrix} .$$

For the third and last variable, u_1, only fixed-limit integration takes place so only the F matrix gets updated. The (1, 1) element covers the case in which u_1 varies over the first u-subinterval and u_2 varies over the second and third u-subinterval. The (1, 2) element covers the case in which u_1 varies over the second u-subinterval and u_2 varies over the second and third u-subinterval. The (1, 3) element allows both u_1 and u_2 to vary over the third u-subinterval. The integrals in this first row are shown below (note again, all previously discussed integration have been carried out in the second and third rows of F and V):

$$F = \begin{bmatrix} \int_0^{\frac{1}{2}-v} (F_{2,2}+F_{2,3})\,du_1 & \int_{\frac{1}{2}-v}^{\frac{5}{6}-v} (V_{2,2}+F_{2,3})\,du_1 & \int_{\frac{5}{6}-v}^{\frac{1}{6}+v} V_{2,3}\,du_1 \\ 0 & -\frac{1}{3}v+\frac{1}{18} & -\frac{1}{9}+\frac{2}{3}v \\ 0 & 0 & \frac{1}{6}+v \end{bmatrix} ,$$

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ 0 & -v^2 + \frac{2}{3}v + \frac{5}{36} - vu_1 - \frac{1}{6}u_1 & \frac{1}{2}v^2 + \frac{1}{2}v + \frac{3}{8} + \frac{1}{2}u_1^2 - u_1 \\ 0 & 0 & 1 - u_2 \end{bmatrix}.$$

The completely evaluated F matrix is given by

$$F = \begin{bmatrix} -v^2 + \frac{5}{9}v - \frac{1}{36} & -\frac{1}{36} + \frac{5}{18}v & -\frac{2}{3}v^3 + v^2 - \frac{2}{9}v - \frac{1}{81} \\ 0 & -\frac{1}{3}v + \frac{1}{18} & -\frac{1}{9} + \frac{2}{3}v \\ 0 & 0 & \frac{1}{6} + v \end{bmatrix}.$$

Finally, the CDF on the 3rd v-subinterval is the sum of the elements in the first row of the F matrix, all multiplied by $3! = 6$.

5.3.4 Phase 4: Shift the distribution

At this point, the CDF is computed in the form $P(D_n < v + \frac{1}{2n})$. Now one converts the distribution into the more usable form $P(D_n < y) = F_{D_n}(y)$ by making the substitution $y = v + \frac{1}{2n}$ in both the polynomials and the v-subintervals of the CDF. Specifically, one adds $\frac{1}{2n}$ to each endpoint of the v-subintervals and one substitutes $(y - \frac{1}{2n})$ for v in the CDF polynomials. One then takes these two lists of numbers and polynomials respectively, simplifies them (using Maple's `simplify()` command), and creates the CDF representation in the form of the "list-of-lists" representation for distributions outlined in Chapter 2. This enables us to use the distribution in the ways that all other distributions are used in APPL. Specifically, one can now verify critical values for the distribution. but more importantly, one can calculate exact significance levels of any given statistic. An example of the shifted distribution for $n = 6$ is:

$$F_{D_6}(y) = \begin{cases} 0 & y < \frac{1}{12} \\ 46080\,y^6 - 23040\,y^5 + 4800\,y^4 - \frac{1600}{3}\,y^3 + \frac{100}{3}\,y^2 - \frac{10}{9}\,y + \frac{5}{324} & \frac{1}{12} \le y < \frac{1}{6} \\ 2880\,y^6 - 4800\,y^5 + 2360\,y^4 - \frac{1280}{3}\,y^3 + \frac{235}{9}\,y^2 + \frac{10}{27}\,y - \frac{5}{81} & \frac{1}{6} \le y < \frac{1}{4} \\ 320\,y^6 + 320\,y^5 - \frac{2600}{3}\,y^4 + \frac{4240}{9}\,y^3 - \frac{785}{9}\,y^2 + \frac{145}{27}\,y - \frac{35}{1296} & \frac{1}{4} \le y < \frac{1}{3} \\ -280\,y^6 + 560\,y^5 - \frac{1115}{3}\,y^4 + \frac{515}{9}\,y^3 + \frac{1525}{54}\,y^2 - \frac{565}{81}\,y + \frac{5}{16} & \frac{1}{3} \le y < \frac{5}{12} \\ 104\,y^6 - 240\,y^5 + 295\,y^4 - \frac{1985}{9}\,y^3 + \frac{775}{9}\,y^2 - \frac{7645}{648}\,y + \frac{5}{16} & \frac{5}{12} \le y < \frac{1}{2} \\ -20\,y^6 + 32\,y^5 - \frac{185}{9}\,y^3 + \frac{175}{36}\,y^2 + \frac{3371}{648}\,y - 1 & \frac{1}{2} \le y < \frac{2}{3} \\ 10\,y^6 - 38\,y^5 + \frac{160}{3}\,y^4 - \frac{265}{9}\,y^3 - \frac{115}{108}\,y^2 + \frac{4651}{648}\,y - 1 & \frac{2}{3} \le y < \frac{5}{6} \\ -2\,y^6 + 12\,y^5 - 30\,y^4 + 40\,y^3 - 30\,y^2 + 12\,y - 1 & \frac{5}{6} \le y < 1 \\ 1 & y \ge 1 \end{cases}$$

The Maple `PlotDist` output of the CDF and PDF of D_6 are shown in Figure 5.1 and Figure 5.2. The commands used in Maple to produce these plots are:

```
Y := KSRV(6);
PlotDist(CDF(Y), 0, 1);
PlotDist(PDF(Y), 0, 1);
```

Note that there is a discontinuity in the PDF at $y = \frac{1}{6}$, and more generally, at $y = \frac{1}{n}$. This corresponds to the value $v = \frac{1}{2n}$ in Birnbaum's original integral. This is the smallest value of v for which the n-dimensional hypercube centered at

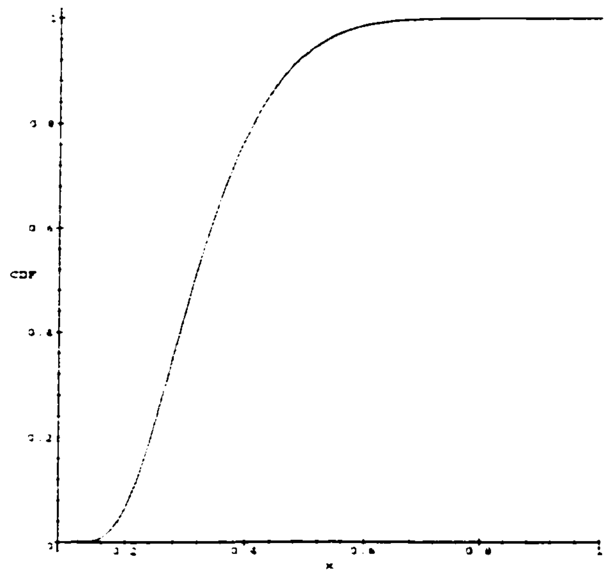


Figure 5.1: The CDF of the D_6 random variable.

$\left(\frac{1}{2n}, \frac{3}{2n}, \ldots, \frac{2n-1}{2n}\right)$ makes contact with the boundary of the region in n-space that satisfies $0 \leq u_1 \leq u_2 \leq \cdots \leq u_n \leq 1$.

The software is designed so that it stores the computed CDF for a particular value of n so that it does not need to be recomputed subsequently. This is of particular importance since Maple is an interpreted language, resulting in slow execution times. The CDF has be stored for $n = 1, 2, \ldots, 10$.

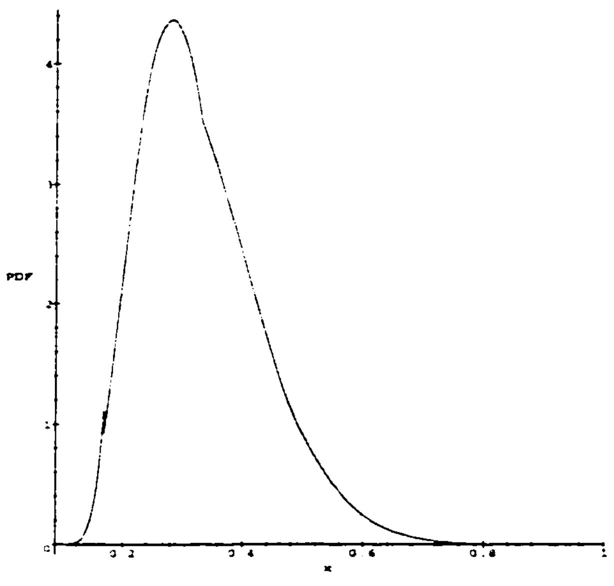


Figure 5.2: The PDF of the D_6 random variable. Note the discontinuity at $y = 1/6$.

5.4 Critical values and significance levels

With these exact distributions, one now can find exact critical values and significance levels. Critical values are determined with the IDF procedure. The program is able to achieve arbitrary accuracy using the Maple command Digits to define the precision. These critical values have been verified with those printed in Owen (1962) for $\alpha = 0.1, 0.05$, and 0.01 and for $n = 1, 2, \ldots, 10$ and those values match ours exactly, except that ours are not limited to five decimal places. More importantly, though, is the ability to produce exact attained significance levels given observed test statistics. These attained significance levels, or p-values, give more information about the statistical test of hypothesis than does the traditional "accept/reject" regions associated with critical values. Using the procedure `CDF()`, one can find these exact significance levels. For example, let $n = 6$ and the observed test statistic $D_6 = 0.5543$. Two critical values are, for example, 0.51963 for $\alpha = 0.05$ and 0.61661 for $\alpha = 0.01$. One can now determine the exact attained significance level of the test statistic with the command `1 - CDF(D6, 0.5543)`, which yields a p-value of 0.02928, a clearer indicator of the level of evidence against H_0. Note that it is not necessary to compute the entire D_n distribution to obtain a particular significance level. One must only determine which segment of support the statistic belongs to, then execute the algorithm for just that piece of the CDF. This shortcut will reduce the integral computation time by approximately $\frac{m-1}{m} \cdot 100\%$, where the exact savings depends on the v-subinterval that is chosen. Table 5.1 indicates that the greatest time savings will probably be in the tails of the distribution. This time savings is particularly important for larger values of n.

5.5 Conclusion

An algorithm for computing the CDF of the KS test statistic in the all-parameters-known case for a positive integer parameter n has been presented. Computing the CDF provides a challenging calculus problem, even for small values of n. The CDF can be constructed from a set of piecewise degree-n polynomials. The algorithm can be used to plot the CDF or to just find particular fractiles or probabilities.

Table 5.3: CDFs of $D_n - \frac{1}{2n}$ for $n = 1, 2, \ldots, 6$.

n	m	k	Coefficients of CDF polynomials	Subinterval
1	1	1	$2, 0$	$0 < v < \frac{1}{2}$
2	2	1	$8, 0, 0$	$0 < v < \frac{1}{4}$
		2	$-2, 3, -\frac{1}{8}$	$\frac{1}{4} < v < \frac{3}{4}$
3	4	1	$48, 0, 0, 0$	$0 < v < \frac{1}{6}$
		2	$-12, 8, 1, -\frac{1}{9}$	$\frac{1}{6} < v < \frac{2}{6}$
		3	$-4, 0, \frac{11}{3}, -\frac{11}{27}$	$\frac{2}{6} < v < \frac{3}{6}$
		4	$2, -5, \frac{25}{6}, -\frac{17}{108}$	$\frac{3}{6} < v < \frac{5}{6}$
4	5	1	$384, 0, 0, 0, 0$	$0 < v < \frac{1}{8}$
		2	$-48, 0, 15, -\frac{9}{8}, \frac{3}{256}$	$\frac{1}{8} < v < \frac{2}{8}$
		3	$16, -40, 21, -\frac{5}{8}, -\frac{29}{256}$	$\frac{2}{8} < v < \frac{3}{8}$
		4	$6, -7, -\frac{27}{16}, \frac{293}{64}, -\frac{853}{2048}$	$\frac{3}{8} < v < \frac{5}{8}$
		5	$-2, 7, -\frac{147}{16}, \frac{343}{64}, -\frac{353}{2048}$	$\frac{5}{8} < v < \frac{7}{8}$
5	7	1	$3840, 0, 0, 0, 0, 0$	$0 < v < \frac{1}{10}$
		2	$0, -288, \frac{624}{5}, -\frac{96}{25}, -\frac{36}{125}, \frac{6}{625}$	$\frac{1}{10} < v < \frac{2}{10}$
		3	$160, -160, \frac{24}{5}, \frac{616}{25}, -\frac{332}{125}, \frac{6}{125}$	$\frac{2}{10} < v < \frac{3}{10}$
		4	$-20, 64, -\frac{318}{5}, \frac{542}{25}, \frac{343}{500}, -\frac{273}{1250}$	$\frac{3}{10} < v < \frac{4}{10}$
		5	$12, 0, -\frac{62}{5}, \frac{6}{5}, \frac{2391}{500}, -\frac{3413}{6250}$	$\frac{4}{10} < v < \frac{5}{10}$
		6	$-8, 18, -\frac{52}{5}, -\frac{19}{5}, \frac{1838}{250}, -\frac{10527}{25000}$	$\frac{5}{10} < v < \frac{7}{10}$
		7	$2, -9, \frac{81}{5}, -\frac{729}{50}, \frac{6561}{1000}, -\frac{9049}{50000}$	$\frac{7}{10} < v < \frac{9}{10}$
6	8	1	$46080, 0, 0, 0, 0, 0, 0$	$0 < v < \frac{1}{12}$
		2	$2880, -3360, 660, 60, -\frac{95}{12}, \frac{5}{24}, -\frac{5}{5184}$	$\frac{1}{12} < v < \frac{2}{12}$
		3	$320, 480, -700, \frac{5620}{27}, -\frac{125}{36}, -\frac{275}{216}, \frac{2195}{46656}$	$\frac{2}{12} < v < \frac{3}{12}$
		4	$-280, 420, -\frac{335}{2}, -\frac{1675}{54}, \frac{26005}{864}, -\frac{3125}{1728}, -\frac{20645}{373248}$	$\frac{3}{12} < v < \frac{4}{12}$
		5	$104, -188, \frac{1235}{6}, -\frac{7435}{54}, \frac{36245}{864}, -\frac{7327}{5184}, -\frac{69797}{373248}$	$\frac{4}{12} < v < \frac{5}{12}$
		6	$-20, 22, \frac{45}{4}, -\frac{2005}{108}, -\frac{185}{1728}, \frac{57971}{10368}, -\frac{406469}{746496}$	$\frac{5}{12} < v < \frac{7}{12}$
		7	$10, -33, \frac{925}{24}, -\frac{3065}{216}, -\frac{22175}{3456}, \frac{134807}{20736}, -\frac{632863}{1492992}$	$\frac{7}{12} < v < \frac{9}{12}$
		8	$-2, 11, -\frac{605}{24}, \frac{6655}{216}, -\frac{73205}{3456}, \frac{161051}{20736}, -\frac{278569}{1492992}$	$\frac{9}{12} < v < \frac{11}{12}$

Chapter 6

Goodness of Fit using Order Statistics

6.1 Introduction

Presented in this chapter is a detailed application of APPL in which a new goodness-of-fit test statistic is devised. The inception and implementation of this test is possible primarily due to the existence of APPL. This application highlights how the paradigm of problem solving shifts with the advent of a probability language. Presented here is a new method of using order statistics for judging the fit of data by a hypothesized distribution. There is a variety of uses of order statistics in statistical inference procedures. One reason is the potential for adapting such procedures for use with censored samples (Cox and Oakes, 1984 and David, 1993). Other applications concern behavior in the tails of a parent distribution (for example, outlier tests, Tietjen, 1986), situations in which the ordered data are collected over time (life testing, for example, Barlow and Proschan, 1975), and estimation of parameters in terms of linear combinations of order statistics (David, 1993). A large literature on the subject exists (see, for example David, 1981 and references cited therein).

One important class of problems involving order statistics is goodness-of-fit testing (D'Agostino and Stephens, 1986). The Shapiro–Wilks normality test (Johnson and Wichern, 1992, p. 158) is based on a statistic that uses, in part, the expected value of each order statistic. Since many of the tests commonly used are defined through the sample distribution function, they may be considered to be functions of the underlying order statistics. It is common practice to transform order statistics from a parent population for the continuous random variable X with cumulative distribution function F to corresponding U(0, 1) order statistics, through the probability integral transformation, $U = F(X)$. If the proposed test statistic is invariant under this transformation, it follows that its distribution may be derived for the case of a U(0, 1) parent distribution, and applied for any other continuous parent distribution. That is, the procedure in this case is "distribution free."

An example of the foregoing is the Kolmogorov–Smirnov (K–S) one-sample test, based on the test statistic $D_n = \sup_x |F(x) - F_n(x)|$, where F_n is the sample distribution function. Since the L_∞ norm is invariant under transformations on x, it follows that one may find critical values for D_n using the distribution of $\max_i\{\max\{|u_i - \frac{i-1}{n}|, |u_i - \frac{i}{n}|\}\}$, where $u_i = F(x_{(i)}), i = 1, 2, \ldots, n$. Following the development of the K–S test, many articles adapted the test to related order statistic-based tests:

tests for data on a circle (Kuiper, 1960 and Stephens, 1965), tests with parameters estimated from data (Lilliefors, 1967 and 1969), tests for censored samples (Barr and Davidson, 1973), and others. In addition to its use in non-parametric goodness-of-fit testing, the K–S procedure may be used in a general way to obtain confidence regions for parameter vectors (Barr and Davidson, 1973); interestingly, the latter is a parametric problem.

In Section 6.2 a vector of U(0, 1) statistics is defined by transforming the order statistics through each of the n hypothesized CDFs. Also suggested are possible uses of the vector in a variety of statistical problems, and it is shown how the marginal distributions of its components can be computed with mathematical software such as Maple. These ideas are applied to define a goodness-of-fit statistic and simulation is used to show it is admissible relative to the K–S and Anderson–Darling tests in Section 6.3. Possible extensions and conclusions are given in Section 6.4.

6.2 The $\mathcal{P}$-vector

Proposed here is a measure of the goodness-of-fit of empirical data by a hypothesized distribution based on the vector of real numbers called the $\mathcal{P}$-vector. Let $X_1, X_2, \ldots, X_n$ be a random sample from a population with continuous CDF F, and let $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ be the corresponding order statistics. Then the individual order statistics have marginal densities $f_{X_{(1)}}(x), f_{X_{(2)}}(x), \ldots, f_{X_{(n)}}(x)$ given by

$$f_{X_{(i)}}(x) = \frac{n!}{(i-1)!(n-i)!}[F_X(x)]^{i-1}[1-F_X(x)]^{n-i}f_X(x),$$

where $f_X(x) = F'_X(x)$ (see Larsen and Marx, 1986, for example). In principle, one can use these distributions to determine the quantiles for each ordered observation in its respective distribution. Define the $\mathcal{P}$-vector to be the n-vector

$$\mathcal{P} = [F_{X_{(1)}}(x_{(1)}), F_{X_{(2)}}(x_{(2)}), \ldots, F_{X_{(n)}}(x_{(n)})].$$

For simplicity in notation, let $p_i = F_{X_{(i)}}(x_{(i)}), i = 1, 2, \ldots, n$. Intuitively, poor fit is indicated by "extreme" components in $\mathcal{P}$. Thus, under the null hypothesis H_0: X has CDF $F_X(x)$ with $n = 3$ observations, for example, a $\mathcal{P}$-vector of $[0.453, 0.267, 0.623]$ intuitively indicates a good fit more so than a $\mathcal{P}$-vector of $[0.001, 0.005, 0.997]$.

Since the individual elements p_i are the result of the probability integral transformation, it is clear that $P_i \sim U(0,1), i = 1, 2, \ldots, n$, for any continuous population CDF F. While the P_i are identically distributed, they are not stochastically independent. There is positive autocorrelation amongst elements of the $\mathcal{P}$-vector.

The $\mathcal{P}$-vector has a number of potential uses as a basis for distribution-free statistical procedures. It can provide a distribution-free tool to identify outliers. Tietjen (1986, p. 497) mentions in his paper on outliers "We shall discuss here only the underlying assumption of normality since there is very little theory for any other case". As is discuss below, it is possible and practical to calculate all p_i for nearly any hypothesized continuous distribution, so one may use the $\mathcal{P}$-vector as a basis to identify outliers from any specified continuous distribution. As one approach, one could examine the first and last few elements of $\mathcal{P}$ (p_1 and p_n, for example) to determine whether they are statistically significant. Also envisioned are uses of the $\mathcal{P}$-vector for statistical inferences involving censored sampling and estimation based on "order agreement." The main application investigated here is goodness-of-fit testing using test statistics based on the $\mathcal{P}$-vector.

6.3 Improving computation of $\mathcal{P}$

Computing the elements of the $\mathcal{P}$-vector can be accomplished with APPL. The procedure `OrderStat(X, n, r)` for example, determines the distribution of the r^{th} out of n order statistics. Combined with the procedure `CDF(X, x)`, which returns $F_X(x)$, the elements p_i are calculated.

It is preferable to calculate the $\mathcal{P}$-vector by first transforming to U(0, 1) order statistics, and then determining the quantiles p_i using corresponding beta$(i, n - i + 1)$ CDFs, which are relatively simple polynomials as indicated in Figure 6.1. The calculation of $p_i = F_{X_{(i)}}(x_{(i)})$ as first described is depicted by the path shown by a solid line in Figure 6.1. This method of computation relies on the ability to calculate quantiles in distributions of all of the order statistics $X_{(i)}$, although recurrence relations for CDFs of order statistics might speed computation (David, 1993). The second approach, using beta CDFs, is depicted by the dashed line in Figure 6.1. It requires the

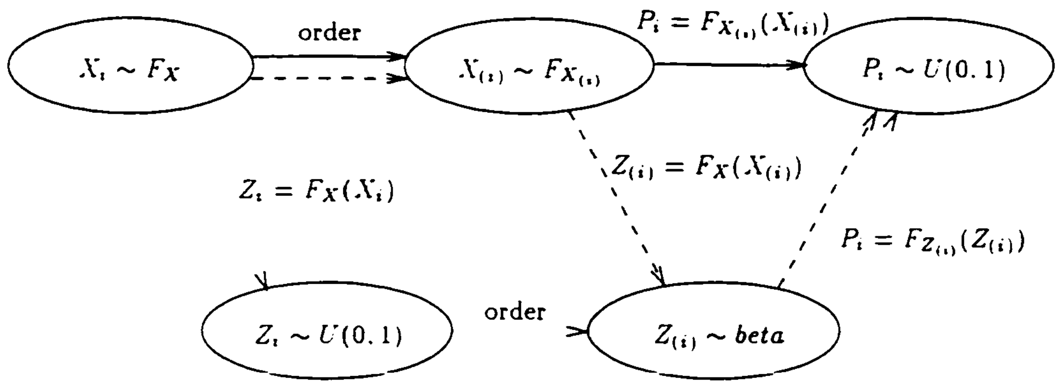


Figure 6.1: Transformations from iid observations $X_1, X_2, \dots, X_n$ to the sorted $\mathcal{P}$-vector elements $P_{(1)}, P_{(2)}, \dots, P_{(n)}$.

transformation of the $x_{(i)}$'s into U(0, 1) random variables and then the determines of the quantiles of the latter using appropriate beta CDFs. The theorem to follow proves that both methods are equivalent.

It is intuitively plausible that the quantiles computed following the two paths in Figure 6.1 will be identical; however, it is useful to provide a careful demonstration of this fact. Since the definition is $p_i = F_{X_{(i)}}(x_{(i)})$, it will suffice to show that $F_{Z_{(i)}}(z_{(i)}) = F_{X_{(i)}}(x_{(i)})$, where $Z = F_X(X)$.

Theorem. Let $X_1, X_2, \dots, X_n$ be a random sample from a population with continuous CDF F_X and probability density function f_X. Suppose $X_{(1)}, X_{(2)}, \dots, X_{(n)}$ are the corresponding order statistics, and let $Z_{(i)} = F_X(X_{(i)})$. Then,

$$F_{Z_{(i)}}(z_{(i)}) = F_{X_{(i)}}(x_{(i)}); \; i = 1, 2, \dots, n. \tag{1}$$

Proof. Both sides of Equation (1) will be reduced to equivalent forms. For the left-hand side of Equation (1), note that $Z_{(i)} \sim$ beta$(i, n - i + 1)$ (Barr and Zehna, 1983) and that

$$F_{Z_{(i)}}(z_{(i)}) = F_{Z_{(i)}}(F_X(x_{(i)})) = F_{Z_{(i)}} \circ F_X(x_{(i)}).$$

The CDF of a beta$(i, n - i + 1)$ random variable is defined by

$$F_{Z_{(i)}}(x) = k \int_0^x z^{i-1}(1 - z)^{n-i} dz,$$

where $k = \dfrac{n!}{(i-1)!(n-i)!}$. Thus the composition above is given by

$$F_{Z_{(i)}} \circ F_X(x_{(i)}) = k \int_0^{F_X(x_{(i)})} z^{i-1}(1 - z)^{n-i} dz.$$

Now consider the right-hand side of Equation (1). The density of $X_{(i)}$ is given by

$$f_{X_{(i)}}(x) = k[F_X(x)]^{i-1}[1 - F_X(x)]^{n-i} f_X(x)$$

(Larsen and Marx, 1986), so the corresponding CDF is

$$F_{X_{(i)}}(x_{(i)}) = k \int_{-\infty}^{x_{(i)}} [F_X(s)]^{i-1}[(1 - F_X(s))]^{n-i} f_X(s) ds.$$

Make the change of variable $z = F_X(s)$ so that $-\infty < s \leq x_{(i)} \Leftrightarrow 0 \leq z \leq F_X(x_{(i)})$

and $dz = f_X(s)ds$. The right-hand side of Equation (1) is therefore given by

$$F_{X_{(i)}}(x_{(i)}) = k\int_0^{F_X(x_{(i)})} z^{i-1}(1-z)^{n-i}dz = F_{Z_{(i)}} \circ F_X(x_{(i)})$$

□

One could also develop the quantile p_i following the lower path in Figure 6.1. The path indicated in Figure 6.1 by the dashed line is generally preferred, since the distributions leading to the p_i elements are polynomials. The computations needed for the solid path are calculable, however they typically take a significantly longer time to calculate. Also, the CDF of X is typically more tractable than the CDFs of $X_{(i)}$'s.

6.4 Goodness-of-fit testing

In general, goodness-of-fit testing involves stating a null hypothesis, $H_0 : f_X(x;\theta) = f_0(x;\theta_0)$ and then assessing whether there is sufficient evidence in a random sample to reject H_0. Since the $\mathcal{P}$-vector was derived under H_0, one sees that its elements provide a direct judgment of fit. Any p_i too low or too high may be an indication of a poor fit. It would be useful to form a single statistic based on the p_i's that indicate good or bad fit. Some obvious candidate test statistics are $\sum_{i=1}^{n} |p_i - 0.5|$, $\sum_{i=1}^{n}(p_i - 0.5)^2$ and $\sum_{i=1}^{n} |p_{(i)} - \frac{i}{n+1}|$. These test statistics, however, appear to suffer from low power, based on a modest set of simulations that were conducted. Instead, a variation of the reasoning that seems implicit in the development of Anderson–Darling (A–D) A^2 statistic was considered. Calculations are done according to the probability integral transformation presented in the second form in Stephens (1986, p. 101). Note that the A–D A^2 statistic is developed with the $Z_{(i)}$ values which are depicted in the lower-right state of Figure 6.1. Define a test statistic in terms of a linear combination of the natural logarithms of p_i and $1 - p_i$. This test statistic will be large whenever one or more p_i's is too close to 1 or 0. Sort the elements of the $\mathcal{P}$-vector, so $p_{(1)}$ is the smallest of the elements of the $\mathcal{P}$-vector and $p_{(n)}$ is the largest. Define the test statistic P_s by

$$P_s = -n - \frac{1}{n}\sum_{i=1}^{n}[(2n + 1 - 2i)\ln(p_{(i)}) + (2i - 1)\ln(1 - p_{(i)})]. \qquad (2)$$

The power of P_s was examined in a series of simulations. The goodness-of-fit tests based on P_s have power about that of A^2 in most cases, and both P_s and A^2 generally dominate the K–S test, as stated in D'Agostino (1986, p. 110), for the A–D test. There is at least one case where P_s dominates the A–D test: the case of guarding against an improper variance parameter under the null and alternate hypotheses of normality.

The test using P_s as a test statistic is significantly more powerful than both the K–S and A–D tests in the following experiment. The power of the three tests was approximated, and a fourth test P_s^* using p_i in place of $p_{(i)}$ in Equation (2), under the following conditions. Let $H_0 : X \sim \mathrm{N}(0,1)$ and $H_a : X \sim N(0, \sigma^2)$. Random samples were generated of size 10 from $N(0, \sigma^2)$ where σ varied from $0.1, 0.2, \ldots, 3.0$. A plot of each test's estimated power at guarding against changes in σ, based on 1000 replications of the simulation, is shown in Figure 6.2.

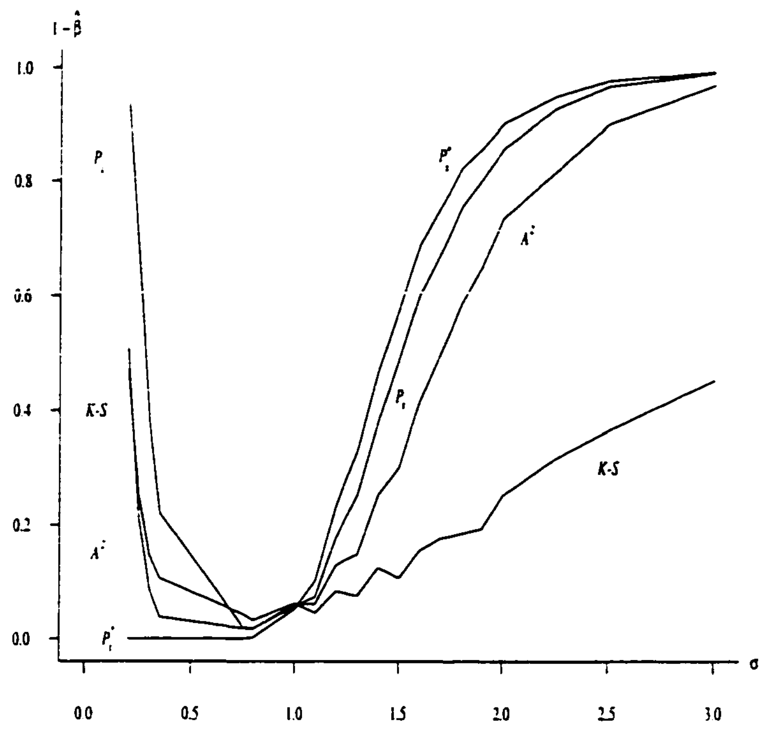


Figure 6.2: Estimated power functions for testing H_0: $X \sim$ N(0, 1) versus H_1: $X \sim$ N(0, σ^2) using K-S, A-D, and two statistics based on the $\mathcal{P}$-vector.

These power plots show that as σ becomes greater than 1, both P_s and P_s^* have significantly higher power than have A^2 and D_n. Also, P_s^* slightly outperforms P_s for $\sigma > 1$. As σ becomes less than 1, P_s clearly outperforms A^2 and D_n, but P_s^* has very low power. The strong performance of P_s, four times the power of A^2 at $\sigma = 0.30$, for example, causes us to conjecture that this is a strong omnibus test statistic for guarding against departures from hypothesized variance for $N(0, \sigma^2)$ populations. The standard error of estimate of a power value $1 - \beta$ is less than $\sqrt{(0.001)(\hat{\beta})(1 - \hat{\beta})}$, or about 0.015 for individual values plotted in the mid-height range of Figure 6.2.

Thus the improvement in power of P_s over A^2 is significant as evident in Figure 6.2. Since P_s seems to provide protection in both directions of departure of σ from 1, it warrants one to investigate more fully its critical points. Due to the dependence of the p_i elements, finding an analytical expression for the distribution of these statistics seems untenable; however, using simulation, approximated critical points for the P_s are shown in Table 6.1. Note that the critical points of all statistics are those of the fully specified null distribution, case zero in D'Agostino's text. The simulation for this table relied on 10,000 iterations of each sample size, so only two digits are significant. Note that the distribution of P_s seems to have a heavy right tail, and as n increases, the tail becomes heavier. It is also interesting to note that for $n \leq 25$ the three critical values increase almost exactly linearly.

Table 6.1: Estimated critical values for P_s at various sample sizes and levels of significance.

n	0.90	0.95	0.99
2	4.9	6.1	8.9
3	7.6	9.1	13.4
4	10.1	12.1	17.0
5	12.6	15.3	21.5
6	15.1	18.1	24.4
7	17.7	21.1	28.2
8	20.4	23.9	32.0
9	22.7	26.8	36.5
10	24.9	29.4	39.5
11	27.9	32.2	43.7
12	30.0	35.2	48.0
15	37.5	44.0	59.6
20	50.7	58.7	81.1
25	63.2	76.2	116.5
30	80.0	107.1	218.4
40	445.0	576.5	776.8
50	1025.4	1108.8	1231.6

6.5 Conclusion

The purposes of this chapter are two-fold. First, the notion of using the information contained in the fit of order statistics is presented, as represented in the $\mathcal{P}$-vector, as a vehicle to define a new goodness-of-fit test statistic P_s. The performance of P_s with simulation experiments was examined. Figure 6.2 shows convincing evidence that P_s is a powerful goodness-of-fit test at least in some instances, so it is admissible relative to D_n and A^2. A secondary purpose is to show that this procedure is practical to carry out, now that computational languages such as Maple are becoming fully developed. Indeed, the code in Maple used to compute the critical values in Table 6.1 consists of about one page. Embedded in that code is the ability to find the n-fold vector of polynomials representing the $F_{Z(i)}$, each of degree up to n, that make up the beta distributed CDFs used to calculate the $\mathcal{P}$-vector elements. The leverage in the approach through the $Z_{(i)}$ is that Maple creates and stores the CDFs, not the programmer. The same simulation in FORTRAN or C would require the programmer to hard-code the n polynomial CDFs for each value of sample size in Table 6.1, and then call a separate subroutine to calculate the test statistics. This could be a significant programming task. The Maple version requires none of this, since the CDFs are in functional form and may be evaluated at specific points. Not only that, but for large n, in the case of $n = 50$ for example, the 50 CDFs of the order statistics may have extremely large integers for the coefficients of the terms, as well as for their exponents. Maple is able to conduct the mathematical operations without representing these constants as floating point approximations. Thus the precision exceeds an implementation in a high-level algorithmic language. It is apparent that the admissibility of P_s suggests attractive procedures for outlier detection and that inferences with censored samples may also be defined in terms of the $\mathcal{P}$-vector.

Chapter 7

Other Applications and Examples

7.1 Introduction

A key contribution of APPL is that it provides a new way of viewing existing problems and opens new avenues for their solution. This chapter provides examples where the problem-solving paradigm can shift due to the ability to find exact distributions of random variables. The six sections that follow highlight some diverse areas of application that can be addressed with APPL. The chapter starts with relatively simple examples involving the central limit theorem (CLT). In these examples one will be able to determine the error of the CLT approximations to distributions involving sums of independent random variables. The second section contains a demonstration of how to use APPL to generate mathematically intractable tables, graphs, and charts. effectively reducing the need for the volumes containing these entities. The third section contains a demonstration of how computation of exact distributions can enhance probabilistic model design using reliability block diagrams. The fourth section contains a demonstration of how the software allows for modeling systems using hazard functions. The fifth section provides a discussion on a simple methodology by which APPL furthers one's ability to identify outliers in samples. Finally, in the sixth section, a discussion of some explorations for creating estimation techniques using maximum likelihood with order statistic distributions is given. These six examples serve only to suggest the nature of applications of APPL; many more applications exist.

7.2 Exactness in lieu of CLT approximations

One use of the CLT with iid samples is to facilitate inference about the population mean. The CLT implies that $\bar{X} = \sum_{i=1}^{n} X_i$ is asymptotically normally distributed when the X_i's are iid with finite mean and variance. Application of the CLT becomes restricted, however, with small sample sizes, with skewed data, or in the presence of outliers (Moore and McCabe 1993, p. 510). In these cases, an alternative method is to determine the exact distribution of $\bar{X}$. Once the distribution $\bar{X}$ is found, one can use it to make statistical inferences about the population mean. A few examples follow.

Let $X \sim$ exponential(1). By the CLT, the approximate distribution of $Y = \bar{X}$ is approximately $\text{N}(1, \frac{1}{\sqrt{n}})$. For the case $n = 3$, one can compute the exact distribution of Y with the following commands:

```
> X := ExponentialRV(1);
> n := 3;
> Y := SumIID(X, n);
> Y := Transform(Y, [[x -> x / n], [0, infinity]]);
```

The resulting PDF is $f_Y(y) = \frac{27}{2}y^2e^{-3y}$ for $y > 0$ which is predictably non-normal in shape, as n is too low for the normality of the CLT to apply to such a skewed underlying distribution. One also notes that Y has the Erlang distribution. Now consider the case $n = 30$, often considered a sufficiently large sample size for an adequate CLT approximation. Here a computation of the error associated with the approximation is provided. To obtain the PDF of $\bar{X}$, one changes the second line of code to `n := 30;` and re-executes the Maple worksheet. The result is the following Erlang PDF:

$$f_Y(y) = \frac{246315313339233398437500000}{10577732774609}y^{29}e^{-30y} \qquad y > 0\ .$$

Listed below are selected fractiles for both the exact distribution of $\bar{X}$ and its CLT approximation, the N$(1, \frac{1}{\sqrt{30}})$ distribution. For the selected fractiles, the approxima-

Table 7.1: Fractiles for exact and approximated distributions.

Approach	Distribution	Fractile			
		0.9	0.95	0.975	0.99
Exact	$\bar{X}$	1.240	1.318	1.388	1.473
CLT	N$(1, \frac{1}{\sqrt{30}})$	1.234	1.300	1.358	1.425

tion is only good to about one digit beyond the decimal point. It is useful to note that the CLT approximation for critical values in confidence intervals for $E(X)$ may be in error; worse, they are not even on the conservative side. For example, a 97.5%

upper confidence bound using the CLT quantile 1.35784 actually provides only 96.6% confidence [note $P(\bar{X} < 1.35784) = 0.965975$ is found in the Maple worksheet with the command `CDF(Xbar, 1.357838829);`]. Note, the additional APPL commands to generate the values in Table 7.1 are the following:

```
> Z := NormalRV(1, 1 / sqrt(30));
> alpha := [0.9, 0.95, 0.975, 0.99];
> for i from 1 to 4 do
>   IDF(Y, alpha[i]);
>   IDF(Z, alpha[i]);
> od;
```

For a related application, consider an old, approximate method of generating random samples from the standard normal distribution. It is well documented (Park and Leemis, 1997, for example) that the following approach can be used to generate approximately standard normal samples, where the U_i's are iid $U(0, 1)$:

$$Z^* = U_1 + U_2 + \cdots + U_{12} - 6.$$

The following commands determine the exact distribution of this 12-fold convolution and the third and fourth moments of Z^* and Z:

```
> U := UniformRV(0, 1);
> U12 := SumIID(U, 12);
> Zstar := Transform(U12, [[x -> x - 6], [-infinity, infinity]]);
> Z := NormalRV(0, 1);
> skew := x -> x ^ 3;
> kurt := x -> x ^ 4;
> evalf(ExpectationRV(Zstar, skew));
> evalf(ExpectationRV(Z, skew));
> evalf(ExpectationRV(Zstar, kurt));
> evalf(ExpectationRV(Z, kurt));
```

The distribution of Z^* is useful because, like the standard normal distribution, its mean is zero, its variance is one, and its skewness (third moment, $E\left[\left(\frac{Z^*-E[Z^*]}{\sqrt{V[Z^*]}}\right)^3\right] = E\left[(Z^*)^3\right]$) is also zero. The first difference is in the fourth moment $E\left[(Z^*)^4\right]$ which equals 2.9 for Z^* and 3.0 for a standard normal random variable Z. Another more obvious difference between the distributions of Z and Z^* is that Z has support $(-\infty, \infty)$ while Z^* has support $(-6, 6)$. The distribution of Z^* is the following:

$$f_{Z^*}(x) = \begin{cases}
\frac{1}{39916800}(6+x)^{11} & -6 < x < -5 \\
-\frac{107}{20160}x^8 - \frac{18595037}{3326400} - \frac{517}{10080}x^7 - \frac{2477}{7200}x^6 - \frac{11737}{7200}x^5 - \frac{250657}{20160}x^3 - \frac{1113317}{60480}x^2 & \\
-\frac{1}{67200}x^{10} - \frac{1}{3628800}x^{11} - \frac{4726777}{302400}x - \frac{11}{30240}x^9 - \frac{54797}{10080}x^4, & -5 < x < -4 \\
\frac{894727}{665280} + \frac{7}{576}x^8 + \frac{99}{1120}x^7 + \frac{631}{1440}x^6 + \frac{1199}{800}x^5 + \frac{12199}{2240}x^3 & \\
+\frac{9385}{1728}x^2 + \frac{1}{17280}x^{10} + \frac{1}{725760}x^{11} + \frac{38533}{11200}x + \frac{11}{10080}x^9 + \frac{1009}{288}x^4, & -4 < x < -3 \\
\frac{61297}{166320} - \frac{25}{2016}x^8 - \frac{33}{560}x^7 - \frac{13}{72}x^6 - \frac{143}{400}x^5 - \frac{583}{1120}x^3 - \frac{1619}{3024}x^2 & \\
-\frac{1}{8064}x^{10} - \frac{1}{241920}x^{11} - \frac{781}{5600}x - \frac{11}{6720}x^9 - \frac{239}{504}x^4, & -3 < x < -2 \\
\frac{65521}{166320} + \frac{1}{252}x^8 + \frac{11}{1680}x^7 + \frac{1}{360}x^6 + \frac{11}{1200}x^5 + \frac{11}{3360}x^3 - \frac{563}{3024}x^2 + \frac{1}{6720}x^{10} & \\
+\frac{1}{120960}x^{11} + \frac{11}{50400}x + \frac{11}{10080}x^9 + \frac{25}{504}x^4, & -2 < x < -1 \\
\frac{655177}{1663200} + \frac{1}{1440}x^8 - \frac{23}{3600}x^6 - \frac{809}{4320}x^2 - \frac{1}{14400}x^{10} - \frac{1}{86400}x^{11} + \frac{31}{720}x^4 & -1 < x < 0 \\
\frac{655177}{1663200} + \frac{1}{1440}x^8 - \frac{23}{3600}x^6 - \frac{809}{4320}x^2 - \frac{1}{14400}x^{10} + \frac{1}{86400}x^{11} + \frac{31}{720}x^4 & 0 < x < 1 \\
\frac{65521}{166320} + \frac{1}{252}x^8 - \frac{11}{1680}x^7 + \frac{1}{360}x^6 - \frac{11}{1200}x^5 - \frac{11}{3360}x^3 - \frac{563}{3024}x^2 + \frac{1}{6720}x^{10} & \\
-\frac{1}{120960}x^{11} - \frac{11}{50400}x - \frac{11}{10080}x^9 + \frac{25}{504}x^4, & 1 < x < 2 \\
\frac{61297}{166320} - \frac{25}{2016}x^8 + \frac{33}{560}x^7 - \frac{13}{72}x^6 + \frac{143}{400}x^5 + \frac{583}{1120}x^3 - \frac{1619}{3024}x^2 & \\
-\frac{1}{8064}x^{10} + \frac{1}{241920}x^{11} + \frac{781}{5600}x + \frac{11}{6720}x^9 - \frac{239}{504}x^4, & 2 < x < 3 \\
\frac{894727}{665280} + \frac{7}{576}x^8 - \frac{99}{1120}x^7 + \frac{631}{1440}x^6 - \frac{1199}{800}x^5 - \frac{12199}{2240}x^3 + \frac{9385}{1728}x^2 + \frac{1}{17280}x^{10} & \\
-\frac{1}{725760}x^{11} - \frac{38533}{11200}x - \frac{11}{10080}x^9 + \frac{1009}{288}x^4, & 3 < x < 4 \\
-\frac{107}{20160}x^8 - \frac{18595037}{3326400} + \frac{517}{10080}x^7 - \frac{2477}{7200}x^6 + \frac{11737}{7200}x^5 + \frac{250657}{20160}x^3 - \frac{1113317}{60480}x^2 & \\
-\frac{1}{67200}x^{10} + \frac{1}{3628800}x^{11} + \frac{4726777}{302400}x + \frac{11}{30240}x^9 - \frac{54797}{10080}x^4, & 4 < x < 5 \\
\frac{1}{1120}x^8 - \frac{3}{280}x^7 + \frac{9}{100}x^6 - \frac{27}{50}x^5 - \frac{243}{35}x^3 + \frac{486}{35}x^2 + \frac{1}{604800}x^{10} & \\
-\frac{1}{39916800}x^{11} - \frac{2916}{175}x - \frac{1}{20160}x^9 + \frac{17496}{1925} + \frac{81}{35}x^4 & 5 < x < 6
\end{cases}$$

In the overlaid Maple plots of the standard normal PDF and $f_{Z^*}(x)$ shown in Figure 7.1, it is apparent that this approximate distribution fits the standard normal distribution in the PDF very nicely except around $x = 0$ [where $f_Z(0) = \frac{1}{\sqrt{2\pi}} \cong 0.399$

and $f_{Z^*}(0) = \frac{655177}{1663200} \cong 0.394$] and, of course, in the tails, since the approximate distribution only has support on $(-6, 6)$. To obtain overlaid plots of these two distributions, one invokes the PlotDist command. First one creates a plot of Z^* and stores it as Plot1. Next, one creates a plot of the standard normal distribution and stores it as Plot2. Finally, used the plots[display] command to combine the two plots. This is summarized in the commands:

```
> Plot1 := PlotDist(Zstar, -6, 6);
> Plot2 := PlotDist(NormalRV(0, 1), -6, 6);
> plots[display](Plot1, Plot2, scaling = unconstrained);
```

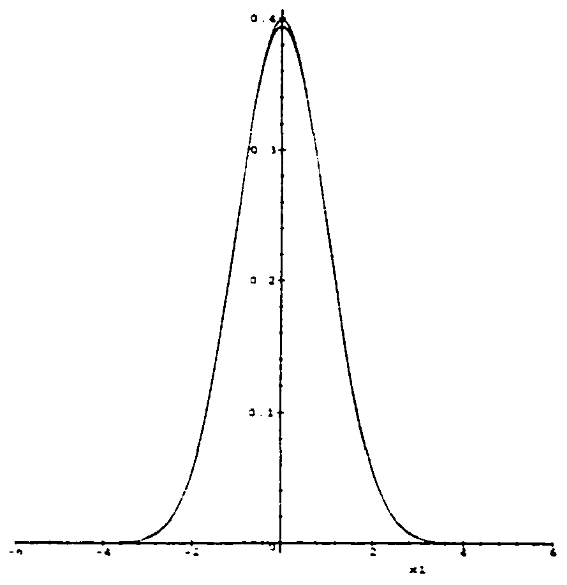


Figure 7.1: Overlaid plots of $f_{Z^*}(x)$ and the standard normal PDF.

7.3 A mathematical resource generator

APPL is also able to generate tables, graphs, and charts. The term *resource* refers to tables, graphs, and charts, to include such items as means, critical points, variances, and so on. APPL can be invoked to reduce the need for such tables. Additionally, since these computations can be produced automatically, programs can be designed to produce needed values in real time, as opposed to being restricted to knowing in advance which element of which table will be needed for program execution. There are many references for critical tables for common distributions that software such as this (and others) effectively replace. Highlighted, however, will be two recent books, *CRC Handbook of Tables for Order Statistics from Inverse Gaussian Distributions with Applications* (Balakrishnan and Chen, 1997) and *CRC Handbook of Tables for the Use of Order Statistics in Estimation* (Harter and Balakrishnan, 1996). These books were selected because of the relatively complex nature of the information they present. Also, they were chosen to highlight that even as late as 1997, the scientific community had to rely on tables and charts for a significant amount of probabilistic information.

First is a comparison of Balakrishnan and Chen's *CRC Handbook of Tables for Order Statistics from Inverse Gaussian Distributions with Applications* with APPL's capabilities. This book begins with 39 pages of theory and references to the inverse Gaussian (IG) distribution and its order statistics. Then the book has 645 pages of PDF plots, expected value tables, covariance tables, estimation tables, and so forth.

APPL is able to create a number of these figures and tables, and in some cases goes beyond those presented in the text. Page 50, for example, shows the N(0, 1) and the standardized IG(0.8) PDFs overlaid in a single plot. The standardized IG distribution is defined as follows. Let $Y \sim$ IG(λ, δ). Let $\sigma = \sqrt{\delta^3/\lambda}$ (the standard deviation of Y). Define $Z = (Y - \delta)/\sigma$ as the standard IG distribution, and let $k = 3\sqrt{\delta/\lambda}$. Then the PDF of Z (which is listed as equation (2.14) on page 7 of the text) is as follows:

$$f_Z(z) = \frac{1}{\sqrt{2\pi}}\left(\frac{3}{3+kz}\right)^{3/2} e^{-3z^2/(6+2kz)} \qquad -\frac{3}{k} < z < \infty .$$

Then the two plots may be overlaid as follows:

```
> X := NormalRV(0, 1);
> k := 0.8;
> Z := [[z -> 1 / sqrt(2 * Pi) * (3 / (3 + k * z)) ^ (3 / 2) *
  exp(-3 * z ^ 2 / (2 * (3 + k * z)))], [-3 / k, infinity],
  ['Continuous', 'PDF']];
> Plot1 := PlotDist(X, -4, 4);
> Plot2 := PlotDist(Z, -4, 4);
> plots[display]({Plot1, Plot2}, scaling = unconstrained);
```

The Maple-produced plot is found in Figure 7.2. Balakrishnan and Chen use such plots to show how the two distribution separate as k increases from zero. For plotting the IG PDF, one is not limited to the standardized version of the distribution. One could plot any IG PDF, CDF, HF, and so on. For example, the command

```
> PlotDist(HF(InverseGaussianRV(1, 0.5)), 0, 3);
```

will generate a plot of the IG(1, 0.5) hazard function. Clearly, one is not limited to just PDF plots in APPL, nor is one limited to plots of only standardized IG distributions.

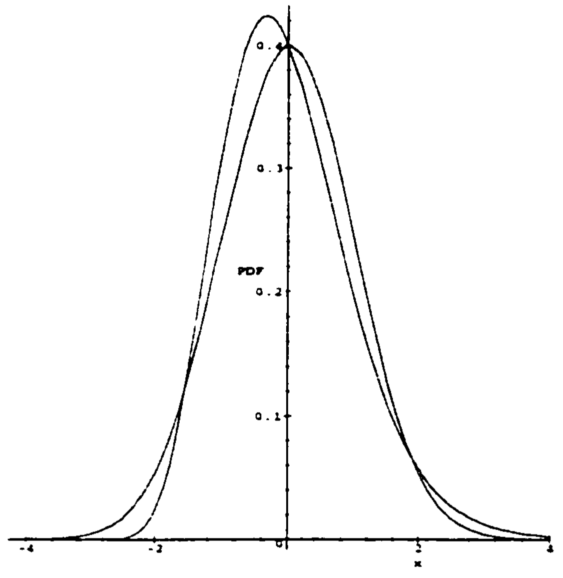


Figure 7.2: Overlaid plots of the standard normal and standardized IG(0.8) distributions.

Another recent resource, *CRC Handbook of Tables for the Use of Order Statistics in Estimation* (Harter and Balakrishnan, 1996), also gives extensive charts for various applications of order statistics. Appendix C of this book, covering pages 326–530, lists tables of various means, variances, and covariances of order statistics from many common distributions. For example, Table C1.1 gives means of order statistics for the N(0, 1) distribution, Table C2.1 for the exponential(1) order statistics, and so on. APPL can replicate such tables. For instance, to produce the mean of the $n = 8, r = 6$ order statistic from the Weibull(1, 0.5) distribution, use the commands:

```
> X := WeibullRV(1, 0.5);
> X86 := OrderStat(X, 8, 6);
> MeanRV(X86);
```

which returned the value 1.760598073, compared to 1.76060 on page 392 of Harter and Balakrishnan (1996).

Unlike Harter's text, the software can go beyond computing means of order statistics. In the following example (continuing the code from the three lines of code above), one can, for example, compute the 98th percentile of the same Weibull distribution and give the probability of exceeding 1.92:

```
> IDF(X86, 0.98);
> SF(X86, 1.92);
```

The values returned are 6.480288415 and 0.0473928543, respectively. Furthermore, we are not limited to the relatively few base distributions that Harter and Balakrishnan present in Appendix C, but can enter any Weibull parameter values.

There is no implication that these, or any of the vast resource collections, are archaic. Actually, what is envisioned is the need of this software to augment the information presented in texts such as these. Foreseen is a shifting away from looking up values in table; instead, one will encode the creation of necessary values directly into programs. The first 208 pages of Harter's text, for example, still review theory behind the creation and use of order statistics. APPL adds to the resource base that books such as these provide.

7.4 Probabilistic model design: reliability block diagrams

The analysis of reliability block diagrams (RBD) in reliability engineering can be facilitated by use of APPL. A program called SHARPE (Symbolic Hierarchical Automated Reliability/Performance Evaluator, presented in Sahner and Trivedi, 1993) uses Mathematica to model reliability block diagrams and other types of systems. In their article, Sahner and Trivedi show how SHARPE is capable of providing system reliabilities. However, according to Trivedi (1997), input distributions are limited to exponential distributions and "exponential polynomials which are equivalent to Cox distributions." APPL can conduct similar computations and is not limited to such distributions. A demonstration is provided by duplicating Trivedi's example, and then showing possible extensions. This is not meant to be a direct comparison of SHARPE and APPL, as the two software packages have completely different goals. The two softwares merely intersect to some degree, as this RBD example shows.

In Sahner and Trivedi (1993), a computer system consists of two processors and three memory units. The computer is designed to operate with as few as one processor and one memory unit. Thus a RBD to represent the system would be as shown in Figure 7.3, in which two parallel processors are in series with three parallel memory units. All components are independent and have exponentially distributed times to failure.

The CDF of of `Xsys`, the time to failure of the system, can be determined with

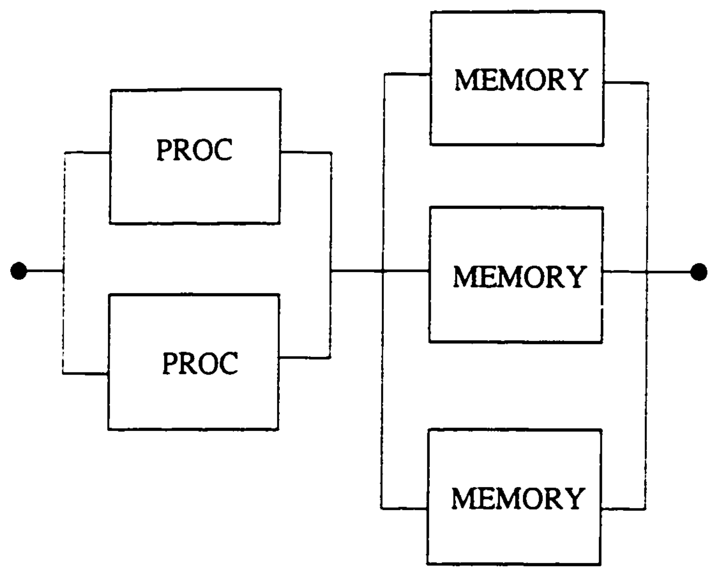


Figure 7.3: RBD of a computer system with two processors and three memory units.

the following commands in APPL:

```
> Xp := ExponentialRV(0.00139);
> Xm := ExponentialRV(0.00764);
> Xsys := MinimumRV(MaximumRV(Xp, Xp), MaximumRV(MaximumRV(Xm, Xm), Xm));
> Xsys := CDF(Xsys);
```

The result is the following CDF

$$F(x) = 1 - 6e^{-0.00903x} + 6e^{-0.01667x} - 2e^{-0.02431x} + 3e^{-0.01042x} - 3e^{-0.01806x} + e^{-0.0257x}$$

where $x > 0$. While this CDF is different from that in Trivedi's article, in a subsequent personal communication (Trivedi 1997), Trivedi agrees this expression is the correct CDF; the one in the article being in error.

The areas for improvement of problem solving by APPL are many. First, with APPL one can model the time to failure of the individual components with distributions other than the exponential. Second, one can analyze the system by looking at the functional form of the CDF (as in SHARPE), but also graphically, by plotting the distribution. Trivedi mentions that a plotting function is currently in beta testing for SHARPE (Trivedi, 1997). One can also determine and plot the SF, HF, CHF, and PDF of the system. One can sum Xsys and Xsys to find the distribution of the lifetime of two such systems in a passive standby configuration using SumRV. Fractiles of the distributions can be produced. One can conduct "what-if" analysis by varying the number of processors and memory units in the system to see the overall effect on the mean system lifetime. One can run a reliability life test to see if observed failure times are adequately fit by this model, using the goodness-of-fit techniques outlined in Chapter 6.

7.5 Modeling with hazard functions

The capabilities of APPL allow a shift in the paradigm of parameterized model design in new ways. In some situations, it may be easy to imagine the reliability of a system in terms of instantaneous rate of failure, i.e., the hazard function. Reliability texts often classify lifetime distributions into *distribution classes* that are named after the shape of the hazard function. Leemis (1995, Chapter 3), among others, defines many of these classes, to include the increasing failure rate (IFR), the decreasing failure rate (DFR), and the bathtub (BT) shaped hazard function.

One possible use for APPL is to model systems using a hypothesized shape of a hazard function. For example, should a system be hypothesized to have a bathtub-shaped HF, there are only a few common distributions with HFs with such a shape. Leemis (1995, p. 100) lists only two of 16 common reliability distributions as having a BT-shaped HF. Instead of being limited to these distributions, one may hypothesize a family of distributions with BT-shaped HFs. For example, a second-order polynomial HF of the form $a(x-b)^2$ will have the BT shape as long as $a > 0$ and $b > 0$. It is easy to verify these polynomials satisfy the requirements for a HF, as listed in Leemis (1995, p. 50):

$$\int_0^\infty h(t)dt = \infty \quad \text{and} \quad h(t) \geq 0 \quad \text{for all} \quad t \geq 0\,.$$

Here is an example of using APPL to model with hypothesized HFs. Let a sample of failure times be [1, 11, 14, 16, 17]. Assuming it is hypothesized that the system should be fit by the quadratic BT-shaped HF distribution, one should fit these data to the unspecified distribution as follows.

```
> assume(a > 0);
> assume(b > 0);
> T := [[t -> a * (t - b) ^ 2], [0, infinity], ['Continuous', 'HF']];
> PDF(T);
```

The last command returns the general form of the PDF having the assumed HF:

$$f_T(t) = a(t^2 - 2tb + b^2)\, e^{-ta\,(t^2-3tb+3b^2)/3} \qquad t > 0.$$

One can find the values of $\hat{a}$ and $\hat{b}$ that maximize likelihood as follows:

```
> sample := [1, 11, 14, 16, 17];
> hat := MLE(T, sample, [a, b]);
> evalf(hat);
```

The results are: $\hat{a} = 0.0037937$, $\hat{b} = 5.93512$. The plot of the SF for T overlaid on the empirical SF is shown in Figure 7.4.

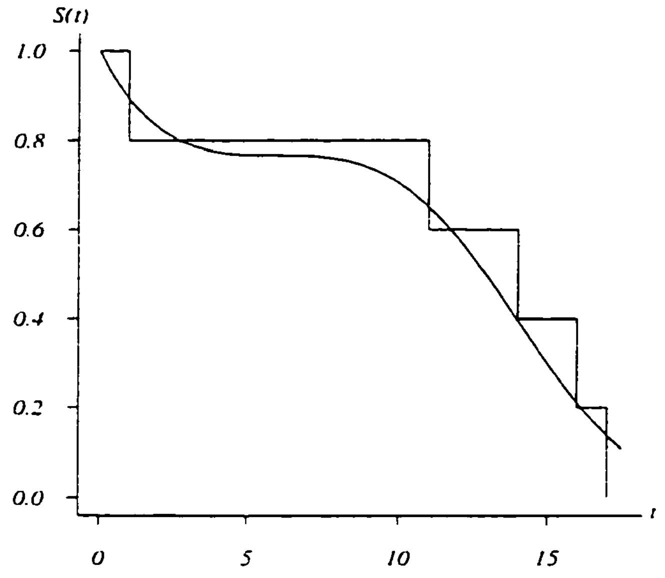


Figure 7.4: The SF of the hypothesized BT-shaped hazard function fit to the sample [1, 11, 14, 16, 17] overlaid on the empirical SF.

For another example of modeling in terms of hazard functions, let us hypothesize that risk to a system is seasonal. Such is the case in many applications such as structures at risk to occurrences of category five hurricanes. A periodic hazard function might be modeled in such a case. Suppose one hypothesizes a family of periodic HF distributions having HFs of the form

$$h_T(t) = a + b\sin(ct). \qquad t > 0;\ a > |b|, \text{ and } a, b, c \in \Re.$$

Once again, these functions satisfy the requirements of HFs. The parameter a represents a measure of the long-term constant risk associated with T. In other words, increased values of a correnspond to a higher likelihood of chance failures. The parameters b and c control the amplitude and period of the HF, modeling the severity and length of the cyclic stresses.

In their article, Lee, Wilson, and Crawford (1991) modeled seasonal ocean storm data in terms of a nonhomogeneous Poisson process with intensity function of the form

$$\lambda(t) = e^{\alpha+\gamma\sin(\omega t+\phi)} \qquad t > 0; \alpha, \gamma, \omega, \phi \in \Re$$

where the exponentiation is used to assure that $\lambda(t) > 0$ for all t. There are two differences between the NHPP model of Lee, Wilson, and Crawford (1991) and modeling with a seasonal hazard function. First, the intensity function method models multiple occurrences of seasonal events while the hazard function method models a single occurrence. Second, the intensity function method assumes that the process has independent increments, among other assumptions (see Ross, 1993, p. 236, for all assumptions). Such assumptions are not necessary with hazard function models. One can instantiate the unspecified distribution with the following command

```
> T := [[t -> a + b * sin(c * t)], [0, infinity], ['Continuous', 'HF']];
```

and the PDF may be found with the command `PDF(X);`. The unspecified PDF of this family of distributions with a periodic hazard function is

$$f_T(t) = (a + b\sin(ct))\,e^{-(atc-\cos(ct)b+b)/c} \qquad t > 0.$$

Letting the parameters take on values of $a = 1$, $b = 0.5$ and $c = 10$. One gets the PDF (using the `PlotDist` command) plotted in Figure 7.5, a peculiarly multi-modal distribution that decreases exponentially over time.

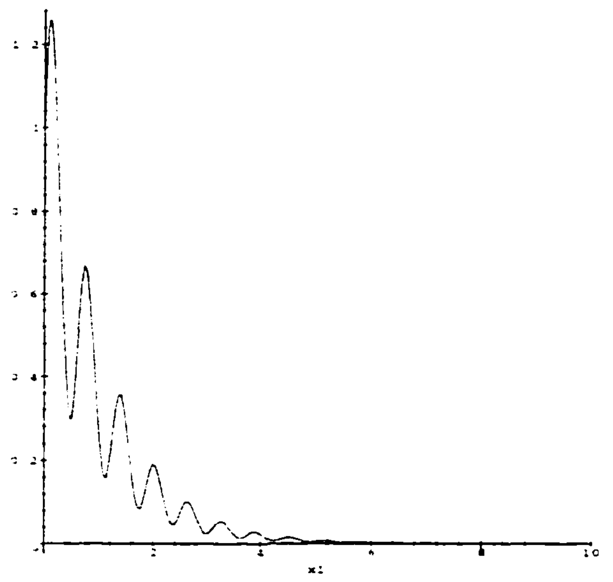


Figure 7.5: The PDF of the distribution having periodic hazard function h_X with parameters $a = 1$, $b = 0.5$ and $c = 10$.

7.6 Outlier detection

The theory and practice of identifying outliers in a data set is another contribution provided by APPL. The literature contains ample methods for identifying outliers in samples related to a normal distribution. Regarding detection of outliers, D'Agostino (1986, p. 497) writes "We shall discuss here only the underlying assumption of normality since there is very little theory for any other case." Sarhan and Greenberg (1962, p. 302) and David (1981) propose a number of test statistics based on standardized order statistics of normally distributed data. They provide charts of acceptance and rejection regions in lieu of p-values. For instance, in the fully specified case where μ and σ are known, Sarhan and Greenberg propose the test statistic $(X_{(n)} - \mu)/\sigma$ for testing for extreme observations in normal samples. As has been discussed in Chapter 6, the distributions of order statistics can be used to identify outliers in data sets.

APPL may contribute to the outlier detection problem in at least three ways:

1. There is no need to standardize the test statistic, since the distribution of the r^{th} order statistic may be found with the procedure `OrderStat`.

2. One need no longer rely on charts of rejection regions, since once the distribution of the r^{th} order statistic is known, one can calculate p-values.

3. Most importantly, one is not limited to the assumption of normality. APPL facilitates finding the distribution of the r^{th} order statistic of a wide range of distributions.

Here are a few examples. In the sample 1, 2, 3, 4, 5, 10, it is possible that the last element in the sample is an outlier. Assuming that the population has a mean of 3, one could find the order statistic probability $P(X_{(6)} \geq 10)$ for samples of size 6 from various underlying distributions. Let us consider the following possible population distributions, each with mean of 3 (approximately 3 for the Weibull case): Weibull(0.295, 2), exponential(1/3), N(3, 2), and N(3, 4). The upper tail probability $P(X_{(6)} \geq 10)$ gives us the significance level against the null hypothesis that $X_{(6)}$ came from this underlying distribution. The four probabilities can be found with the following commands:

```
> X1 := WeibullRV(0.295, 2.0);
> X2 := ExponentialRV(1 / 3);
> X3 := NormalRV(3, 2);
> X4 := NormalRV(3, 4);
> OrderStat(X1, 6, 6);
> SF(", 10);
> evalf(SF(OrderStat(X2, 6, 6), 10));
> evalf(SF(OrderStat(X3, 6, 6), 10));
> evalf(SF(OrderStat(X4, 6, 6), 10));
```

The results are shown in Table 7.2. An interpretation is that for $n = 6$ the value 10

Table 7.2: $P(X_{(6)} \geq 10)$ for $n = 6$ for several population distributions.

Distribution	$P(X_{(n)} \geq 10)$
Weibull(0.295, 2)	0.0009966
exponential(1/3)	0.1958385
N(3, 2)	0.0013950
N(3, 4)	0.2175316

is an outlier with respect to the N(3, 2) distribution at the 0.0014 level of significance and to the Weibull(0.295, 2) population distribution at the 0.001 level of significance.

It is not a significant outlier to the other two population distributions. Clearly one is not limited to maximum values of a sample, one could also consider any of the order statistics (as outlined in the Chapter 6, using the $\mathcal{P}$-vector). Note, these four upper tail probabilities are the 6th element of the four $\mathcal{P}$-vectors. Also note the lack of parallelism in the commands. One would expect the command `evalf(SF(OrderStat(X1, 6, 6), 10))` would have produced the appropriate probability for the Weibull distribution. However, Maple locked up with this embedding. By separating the command into two parts, Maple computed the probability correctly and quickly.

An extension to this detection example is found in the last two population distributions, the normal distributions with σ equal to 2 and 4. One sets a specified level of significance, say $\alpha = 0.05$, and solves for the value of σ that will show a maximum value of 10 to be an outlier for the normal family of distributions. There are two directions to this solution. First one might be tempted to conduct a binary search on the interval $2 < \sigma < 4$ to approximate s to a certain degree of accuracy. The following code will accomplish that approximation and will derive $\hat{\sigma} = \frac{375}{128} \cong 2.9297$.

```
alpha := 0.05; sigold := 4; signew := 2; found := false;
while(found = false) do
  X := NormalRV(3, signew);
  significance := evalf(SF(OrderStat(X, 6, 6), 10));
  delta := abs(significance - alpha);
  if(delta < 0.00001) then
    found = true;
    print(sigmaisnow, sigold); break;
  else
    if(significance < alpha) then
      temp := signew;
      signew := signew + abs(signew - sigold) / 2;
      sigold := temp;
```

```
        print(newsigma, signew);
      else
        temp := signew;
        signew := signew - abs(signew - sigold) / 2;
        sigold := temp;
        print(newsigma, signew);
      fi;
    fi;
  od;
```

A second approach is to rely on the exactness of distributions that APPL provides. In effect what is desired is a solution to the following. Letting $X \sim$ N(3, σ), find σ such that

$$P(X_{(6)} < 10) = 0.95.$$

One can find that exactly by manipulating the CDF list-of-lists representation of $X_{(6)}$ in the following way. The first element of the first list will contain the CDF, so it is set equal to 0.95 and solved for σ. The following code produces the solution $\sigma \cong 2.933571640$.

```
> X := NormalRV(3, s);
> X6 := OrderStat(X, 6, 6);
> X6 := CDF(X6);
> fsolve(X6[1][1](10) = 0.95, s = 2 ..  4);
```

The first line of code sets the semi-specified normal distribution as the population distribution for X. The second line determines the PDF of $X_{(6)}$. The third line converts the PDF of $X_{(6)}$ to a CDF. The last line isolates the first element of the first list of `X6` which is the CDF function of the independent variable x and the unknown parameter σ. The entry `(10)` provides the independent variable a value of 10. Then the `fsolve` command solves for the unknown parameter `s` which represents σ.

The same type of determination could be made for any of the distribution families. One can likewise, for example, determine the value of μ that rejects 10 as an outlier. In effect, one is able to establish a distribution's tolerance interval for a parameter, given a specific minimum or maximum data value. Thus the detection of outliers is relative to an assumed underlying distribution, and one can partition the parameter space of underlying distributions in terms of outlier consistency.

7.7 Maximum likelihood of order statistics distributions

An idea is presented in this section that has some intuitive appeal towards improving estimation of parameters. Borrowing from the idea of estimating parameters via maximum likelihood, and also suggesting that order statistics might be helpful in estimating parameters, the usual form of the likelihood function,

$$L_1(\mathbf{x}, \theta) = \prod_{i=1}^{n} f_X(x_i, \theta)$$

(where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$), is replaced with a form involving the PDFs of the order statistics:

$$L_2(\mathbf{x}, \theta) = \prod_{i=1}^{n} f_{X_{(i)}}(x_{(i)}, \theta) .$$

The second method is referred to as maximum likelihood estimation with order statistics (MLEOS). Note that L_1 is the joint distribution of iid random variables. The function L_2 does not lend itself to such a direct representation, as order statistics are not independent.

This approach to estimation borrows from the intuition associated with traditional maximum likelihood estimation. In his Figure 7.7, Leemis (1995, p. 172–173) gives a graphical interpretation of the MLE method. Leemis suggests the goal of MLE is to find the parameter θ that "maximizes the product of the density values at the data points." Since in MLE one seeks a value of θ that maximizes the product of the density values from the PDF, it was wondered what would happen if the product of the density values of the n order statistics was maximized. Maximizing the product of the "hump-shaped" order statistic PDFs evaluated at their respective ordered data values is graphically depicted as finding a θ value that maximizes the product of the lengths of the vertical lines in Figure 7.6 (the lines from the ordered data points to their ordered density values). A good fit places each of the data values near the center of the n humps of the n PDFs. MLEOS will find a λ value that maximizes the products of the respective density values. Note the x marks on the horizontal axis correspond to the sample [0.2, 0.4, 0.9, 1.5].

Parameter estimates were computed from various population distributions from simulation using both MLE and MLEOS. The empirical mean square errors (MSE) of the two estimation techniques were compared. Also compared were estimates for λ from the exponential distribution, μ and σ from the normal distribution, and θ from the $U(0, \theta)$ distribution. Of all these population distributions, only the last case, estimating θ from the uniform distribution, resulted in a decrease in MSE of the

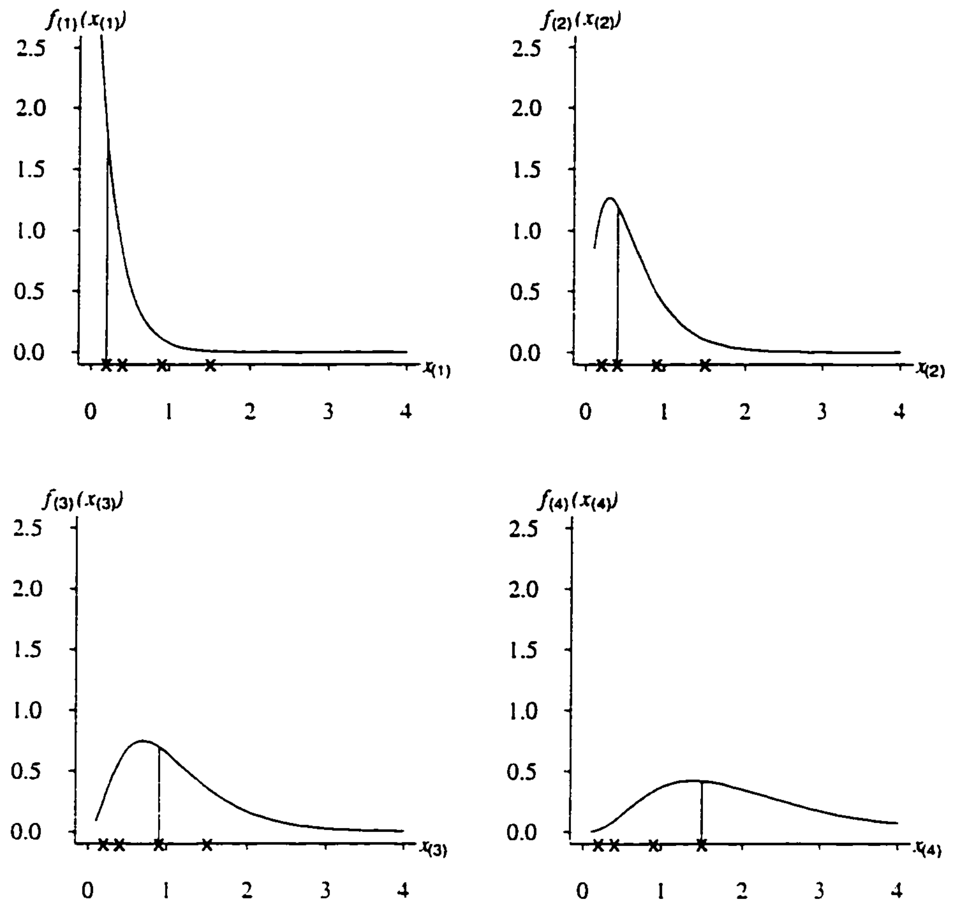


Figure 7.6: The PDFs of the four order statistics from an exponential distribution.

estimator. Since some ML estimators are also minimum variance unbiased estimators (MVUE), it not feasible for MLEOS to dominate MLE with respect to MSE. This is the case for estimating λ for the exponential distribution and μ for the normal distribution (Kendall and Stuart, 1963). However, the parameter σ from the normal distribution is not a MVUE (Kendall and Stuart, 1963, p. 10) and the MLE for θ in the uniform distribution is biased (Larsen and Marx, 1986, p. 244). For the uniform distribution, the reason that the MLEOS estimator dominated the MLE estimator is easy to see. Each MLE estimate of θ is the maximum of the sample. The MLEOS estimator for θ is either the maximum data value, or the solution to $\frac{\partial L_2(\mathbf{x},\theta)}{\partial\theta} = 0$. In about 32% of the samples, the solution to this equation was greater than the maximum of the sample. The simulation comparisons of the MLE and MLEOS estimators were designed as follows: set the sample size to 25, and the number of replications to 1000. For the simulation it was assumed that $\theta = 1$, and population variates from the U(0, 1) distribution were generated. The Maple code for this simulation is included in Appendix H. Also included is the performance of the unbiased estimator for θ, which is $\frac{n+1}{n}\max\{X_1, X_2, \ldots, X_n\}$. The results are given in Table 7.3. The MLEOS estimator had less bias and a lower MSE than the

Table 7.3: The MSEs of the MLE and MLEOS and adjusted-for-bias MLE techniques of parameter estimation.

Technique	Average Estimate	MSE of Estimates
MLE	0.9615	0.002768
MLEOS	0.9720	0.002020
MLE with unbiased factor	0.9999	0.001388

MLE estimator. As could be expected, however, the ML estimator with the unbiasing factor, $\frac{n+1}{n}\max\{X_1, X_2, \ldots, X_n\}$, dominates on both measures.

Chapter 8

Conclusion and Further Work

This dissertation presents work on a proposed probability software language. We have demonstrated some areas of applicability and contributions to the applied community provided by this software. As it exists, APPL is a viable language that demonstrably contributes to the applied community. At its core, it is a tool to produce the distribution functions of random variables. Once these distribution functions are known, then all the typical probability work that is done with distributions is possible, i.e., significance levels, critical points, expectations, plots, and so on. There is a veritable lifetime's amount of research that remains with APPL. The areas for further work fall into two categories, improvements to APPL itself, and further areas of application.

The language itself has potential for further improvement. So far it is restricted to univariate, continuous distributions. Multivariate considerations could follow. Also, more procedures could be devised to include areas of probability such as conditioning, Bayesian analysis, time series analysis, point processes, and others. Additionally, the software could encompass discrete distributions, as well as mixed discrete and continuous distributions.

Application areas are endless for APPL. A number were presented here, but there are many more areas that could benefit from such software. These areas include parameter estimation and goodness-of-fit testing in censored samples, finding the distribution of the total-time-on-test in a life-testing environment (surprising similarities exist here with the difficult integration of the K–S CDF), and renewal theory, to name a few. As APPL becomes more exposed, countless other areas will emerge.

Finally, there is another inherent contribution in that the software has been kept extremely general on purpose. Consider the creators of the spreadsheet. They had the insight to make the application general enough that it solved their current problem yet could be applied to many larger problems. Decades later no researcher, business office, or student's room is complete without a state-of-the-art spreadsheet at their disposal. This software research has maintained this same spirit of generality for similar reasons. For example, by allowing for parameterized, piecewise distribution in the "list-of-lists" format, nearly every continuous, univariate distribution can be represented by APPL. Thus the software can be of use to the reliability engineer concerned with a random variable having support $(0, \infty)$, as well as the theorist who is interested in the order statistics from populations having segmented distributions with support in the negative range. While probably not as common or widely needed as the spreadsheet, the author foresees the possibility that eventually no probabilist's resources will be complete without a symbolic probability package with procedures such as these. Thus, the generality of APPL is yet another contribution because it allows for yet unseen contributions by those interested in expanding the probabilistic world via an automated, programming environment.

Appendix A

The Arctangent Survival Distribution

A.1 Introduction

This appendix contains some early work on creating new distributions of random variables based on the SF-like qualities of the arctangent function. The work introduced the author to the need for software that could assist in probability calculations, plots, and other functions. Presented here is the development of a two-parameter survival distribution that has an upside-down bathtub (UBT, or humped-shaped) hazard function. This distribution provides biostatisticians, reliability engineers, and other statisticians with a second two-parameter UBT model whose closed-form survivor function simplifies the analysis of right-censored data sets. Maximum likelihood estimators of the parameters are found using numerical methods. Approximate confidence intervals can be determined by using the observed information matrix or the likelihood ratio statistic. Examples are given in which the arctangent distribution is a reasonable alternative to other common lifetime distributions.

Parametric lifetime distributions have been used in reliability for modeling the time to failure of components and systems. Although emphasis has traditionally been placed on models where the hazard function has a bathtub shape, applications have been found where an upside-down bathtub (UBT), or hump-shaped hazard function is the appropriate model. Kececioglu (1991, p. 425) lists transistors, metals subjected to alternating stress levels, insulation degradation, mechanical devices subjected to wear, and bearings as potential UBT applications. Chhikara and Folks (1989, p. 156) state that "When early occurrences such as product failures or repairs are dominant in a lifetime distribution, its failure rate is expected to be non-monotonic, first increasing and later decreasing" and cite airborne communication transceivers (p. 5, 139–140) as an application. Lee (1992, p. 12) further supports the validity of a UBT risk model in describing patients with tuberculosis who have risks that "increase initially, then decrease after treatment." To further substantiate the usefulness of the UBT model, Barr (1994) opines that the UBT risk function would apply in modeling the probability of a soldier becoming a casualty as a result of artillery fire. In this example, casualty risk starts out low as the fire is initially inaccurate, increases as the shooter hones in on the target, and then decreases as the remaining soldiers are able to "dig in" for protection. Although reliability engineers generally have an abundance of two-parameter survival distributions to choose from, relatively few have a UBT hazard function. The commonly-used UBT distributions are the inverse Gaussian, log normal, and log logistic distributions. Of these, only the log logistic distribution has a closed-form survival function. This distribution is most often used in biostatistical applications, whereas the inverse Gaussian and log normal are typically used in reliability.

The arctangent distribution developed here gives a survival distribution with a UBT hazard function and closed-form survivor function, a useful feature in the analysis of a right-censored data set. Additionally, the survivor function can be inverted in closed-form, which enables synchronization and monotonicity in variate generation. Unlike most survival distributions, the arctangent distribution's development uses trigonometric functions. We present the arctangent distribution's development, probabilistic properties, and statistical inference. Parameter estimation for complete and right-censored data sets is found by maximum likelihood. Finally, three examples illustrate situations when the distribution is a demonstrably reasonable alternative to other survival distributions.

A.2 Development

The arctangent function, when negated and shifted vertically, resembles a survivor function. Further, by shifting the function so that it crosses the vertical axis at 1 and then asymptotically decreases to 0, one obtains a flexible survivor function. Adding a phase shift parameter ϕ and a positive scaling parameter α yields the function

$$g(t) = -\arctan[\alpha(t-\phi)] + \frac{\pi}{2} \qquad -\infty < t < \infty,$$

which is a decreasing function with a range of $(0, \pi)$. Finally, since $g(0) = \arctan(\alpha\phi) + \frac{\pi}{2}$, the appropriate way to adjust this function so that it assumes the value 1 when $t = 0$ is to divide $g(t)$ by $g(0)$, yielding the survivor function for the random lifetime T

$$S_T(t) = \frac{-\arctan[\alpha(t-\phi)] + \frac{\pi}{2}}{\arctan(\alpha\phi) + \frac{\pi}{2}} \qquad t \geq 0.$$

Since the arctangent is an odd function, the form of the survivor function that will be used here is

$$S_T(t) = \frac{\arctan[\alpha(\phi - t)] + \frac{\pi}{2}}{\arctan(\alpha\phi) + \frac{\pi}{2}} \qquad t \geq 0, \qquad (1)$$

where $\alpha > 0$ and $-\infty < \phi < \infty$. This survivor function satisfies the three existence conditions: $S(0) = 1$, $\lim_{t\to\infty} S(t) = 0$, and $S(t)$ is nonincreasing. Furthermore, the distribution's probability density function and hazard function are

$$f_T(t) = -S_T'(t) = \frac{\alpha}{\left[\arctan(\alpha\phi) + \frac{\pi}{2}\right]\left[1 + \alpha^2(t-\phi)^2\right]} \qquad t \geq 0, \qquad (2)$$

$$h_T(t) = \frac{f_T(t)}{S_T(t)} = \frac{\alpha}{\left[\arctan[\alpha(\phi - t)] + \frac{\pi}{2}\right]\left[1 + \alpha^2(t-\phi)^2\right]} \qquad t \geq 0. \qquad (3)$$

This arctangent distribution is equivalent to a Cauchy distribution truncated on the left at $t = 0$. Thus the parameter α is similar to a scale parameter and the parameter ϕ is similar to a phase shift or location parameter. Figure A.1 shows three different PDFs of the arctangent distribution. Notice that as ϕ becomes negative, the

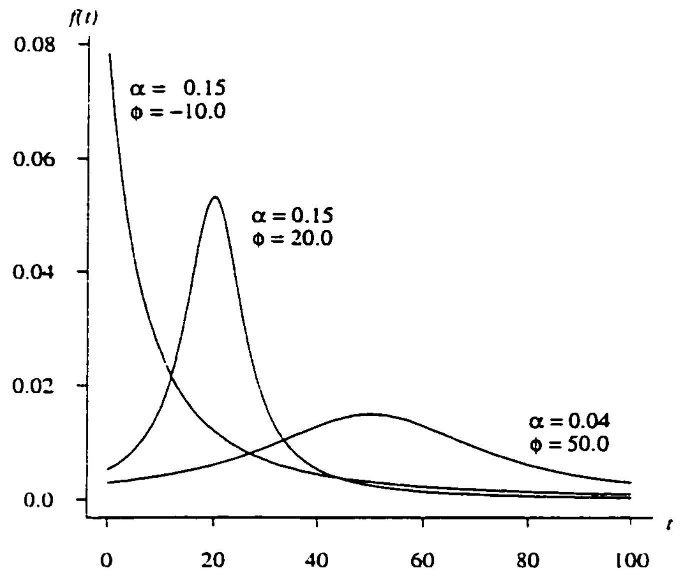


Figure A.1: Examples of the arctangent probability density function.

distribution changes from a bell-shaped distribution to a distribution with a mode of 0. Also notice that the parameter α controls the "peakedness" of the distribution: the dispersion of the distribution is a decreasing function of α.

Initially the name "Arctangent Distribution," was chosen because of the similarities that the arctangent function has with a generic survivor function. The relationship to the Cauchy distribution was noticed later. The CDF of the Cauchy distribution is also a shifted and scaled arctangent function.

A.3 Probabilistic properties

The arctangent distribution has several useful probabilistic properties that make it a viable distribution for lifetime data analysis. Specifically, it enjoys closed-form survivor and hazard functions, unlike most distributions in the UBT class. The closed-form survivor function simplifies parameter estimation for censored data sets and allows for variate generation via inversion. The probabilistic properties include:

- The distribution is in the UBT class when $\alpha\phi > c$ and is in the decreasing failure rate (DFR) class when $\alpha\phi \le c$ where

$$1 + 2c\arctan(c) + c\pi = 0,$$

  found by simplifying $h'_T(0) = 0$. Using numerical methods, $c \approx -0.42898$.

- The mode of the distribution is $t_{\text{mode}} = \phi$, $\forall\ \phi > 0$. For the case where the probability density function is monotonically decreasing, $t_{\text{mode}} = 0$, $\forall\ \phi \le 0$.

- The p^{th} fractile of the distribution is

$$t_p = \phi + \frac{1}{\alpha}\tan\left[\frac{\pi}{2} - (1-p)\left(\arctan(\alpha\phi) + \frac{\pi}{2}\right)\right]. \qquad (4)$$

  which yields a median of

$$t_{0.5} = \phi + \frac{1}{\alpha}\tan\left[\frac{\pi}{4} - \frac{1}{2}\arctan(\alpha\phi)\right].$$

This closed-form expression for the fractiles will be useful in determining initial estimates for the numerical methods required to determine the maximum likelihood estimates.

- Variates can be generated via inversion by

$$T \leftarrow \phi + \frac{1}{\alpha}\tan\left[\frac{\pi}{2} - (1 - U)\left(\arctan(\alpha\phi) + \frac{\pi}{2}\right)\right],$$

where U is uniformly distributed between 0 and 1.

- The conditional survivor function is given by

$$\begin{aligned} S_{T|T>a}(t) &= \frac{S_T(t)}{S_T(a)} \\ &= \frac{\arctan[\alpha(\phi - t)] + \frac{\pi}{2}}{\arctan[\alpha(\phi - a)] + \frac{\pi}{2}} \\ &= \frac{\arctan[\alpha(\phi - a) - (t - a)] + \frac{\pi}{2}}{\arctan[\alpha(\phi - a)] + \frac{\pi}{2}} \\ &= \frac{\arctan[\alpha(\gamma - y)] + \frac{\pi}{2}}{\arctan[\alpha\gamma] + \frac{\pi}{2}}. \end{aligned}$$

for $y \geq 0$, which is again an arctangent distribution with the same α as the unconditional distribution and where $\gamma = \phi - a$ and $y = t - a$.

- The limiting distribution as $\alpha \to \infty$ is a degenerate distribution at ϕ.

- The arctangent distribution is related to the Cauchy distribution. Specifically, a Cauchy distribution truncated on the left at zero will yield the arctangent distribution. Not surprisingly, the mean and higher-order moments of the distribution are undefined. This poses a challenge when discussing mean time to failure (MTTF) of components. Since the mean is undefined. central tendencies of the distribution could be discussed in terms of its mode and median. The disadvantage of this limitation is that practitioners are generally more comfortable with the mean as the primary measure of central tendency.

- The arctangent distribution has a heavy right tail which makes it useful for evaluating an item that fails with less risk once it has survived to a certain time threshold. Certain biostatistical data sets indicate such heavy right tails in cancer data. The arctangent distribution is capable of modeling lifetime distributions with a heavier tail than the log normal or log logistic. An example is provided where the distribution models the survival time of rats given a cancer accelerating drug. where there is a heavy right tail. Competing risks models are also useful in modeling heavy right tails.

A.4 Statistical inference

Statistical inference of the arctangent distribution for complete and right-censored data sets is straightforward. A comparison with other distributions, including other commonly used UBT distributions is presented in this section. The arctangent distribution requires numerical methods to determine the maximum likelihood estimators of its parameters. which is typical of most two-parameter lifetime models. First, statistical inference procedures for uncensored data are presented using a reliability example. Second, inferences are illustrated with censored data using a biostatistical data set. Finally, a biostatistical example that illustrates the heavy right-hand tail of the distribution is presented.

For the uncensored case, let $t_1, t_2, \ldots, t_n$ be the failure times. The likelihood function is

$$L(\alpha, \phi) = \prod_{i=1}^{n} f(t_i, \alpha, \phi) = \prod_{i=1}^{n} \frac{\alpha}{\left[\arctan(\alpha\phi) + \frac{\pi}{2}\right]\left[1 + \alpha^2(t_i - \phi)^2\right]}.$$

The first partial derivatives of $\log L(\alpha, \phi)$ with respect to the two parameters yield

$$\frac{\partial \log L(\alpha, \phi)}{\partial \alpha} = \frac{-n\phi}{\left[1 + (\alpha\phi)^2\right]\left[\frac{\pi}{2} + \arctan(\alpha\phi)\right]} + \frac{n}{\alpha} + \sum_{i=1}^{n} \frac{-2\alpha(t_i - \phi)^2}{\left[1 + \alpha^2(t_i - \phi)^2\right]} \tag{5}$$

and

$$\frac{\partial \log L(\alpha, \phi)}{\partial \phi} = \frac{-n\alpha}{\left[1 + (\alpha\phi)^2\right]\left[\frac{\pi}{2} + \arctan(\alpha\phi)\right]} + \sum_{i=1}^{n} \frac{2\alpha^2(t_i - \phi)}{\left[1 + \alpha^2(t_i - \phi)^2\right]}. \tag{6}$$

Equating (5) and (6) to zero does not yield closed-form solutions for the maximum likelihood estimators $\hat{\alpha}$ and $\hat{\phi}$. For the numerical methods to be effective in finding $\hat{\alpha}$ and $\hat{\phi}$ it is necessary to have appropriate initial estimates of the parameters. Since the mean and higher order moments are undefined, one must rely on a "method of fractiles", as opposed to the method of moments, to find initial estimates $\hat{\alpha}_0$ and $\hat{\phi}_0$. This entails an initial system of two equations based on the p^{th} fractile of the the distribution, where the fractiles are chosen based on the observed lifetime data.

To illustrate the applicability of the arctangent distribution, consider Lieblein and Zelen's (1956) data set of $n = 23$ ball bearing failure times (each measurement in 10^6 revolutions):

17.88	28.92	33.00	41.52	42.12	45.60	48.48
51.84	51.96	54.12	55.56	67.80	68.64	68.64
68.88	84.12	93.12	98.64	105.12	105.84	127.92
128.04	173.40					

Although it's an old data set, this is the first example because Crowder et al. (1991, p. 63) conjectured that the UBT shaped distributions may fit the ball bearing data better than the IFR distributions, based on the values of the log likelihood function at the maximum likelihood estimators.

Using the "method of fractiles" to find initial parameter estimates for the parameters, note from the empirical survivor function for the data (Figure A.2) that time 42.12 corresponds to the $\frac{5}{23} \cdot 100 = 22.7^{\text{th}}$ percentile of the distribution and that time 105.12 corresponds to the $\frac{19}{23} \cdot 100 = 82.6^{\text{th}}$ percentile of the distribution. Thus using (4), initial estimates for the MLEs are found by solving

$$42.12 = \phi + \frac{1}{\alpha}\tan\left[\frac{\pi}{2} - \left(1 - \frac{5}{23}\right)\left(\arctan(\alpha\phi) + \frac{\pi}{2}\right)\right],$$

$$105.12 = \phi + \frac{1}{\alpha}\tan\left[\frac{\pi}{2} - \left(1 - \frac{19}{23}\right)\left(\arctan(\alpha\phi) + \frac{\pi}{2}\right)\right]$$

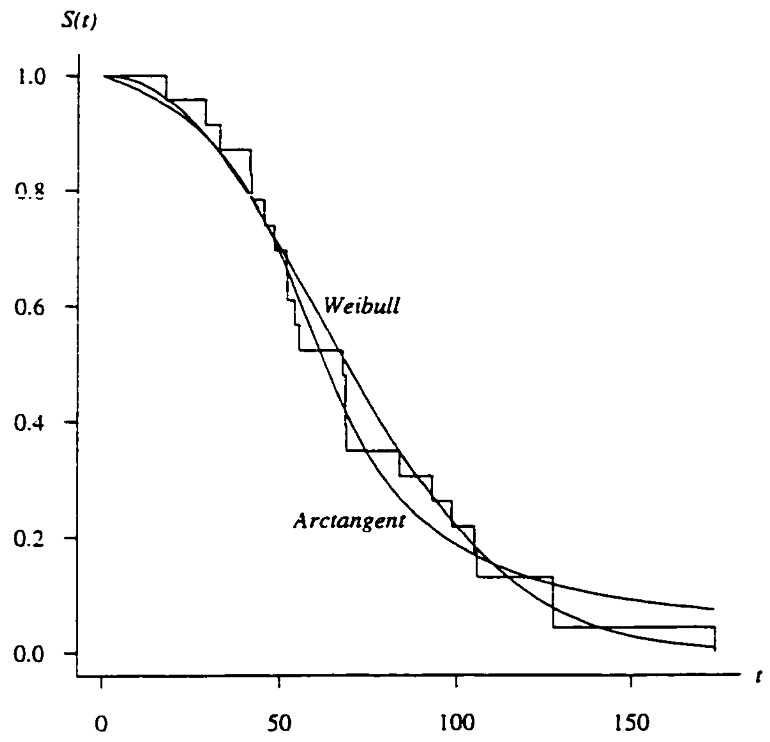


Figure A.2: Empirical, fitted arctangent, and fitted Weibull survivor functions for the ball bearing lifetimes.

for α and ϕ. This system yields our initial estimates of the parameters as follows: $\hat{\alpha}_0 = 0.04102$ and $\hat{\phi}_0 = 57.96$. Using these values as initial estimates, one may now solve equations (5) and (6) numerically yielding $\hat{\alpha} = 0.04238$ and $\hat{\phi} = 58.08$. Taking second partial derivatives of the log likelihood function and evaluating at the MLEs yields the 2 × 2 observed information matrix (see Cox and Oakes, 1984)

$$I = \begin{pmatrix} 5989 & 1.305 \\ 1.305 & 0.021 \end{pmatrix}.$$

Inverting the matrix and taking the square roots of the diagonal elements gives asymptotically valid 95% approximate confidence intervals for α and ϕ:

$$0.01727 < \alpha < 0.06753$$

$$44.46 < \phi < 71.70.$$

Figure A.2 gives a graphical comparison of the arctangent fit versus the Weibull fit of the empirical data. One can see the fits are virtually identical in the early stages, then the arctangent fits more closely than does the Weibull in the center. In the right tail, the Weibull fits closer, due to the arctangent distribution's propensity for a heavy tail. Finally, one can compare the arctangent distribution's model adequacy with that of other popular two-parameter lifetime distributions. The Kolmogorov–Smirnov (K–S) goodness-of-fit statistic for the arctangent distribution is $D_n = 0.093$. Table A.1 gives D_n values for some popular distributions fitted to the ball bearing data evaluated at the maximum likelihood estimators of their parameters. [Chhikara and Folks (1989, p. 74) fit the inverse Gaussian distribution to this data set.] The K-S statistic is one measure to gauge quality of fit. Table A.1 indicates that the lower K–S

Table A.1: Kolmogorov–Smirnov Goodness-of-fit Statistics for the Ball Bearing Data.

Distribution	D_{23}
Exponential	0.301
Weibull	0.152
Gamma	0.123
Inverse Gaussian	0.099
Arctangent	0.093
Log normal	0.090

statistic values are associated with the three UBT distributions, the inverse Gaussian, arctangent, and log normal, adding further credibility to Crowder's conjecture that UBT models seem to fit this data set better than those with increasing failure rates (i.e., the Weibull and gamma distributions). Using a K–S statistic with estimated parameters is problematic. Therefore, one uses this statistic not for a formal test, but to further the conjecture that UBT distributions fit this data set better.

Now consider statistical inference for a censored data. The arctangent distribution's closed-form survivor function yields a closed-form likelihood function, thus simplifying the analysis of right-censored data. The only other UBT distribution with this property is the log logistic distribution. The statistical methods are similar to those of the uncensored case; however, the numerical methods are a bit more tedious. An analysis follows for Gehan's (1965) test data of remission times from the drug 6-MP when used on $n = 21$ leukemia patients of which there were $r = 9$ observed remissions and 12 individuals who were randomly right censored. Letting an asterisk denote a right-censored observation, the remission times in weeks are:

6	6	6	6*	7	9*	10
10*	11*	13	16	17*	19*	20*
22	23	25*	32*	32*	34*	35*

To fit this data to the arctangent distribution, let $t_1, t_2, \ldots, t_n$ be the remission times and $c_1, c_2, \ldots, c_n$ be the associated censoring times. Our maximum likelihood estimation is now based on the likelihood function

$$\begin{aligned} L(\alpha,\phi) &= \prod_{i\in U} f(t_i,\alpha,\phi) \prod_{i\in C} S(c_i,\alpha,\phi) \\ &= \prod_{i\in U} \frac{\alpha}{\left[\arctan(\alpha\phi)+\frac{\pi}{2}\right]\left[1+\alpha^2(t_i-\phi)^2\right]} \prod_{i\in C} \frac{\arctan(\alpha(\phi-c_i))+\frac{\pi}{2}}{\arctan(\alpha\phi)+\frac{\pi}{2}}. \end{aligned}$$

where U and C are the sets of indices of uncensored and censored observations, respectively. The log likelihood function is

$$\begin{aligned} \log L(\alpha,\phi) &= r\log\alpha - r\log\left[\arctan(\alpha\phi)+\frac{\pi}{2}\right] - \sum_{i\in U}\log[1+\alpha^2(t_i-\phi)^2] \\ &+ \sum_{i\in C}\log[2\arctan[\alpha(\phi-c_i)]+\pi] - (n-r)\log[2\arctan(\alpha\phi)+\pi]. \end{aligned}$$

A "method of fractiles" initial estimate for the parameters yields $\hat{\alpha}_0 = 0.0562$ and $\hat{\phi}_0 = 9.58$. Now one takes the two partial derivatives of $\log L$ with respect to α and ϕ, sets them equal to zero, and computes $\hat{\alpha} = 0.0455$ and $\hat{\phi} = 11.2$. This example illustrates the methodology used to fit the arctangent distribution to censored data sets.

A third example illustrates the usefulness of the distribution's heavy right tail. Cox and Snell (1981, p. 169) present data on the life span of rats who have been given a cancer accelerator. The following complete data set gives the number of days the rats survived:

37 38 42 43 43 43 43 43 48 49 51 51 55 57 59 62 66 69 86 177

The arctangent MLE estimates for this data set are $\hat{\alpha} = 0.127$ and $\hat{\phi} = 48.0$: the resulting arctangent distribution is plotted against the empirical survivor function in Figure A.3. Note how the heavy right tail of the arctangent distribution models the

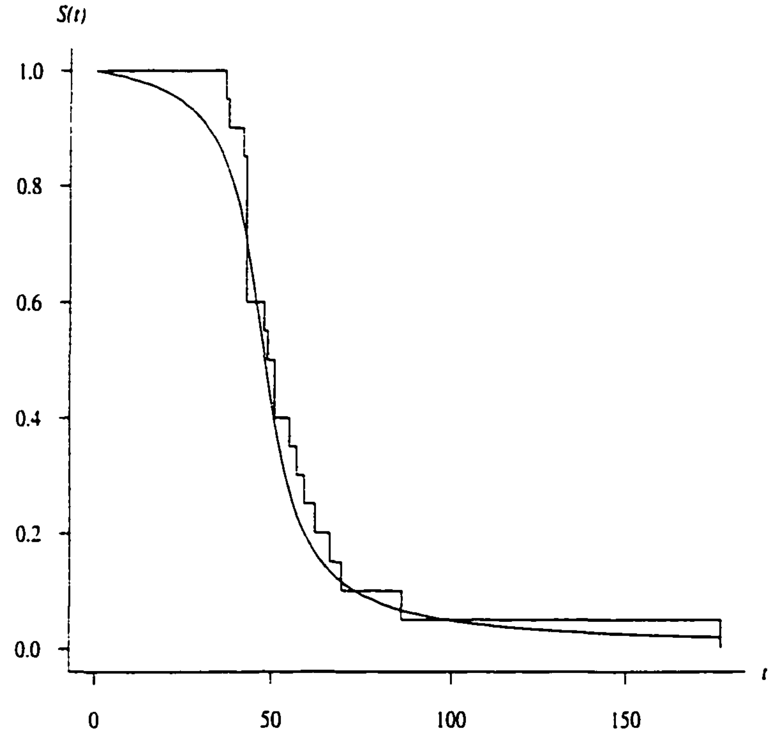


Figure A.3: The arctangent distribution fit to the rat cancer data.

heavy right tail of the rat lifetimes. The rat with survival time 177 is driving the fit. Thus, for data sets with heavy tails such as this one, the practitioner may make use of the arctangent distribution's propensity for a heavy right tail.

A.5 Conclusion

The arctangent distribution is a two-parameter lifetime distribution in the UBT class with closed-form survivor function. It gives reliability engineers, biostatisticians, and others another tool in the complex task of statistical modeling. Although a UBT model has a smaller number of applications than does the IFR or bathtub-shaped models, there are enough references in the literature to indicate a need for more distributions in this class. This appendix gives the UBT model a second distribution that enjoys a closed-form survivor function and has been demonstrated to adequately describe well-known data sets.

Appendix B

Continuous Distributions

The continuous distributions given in Table B.1 are available when loading the file `contdist.map`. The table contains the function name used to define a random variable, the PDF or CDF, the support, and the parameter names, and parameter restrictions. Original references to the more obscure distributions are given in Leemis (1995). Note that the `IDBRV` is entered as a CDF instead of a PDF in Table B.1. Maple is able to differentiate the CDF into a PDF, but not able to reverse this operation. Converting from a PDF to a CDF for this distribution results in an unevaluated integral, which considerably slows computations. Random variables in APPL defined in this fashion are in the usual list-of-lists data structure, and the Maple **assume** function is used to restrict the values of unspecified parameters.

Table B.1: Continuous distributions of random variables available in APPL.

Distribution name	PDF or CDF	Support	Parameter Space
ErlangRV()	$f_X(x) = \lambda(\lambda x)^{n-1}e^{-\lambda x}/(n-1)!$	$x > 0$	$0 < \lambda < \infty$, integer n
ExponentialRV()	$f_X(x) = \lambda e^{-\lambda x}$	$x > 0$	$\lambda > 0$
ExponentialPowerRV()	$f_X(x) = e^{1-e^{\lambda x^\kappa}} e^{\lambda x^\kappa} \lambda\kappa x^{\kappa-1}$	$x > 0$	$\lambda > 0;\ \kappa > 0$
GammaRV()	$f_X(x) = \lambda(\lambda x)^{\kappa-1}e^{-\lambda x}/\Gamma(\kappa)$	$x > 0$	$\lambda > 0;\ \kappa > 0$
GeneralizedParetoRV()	$f_X(x) = \left(\gamma + \frac{\kappa}{x+\delta}\right)(1 + x/\delta)^{-\kappa}e^{-\gamma x}$	$x > 0$	$\delta > 0$, $\gamma \geq 0, \kappa \geq -\delta\gamma$
GompertzRV()	$f_X(x) = \delta\kappa^x e^{-\delta(\kappa^{x-1})/\log\kappa}$	$x > 0$	$\delta > 0;\ \kappa > 1$
IDBRV()	$F_X(x) = 1 - (1 + \kappa x)^{-\gamma/\kappa}e^{-\delta x^2/2}$	$x > 0$	γ, δ, κ all ≥ 0
InverseGaussianRV()	$f_X(x) = \sqrt{\lambda/(2\pi x^3)}e^{-\lambda(x-\mu)^2/(2\mu^2 x)}$	$x > 0$	$\lambda > 0,\ \mu > 0$
LogNormalRV()	$f_X(x) = \frac{1}{\sqrt{2\pi}x\sigma}e^{-(\log x - \mu)^2/(2\sigma^2)}$	$x > 0$	$-\infty < \mu < \infty,\ \sigma > 0$
MakehamRV()	$f_X(x) = (\gamma + \delta\kappa^x)e^{-\gamma x - \delta(\kappa^x - 1)/\log\kappa}$	$x > 0$	$\gamma, \delta > 0,\ \kappa > 1$
MuthRV()	$f_X(x) = (\epsilon^{\kappa x} - \kappa)e^{-e^{\kappa x}/\kappa + \kappa x + 1/\kappa}$	$x > 0$	$0 < \kappa \leq 1$
NormalRV()	$f_X(x) = e^{-(x-\mu)^2/\sigma^2}/\sqrt{2\pi\sigma^2}$	$-\infty < x < \infty$	$-\infty < \mu < \infty,\ \sigma > 0$
ParetoRV()	$f_X(x) = \kappa\lambda^\kappa/x^{\kappa+1}$	$x > \lambda$	$\lambda > 0$
TriangularRV()	$f_X(x) = \begin{cases} \frac{2(x-a)}{(b-a)(m-a)} \\ \frac{2(b-x)}{(b-a)(b-m)} \end{cases}$	$a < x < m$ $m < x < b$	$-\infty < a < m < b < \infty$
UniformRV()	$f_X(x) = 1/(b-a)$	$a < x < b$	$-\infty < a < b < \infty$
WeibullRV()	$f_X(x) = \kappa\lambda^\kappa x^{\kappa-1}e^{-(\lambda x)^\kappa}$	$x > 0$	$\lambda > 0,\ \kappa > 0$

Appendix C

Algorithm for 6×6 Conversions

Table C.1 shows the mathematical conversions of the functional forms of random variable distributions. The table can be viewed as a from-to matrix in that the columns represent the procedure name and the rows represent the prior form of the distribution. Thus, if X is currently in CDF form, the call to `X := PDF(X)` will convert to the CDF form via the conversion listed in the PDF column on the CDF row.

Table C.1: The conversions of continuous random variable distribution forms.

	Procedure call with argument X					
Prior Form	`PDF(X)`	`CDF(X)`	`SF(X)`	`HF(X)`	`CHF(X)`	`IDF(X)`
PDF	Return(X)	$\int_{-\infty}^{x} f(t)dt$	$\int_{x}^{\infty} f(t)dt$	$\frac{f(x)}{\int_{x}^{\infty} f(t)dt}$	$-\ln\left[\int_{x}^{\infty} f(t)dt\right]$	$F^{-1}(x)$
CDF	$F'(x)$	Return(X)	$1 - F(x)$	$\frac{F'(x)}{1-F(x)}$	$-\ln[1 - F(x)]$	$F^{-1}(x)$
SF	$-S'(x)$	$1 - S(x)$	Return(X)	$\frac{-S'(x)}{S(x)}$	$-\ln[S(x)]$	$F^{-1}(x)$
HF	$h(x)e^{-\int_{-\infty}^{x} h(t)dt}$	$1 - e^{-\int_{-\infty}^{x} h(t)dt}$	$e^{-\int_{-\infty}^{x} h(t)dt}$	Return(X)	$\int_{-\infty}^{x} h(t)dt$	$F^{-1}(x)$
CHF	$H'(x)e^{-H(x)}$	$1 - e^{-H(x)}$	$e^{-H(x)}$	$H'(x)$	Return(X)	$F^{-1}(x)$
IDF	$F'(x)$	$F(x)$	$1 - F(x)$	$\frac{F'(x)}{1-F(x)}$	$-\ln[1 - F(x)]$	Return(X)

Note that the conversions to and from the IDF require each functional form to be converted to a CDF first. as that form is the necessary form for inversion. Thus $F^{-1}(x)$ appreas in the `PDF, IDF(X)` entry of Table C.1 rather than the more complicated $\left[\int_{-\infty}^{x} f(t)dt\right]^{-1}$.

Appendix D

Algorithms for Various Procedures

D.1 Algorithm for `VerifyPDF`

Procedure `VerifyPDF` verifies the area under a PDF is one.

`Input:` A list-of-lists represented random variable X with n segments.

`Output:` "True" or "False" depending if the area is equal to one.

```
$X \leftarrow$ PDF(X)
$area \leftarrow 0$
for $i \leftarrow 1$ to $n$
    $area \leftarrow area + \int_{X_{2,i}}^{X_{2,i+1}} X_{1,i}(x)\,dx$
if ($area > 0.9999999$ and $area < 1.0000001$)
    RETURN(True)
else
    RETURN(False)
```

D.2 Algorithm for ExpectationRV

Procedure `ExpectationRV(X, g)` finds the expected value of a function of a random variable, i.e.. $E[g(x)]$.

Input: A list-of-lists represented random variable X with n segments and a function $g(x)$.

Output: The expected value of the function.

```
$X$ ← PDF(X)
$expval$ ← 0
for $i$ ← 1 to $n$
    $expval \leftarrow expval + \int_{X_{2,i}}^{X_{2,i+1}} X_{1,i}(x) \cdot g(x)\, dx$
RETURN($expval$)
```

D.3 Algorithm for OrderStat

Procedure OrderStat(X, n, r) determines the PDF for the r^{th} order statistic from a sample size n sampled randomly from a population with the same distribution as the random variable X.

Input: A list-of-lists represented random variable X with m segments and the integers n and r.

Output: The PDF of the order statistic $X_{(r)}$ in the list-of-lists format.

```
$fX \leftarrow$ PDF(X)
$FX \leftarrow$ CDF(X)
for $i \leftarrow 1$ to $m$
    $X_{(r)1,i}(y) \leftarrow \frac{n!}{(r-1)!(n-r)!} FX_{1,i}(y)^{r-1} \cdot (1 - FX_{1,i}(y))^{n-r} \cdot fX_{1,i}(y)$
RETURN($X_{(r)}$)
```

D.4 Algorithm for ProdIID

Procedure `ProdIID(X, n)` determines the PDF for the product of n iid random variables X.

Input: A list-of-lists represented random variable X and the integer $n > 1$.

Output: The PDF of $Y = X^n$ in the list-of-lists format.

```
$X \leftarrow$ PDF(X)
$Y \leftarrow$ ProductRV(X.X)
for $i \leftarrow 3$ to $n$
    $Y \leftarrow$ ProductRV($X$, $Y$)
RETURN($Y$)
```

D.5 Algorithm for SumRV

Procedure SumRV(X, Y) determines the PDF for the sum of the independent random variables X and Y.

Input: Independent random variables X and Y in the list-of-lists format.

Output: The PDF of $V = X + Y$ in the list-of-lists format.

```
$W \leftarrow e^X$                                  [using Transform]
$Z \leftarrow e^Y$                                  [using Transform]
$V \leftarrow \ln(W \cdot V)$           [using ProductRV and Transform]
RETURN($V$)
```

D.6 Algorithm for SumIID

Procedure SumIID determines the PDF for the sum of n iid random variables $X_1, X_2, \ldots, X_n$, each having the same distribution as X.

Input: A list-of-lists represented random variable X with and the integer $n > 1$.

Output: The PDF of $Y = \sum_{i=1}^{n} X_i$ in the list-of-lists format.

```
$X \leftarrow$ PDF($X$)
$Y \leftarrow$ SumRV($X, X$)
for $i \leftarrow 3$ to $n$
    $Y \leftarrow$ SumRV($X$, $Y$)
RETURN($Y$)
```

D.7 Algorithm for MinimumRV

Procedure MinimumRV determines the PDF for the minimum of two independent random variables.

Input: Two list-of-lists represented random variables X and Y.

Output: The PDF of $V = \min\{X, Y\}$ in the list-of-lists format.

```
$fX \leftarrow$ PDF($X$)
$fY \leftarrow$ PDF($Y$)
$fV_2 \leftarrow fX_2 \cup fY_2$
$highest \leftarrow \min\{\max\{fX_2\}, \max\{fY_2\}\}$ [Delete elements above lower support max]
$stoppoint \leftarrow ||fV_2||$                         [where $||\cdot||$ denotes cardinality]
for $i \leftarrow 1$ to $stoppoint$                     [Identify the max support of the new RV]
   if $fV_{2,i} = highest$
      $highesti \leftarrow i$
      break
if $highesti \neq fV_{2,stoppoint}$                     [Remove all values too high from support list]
   for $j \leftarrow (highesti + 1)$ to $stoppoint$
      $fV_{2,j} \leftarrow$ NULL
$FX \leftarrow$ CDF($fX$)
$FY \leftarrow$ CDF($fY$)
$nsegments \leftarrow ||fV_2|| - 1$
$Xindex \leftarrow 1$
$Yindex \leftarrow 1$
for $i \leftarrow 1$ to $nsegments$
   if $fV_{2,i} < fX_{2,1}$
      $currFX(x) \leftarrow 0$
   else if $fV_{2,i} = fX_{2,Xindex}$
      $currFX(x) \leftarrow FX_{1,Xindex}$
      $Xindex \leftarrow Xindex + 1$
   if $fV_{2,i} < fY_{2,1}$
      $currFY(x) \leftarrow 0$
   else if $fV_{2,i} = fY_{2,Yindex}$
      $currFY \leftarrow FY_{1,Yindex}$
      $Yindex \leftarrow Yindex + 1$
   $fV_{1,i} \leftarrow 1 - (1 - currFX(x)) \cdot (1 - currFY(x))$
   $fV_{1,i} \leftarrow \frac{d fV_{1,i}(x)}{dx}$
RETURN($fV$)
```

D.8 Algorithm for MaximumRV

Procedure MaximumRV determines the PDF for the maximum of two independent random variables.

Input: Two list-of-lists represented random variables X and Y.

Output: The PDF of $V = \max\{X, Y\}$.

```
$fX \leftarrow$ PDF($X$)
$fY \leftarrow$ PDF($Y$)
$fnegX \leftarrow -X$                              [using Transform]
$fnegY \leftarrow -Y$                              [using Transform]
$fV \leftarrow$ MinimumRV($fnegX, fnegY$)
$fV \leftarrow -fV$                                [using Transform]
RETURN($fV$)
```

D.9 Algorithm for MLE

Procedure MLE determines the maximum likelihood parameter estimates associated with a random sample drawn from a population with presumed underlying distribution X.

Input: The list-of-lists represented random variable X, the list of data values s, and the list of unknown parameters p.

Output: The maximum likelihood estimates of each element in p.

```
$fX \leftarrow$ PDF($X$)
$L \leftarrow 0$                                   [initialize the log-likelihood function]
$n \leftarrow ||s||$
for $i \leftarrow 1$ to $n$
    $L \leftarrow L + \ln(fX_{1,1}(s_i))$
$npars \leftarrow ||p||$
for $j \leftarrow 1$ to $npars$
    $L'_j \leftarrow \frac{\partial L}{\partial p_j}$                         [set up the derivatives]
[Using Maple's solve command, solve the system of equations below]
```

```
eqnset := {seq(L'[i] = 0, i = 1 .. npars)};
paramset := {seq(p[i], i = 1 .. npars)};
solns := solve(eqnset, paramset);
```

RETURN(solns)

Appendix E

Algorithm for Transform

Introduced in Chapter 3, the following is the algorithm for the `Transform` procedure.

`Input`: The pdf of X, $f(x) = f_p(x)$ for $z_p < x < z_{p+1}$, $p = 1, 2, \ldots, q$.
The transformation $g(x) = g_i(x)$ for $x_i < x < x_{i+1}$, where $g_i(x)$ is monotone and not defined piecewise on $x_i < x < x_{i+1}$, and for some p, $[x_i, x_{i+1}] \subseteq [z_p, z_{p+1}]$, $i = 1, 2, \ldots, n$.

`Output`: The pdf of Y, $h(y) = h_j(y)$ for $y_j < y < y_{j+1}$, $j = 1, 2, \ldots, m$.

```
$X^* \leftarrow \{x_1, x_2, \ldots, x_{n+1}\}$                    [by definition, $X^*$ is a sorted list]
For $i \leftarrow 1$ to $n$
    $FF(i) \leftarrow k$                                    [where $f(x) = f_k(x)$ on $(x_i, x_{i+1})$]
If $x_1 = -\infty$ and $x_2 = \infty$ then
    $c_1 \leftarrow 0$
else
    $BEGIN \leftarrow 1$
    $END \leftarrow n$
    If $x_1 = -\infty$ then
        $c_1 \leftarrow x_2 - 1$
        $BEGIN \leftarrow 2$
    If $x_{n+1} = \infty$ then
        $c_n \leftarrow x_n + 1$
        $END \leftarrow n - 1$
    For $i \leftarrow BEGIN$ to $END$
        $c_i \leftarrow (x_i + x_{i+1})/2$
```

```
For $i \leftarrow 1$ to $n$
    $a \leftarrow \lim_{x \downarrow x_i} g_i(x)$
    $b \leftarrow \lim_{x \uparrow x_{i+1}} g_i(x)$
    $m_i \leftarrow \min\{a, b\}$
    $M_i \leftarrow \max\{a, b\}$

$Y^* \leftarrow \cup_{i=1}^{n} \{m_i, M_i\}$                                        [$Y^*$ is a sorted list]
$m \leftarrow |Y^*| - 1$

For $j \leftarrow 1$ to $m$
    $h(y) \leftarrow 0$
    For $i \leftarrow 1$ to $n$
        If $m_i \leq y_j$ and $y_{j+1} \leq M_i$
            Find $g_i^{-1}$ such that $g_i^{-1}(g_i(c_i)) = c_i$
            $h(y) \leftarrow h(y) + f_{FF(i)}(g_i^{-1}(y)) \cdot |\frac{d}{dy} g_i^{-1}(y)|$
    $h_j(y) \leftarrow h(y)$
```

Appendix F

Algorithm for ProductRV

As introduced in Chapter 4, the following is the algorithm for the ProductRV procedure.

Input: The PDF of X, $f(x)$, and the pdf of Y, $g(y)$, where X and Y are independent, continuous random variables. PDFs are in the list-of-lists format: e.g., $[fX_1, fX_2, [\text{'Continuous','PDF'}]]$.

Output: The PDF of $V = X \cdot Y$, $h(v)$, in the list-of-lists format. The PDF is determined by the general result

$$h(v) = \int_{-\infty}^{\infty} f(x) \cdot g\left(\frac{v}{x}\right) \frac{1}{|x|} dx.$$

```
$X^* \leftarrow fX_2$ (from the list-of-lists format), $n \leftarrow ||X^*||$
$Y^* \leftarrow fY_2, m \leftarrow ||Y^*||$
$f \leftarrow fX_1$
$g \leftarrow fY_1$
$V^* \leftarrow [\,]$, (the empty list)
$h \leftarrow [\,]$
If ($X_1^* < 0$ and $X_n^* > 0$ and $0 \notin X^*$) then          [Insert 0 into $X^*$ if necessary]
    For $i \leftarrow 1$ to $n$
        If ($X_i^* < 0$ and $X_{i+1}^* > 0$) then
            Insert 0 between positions $X_i^*$ and $X_{i+1}^*$
            Insert $f_i$ at position $f_{i+1}$
            $n \leftarrow n + 1$
            break
$If (Y_1^* < 0$ and $Y_m^* > 0$ and $0 \notin Y^*)$ then            [Insert 0 into $Y^*$ if necessary]
    For $i \leftarrow 1$ to $m$
        If $(Y_i^* < 0$ and $Y_{i+1}^* > 0)$ then
            Insert 0 between positions $Y_i^*$ and $Y_{i+1}^*$
            Insert $g_i$ at position $g_{i+1}$
            $m \leftarrow m + 1$
            break
For $i \leftarrow 1$ to $n$                                  [Set up the support list $V^*$]
    For $j \leftarrow 1$ to $m$
        $V^* \leftarrow V^* \cup (X_i^* \cdot Y_j^*)$
$l \leftarrow \|V^*\|$
For $i \leftarrow 1$ to $l - 1$
    $h_i \leftarrow 0$
For $i \leftarrow 1$ to $n - 1$          [Integrate each segment of $X$ times each segment of $Y$]
    For $j \leftarrow 1$ to $m - 1$
        $a \leftarrow X_i^*$
        $b \leftarrow X_{i+1}^*$
        $c \leftarrow Y_j^*$
        $d \leftarrow Y_{j+1}^*$
        $xy \leftarrow X_i^* \cdot Y_j^*$
        If $(X_i^* \geq 0$ and $xy \geq 0)$ then
            $f1 \leftarrow \int_a^b f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$                  [1st Quadrant]
            If $(d < \infty)$ then $f2 \leftarrow \int_{v/d}^b f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
            If $(c > 0)$ then $f3 \leftarrow \int_a^{v/c} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
            If $(c > 0$ and $d < \infty$ and $ad < bc)$ then $f4 \leftarrow \int_{v/d}^{v/c} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
            If $(c = 0$ and $d = \infty)$ then                   [1st quadrant, Scenario A]
                For $ii \leftarrow 1$ to $l - 1$
                    If $(V_{ii}^* \geq 0$ and $V_{ii+1}^* \leq \infty)$ then
                        $h_{ii} \leftarrow h_{ii} + f1$
            If $(c = 0$ and $d < \infty)$ then                   [1st quadrant, Scenario B]
                For $ii \leftarrow 1$ to $l - 1$
                    If $(V_{ii}^* \geq 0$ and $V_{ii+1}^* \leq ad)$ then
                        $h_{ii} \leftarrow h_{ii} + f1$
                    If $(V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq bd)$ then
                        $h_{ii} \leftarrow h_{ii} + f2$
            If $(c > 0$ and $d = \infty)$ then                   [1st quadrant, Scenario C]
                For $ii \leftarrow 1$ to $l - 1$
                    If $(V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq \infty)$ then
                        $h_{ii} \leftarrow h_{ii} + f1$
                    If $(V_{ii}^* \geq ac$ and $V_{ii+1}^* \leq bc)$ then
                        $h_{ii} \leftarrow h_{ii} + f3$
    If ($c > 0$ and $d < \infty$) then                                    [1st quadrant, Scenario D]
      If ($ad < bc$) then
        For $ii \leftarrow 1$ to $l-1$                                                  [1st Case]
          If ($V_{ii}^* \geq ac$ and $V_{ii+1}^* \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq bc$) then
            $h_{ii} \leftarrow h_{ii} + f4$
          If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq bd$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ad = bc$) then
        For $ii \leftarrow 1$ to $l-1$                                                  [2nd Case]
          If ($V_{ii}^* \geq ac$ and $V_{ii+1}^* \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq bd$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ad > bc$) then
        For $ii \leftarrow 1$ to $l-1$                                                  [3rd Case]
          If ($V_{ii}^* \geq ac$ and $V_{ii+1}^* \leq bc$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f1$
          If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq bd$) then
            $h_{ii} \leftarrow h_{ii} + f2$
  If ($X_i^* < 0$ and $xy \geq 0$) then
    $f1 \leftarrow -\int_a^b f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x} dx$                                   [2nd Quadrant]
    If ($d < 0$) then $f2 \leftarrow -\int_{v/d}^{b} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x} dx$
    If ($c > -\infty$) then $f3 \leftarrow -\int_a^{v/c} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x} dx$
    If ($c > -\infty$ and $d < 0$ and $ad > bc$) then $f4 \leftarrow -\int_{v/d}^{v/c} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x} dx$
    If ($c = -\infty$ and $d = 0$) then                                   [2nd quadrant, Scenario A]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq 0$ and $V_{ii+1}^* \leq \infty$) then
          $h_{ii} \leftarrow h_{ii} + f1$
    If ($c = -\infty$ and $d < 0$) then                                   [2nd quadrant, Scenario B]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq \infty$) then
          $h_{ii} \leftarrow h_{ii} + f1$
        If ($V_{ii}^* \geq bd$ and $V_{ii+1}^* \leq ad$) then
          $h_{ii} \leftarrow h_{ii} + f2$
    If ($c > -\infty$ and $d = 0$) then                                   [2nd quadrant, Scenario C]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq 0$ and $V_{ii+1}^* \leq bc$) then
          $h_{ii} \leftarrow h_{ii} + f1$
        If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq ac$) then
```

```
            $h_{ii} \leftarrow h_{ii} + f3$
    If ($c > -\infty$ and $d < 0$) then                              [2nd quadrant, Scenario D]
      If ($ad > bc$) then
        For $ii \leftarrow 1$ to $l-1$                                             [1st Case]
          If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f4$
          If ($V_{ii}^* \geq bd$ and $V_{ii+1}^* \leq bc$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ad = bc$) then
        For $ii \leftarrow 1$ to $l-1$                                             [2nd Case]
          If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq bd$ and $V_{ii+1}^* \leq bc$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ad < bc$) then
        For $ii \leftarrow 1$ to $l-1$                                             [3rd Case]
          If ($V_{ii}^* \geq bc$ and $V_{ii+1}^* \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq bc$) then
            $h_{ii} \leftarrow h_{ii} + f1$
          If ($V_{ii}^* \geq bd$ and $V_{ii+1}^* \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f2$
  If ($X_i^* < 0$ and $xy < 0$) then
    $f1 \leftarrow -\int_a^b f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$                [3rd Quadrant]
    If ($d < \infty$) then $f2 \leftarrow -\int_a^{v/d} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
    If ($c > 0$) then $f3 \leftarrow -\int_{v/c}^{b} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
    If ($c > 0$ and $d < \infty$ and $bd > ac$) then $f4 \leftarrow -\int_{v/c}^{v/d} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
    If ($c = 0$ and $d = \infty$) then                               [3rd quadrant, Scenario A]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq -\infty$ and $V_{ii+1}^* \leq 0$) then
          $h_{ii} \leftarrow h_{ii} + f1$
    If ($c = 0$ and $d < \infty$) then                               [3rd quadrant, Scenario B]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq bd$ and $V_{ii+1}^* \leq 0$) then
          $h_{ii} \leftarrow h_{ii} + f1$
        If ($V_{ii}^* \geq ad$ and $V_{ii+1}^* \leq bd$) then
          $h_{ii} \leftarrow h_{ii} + f2$
    If ($c > 0$ and $d = \infty$) then                               [3rd quadrant, Scenario C]
      For $ii \leftarrow 1$ to $l-1$
        If ($V_{ii}^* \geq -\infty$ and $V_{ii+1}^* \leq ac$) then
          $h_{ii} \leftarrow h_{ii} + f1$
```

```
            If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq bc$) then
              $h_{ii} \leftarrow h_{ii} + f3$
        If ($c > 0$ and $d < \infty$) then                    [3rd quadrant, Scenario D]
          If ($bd > ac$) then
            For $ii \leftarrow 1$ to $l - 1$                                  [1st Case]
              If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq bc$) then
                $h_{ii} \leftarrow h_{ii} + f3$
              If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq bd$) then
                $h_{ii} \leftarrow h_{ii} + f4$
              If ($V^*_{ii} \geq ad$ and $V^*_{ii+1} \leq ac$) then
                $h_{ii} \leftarrow h_{ii} + f2$
          If ($ac = bd$) then
            For $ii \leftarrow 1$ to $l - 1$                                  [2nd Case]
              If ($V^*_{ii} \geq ad$ and $V^*_{ii+1} \leq ac$) then
                $h_{ii} \leftarrow h_{ii} + f2$
              If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq bc$) then
                $h_{ii} \leftarrow h_{ii} + f3$
          If ($ac > bd$) then
            For $ii \leftarrow 1$ to $l - 1$                                  [3rd Case]
              If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq bc$) then
                $h_{ii} \leftarrow h_{ii} + f3$
              If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq ac$) then
                $h_{ii} \leftarrow h_{ii} + f1$
              If ($V^*_{ii} \geq ad$ and $V^*_{ii+1} \leq bd$) then
                $h_{ii} \leftarrow h_{ii} + f2$
    If ($X^*_i \geq 0$ and $xy < 0$) then
        $f1 \leftarrow \int_a^b f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$           [4th Quadrant]
        If ($d < 0$) then $f2 \leftarrow \int_a^{v/d} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
        If ($c > -\infty$) then $f3 \leftarrow \int_{v/c}^{b} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
        If ($c > -\infty$ and $d < 0$ and $ac > bd$) then $f4 \leftarrow \int_{v/c}^{v/d} f_i(x) \cdot g_j(\frac{v}{x}) \cdot \frac{1}{x}dx$
        If ($c = -\infty$ and $d = 0$) then                   [4th quadrant, Scenario A]
          For $ii \leftarrow 1$ to $l - 1$
            If ($V^*_{ii} \geq -\infty$ and $V^*_{ii+1} \leq 0$) then
              $h_{ii} \leftarrow h_{ii} + f1$
        If ($c = -\infty$ and $d < 0$) then                   [4th quadrant, Scenario B]
          For $ii \leftarrow 1$ to $l - 1$
            If ($V^*_{ii} \geq -\infty$ and $V^*_{ii+1} \leq bd$) then
              $h_{ii} \leftarrow h_{ii} + f1$
            If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq ad$) then
              $h_{ii} \leftarrow h_{ii} + f2$
        If ($c > -\infty$ and $d = 0$) then                   [4th quadrant, Scenario C]
          For $ii \leftarrow 1$ to $l - 1$
            If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq 0$) then
          $h_{ii} \leftarrow h_{ii} + f1$
        If ($V^*_{ii} \geq bc$ and $V^*_{ii+1} \leq ac$) then
          $h_{ii} \leftarrow h_{ii} + f3$
    If ($c > -\infty$ and $d < 0$) then                    [4th quadrant, Scenario D]
      If ($ac > bd$) then
        For $ii \leftarrow 1$ to $l - 1$                   [1st Case]
          If ($V^*_{ii} \geq bc$ and $V^*_{ii+1} \leq bd$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f4$
          If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ad = bc$) then
        For $ii \leftarrow 1$ to $l - 1$                   [2nd Case]
          If ($V^*_{ii} \geq bc$ and $V^*_{ii+1} \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f2$
      If ($ac < bd$) then
        For $ii \leftarrow 1$ to $l - 1$                   [3rd Case]
          If ($V^*_{ii} \geq bc$ and $V^*_{ii+1} \leq ac$) then
            $h_{ii} \leftarrow h_{ii} + f3$
          If ($V^*_{ii} \geq ac$ and $V^*_{ii+1} \leq bd$) then
            $h_{ii} \leftarrow h_{ii} + f1$
          If ($V^*_{ii} \geq bd$ and $V^*_{ii+1} \leq ad$) then
            $h_{ii} \leftarrow h_{ii} + f2$
$fV_1 \leftarrow h$                                        [Make a new list-of-lists for $V$]
$fV_2 \leftarrow V^*$
$fV_3 \leftarrow$ ['Continuous','PDF']
Return($fV$)
```

Appendix G

Algorithm for KSRV

Procedure KSRV: Computing the CDF of the Kolmogorov–Smirnov test statistic.

Input: A positive integer n, the sample size associated with the KS test statistic when all parameters are known.

Output: The CDF of the KS test statistic when all parameters are known is returned. The random variable will be in the usual list-of-lists format.

[**Phase 1:** Determine the endpoints of the segments that define the support of the KS test statistic.]

```
Dimension $v\left[0..\left\lceil\frac{3n}{2}\right\rceil - 1\right]$        [$v$ holds the partition values for the support of $D_n$]
$m \leftarrow 0$
$v_m \leftarrow 0$
$g \leftarrow \frac{1}{2n}$        [$g$ is the gap between adjacent integration limits when $v = 0$]
For $i \leftarrow 1$ to $n - 1$ by 1
    $m \leftarrow m + 1$
    $v_m \leftarrow i \cdot g$
For $j \leftarrow 2 \cdot \lfloor\frac{n}{2}\rfloor + 1$ to $2n - 1$ by 2        [Lower limit is $n$ or $n + 1$, whichever is odd]
    $m \leftarrow m + 1$
    $v_m \leftarrow j \cdot g$
```

[**Phase 2:** The next phase of the algorithm has three parts to it.

1. Define $c_1, c_2, \ldots, c_m$ as the midpoints of the support intervals

2. Define $x_1, x_2, \ldots, x_n$ as the limits of integration when $v = 0$

3. Define the $n \times n$ indicator matrices $A_1, A_2, \ldots, A_m$.]

```
Dimension $c[m]$          [Define vector $c$ of midpoints of the support subintervals]
For $k \leftarrow 1$ to $m$
    $c_k \leftarrow (v_{k-1} + v_k)/2$

Dimension $x[n]$                 [Define vector $x$ of centers of integration limits]
For $i \leftarrow 1$ to $n$
    $x_i \leftarrow (2i - 1) \cdot g$

Dimension $A[n, n]$

For $i \leftarrow 2$ to $n$
    For $j \leftarrow 1$ to $i - 1$
        $A_{ij} \leftarrow 0$                             [Zero out lower triangle of $A$]

For $k \leftarrow 1$ to $m$                   [Loop through all subintervals of support]
    For $i \leftarrow 1$ to $n$
        For $j \leftarrow i$ to $n$
            $A_{ij} \leftarrow 0$                         [Zero out upper triangle of $A$]
    $z \leftarrow \max\{\lfloor nc_k - \frac{1}{2}\rfloor, 0\}$          [$z$ is the number of leading zero columns in $A$]
    $l \leftarrow \min\{\lceil 2nc_k\rceil, n\}$              [$l$ is the number of $u$-subintervals that intersect
$\left(\frac{1}{2n} - v, \frac{1}{2n} + v\right)$]
    For $i \leftarrow 1$ to $n$
        For $j \leftarrow \max\{i, z + 1\}$ to $\min\{n, i + l - 1\}$
            $A_{ij} \leftarrow 1$                                      [Place 1's in $A$]
```

[**Phase 3:** Compute the fixed and variable limit matrices F and V and combine to give the piecewise polynomial CDF]

```
Dimension $P[m]$      [$n$th order polynomials for CDF segments, parameterized by $v$]
Dimension $F[n, n]$              [Array of Fixed integrals, parameterized by $v$]
Dimension $V[n, n]$           [Array of Variable integrals, parameterized by $v$]
Real $S(v)$                              [A function of $v$, used as a running sum]
  [$S(v)$ is the sum of all $F$ elements one row down, starting one element to the right]

$P_1(v) \leftarrow n!(2v)^n$          [$P_1(v) \leftarrow n! \int_{x_1-v}^{x_1+v} \int_{x_2-v}^{x_2+v} \cdots \int_{x_n-v}^{x_n+v} 1\, du_n \ldots du_2\, du_1$]
For $k \leftarrow 2$ to $m$                   [Loop through all subintervals of support]
    $z \leftarrow \max\{\lfloor nc_k - \frac{1}{2}\rfloor, 0\}$
    $l \leftarrow \min\{\lceil 2nc_k\rceil, n\}$
    $F_{n,n}(v) \leftarrow \int_{x_n-v}^{1} 1\, du_n$
    $V_{n,n}(v) \leftarrow \int_{u_{n-1}}^{1} 1\, du_n$
```

```
For $i \leftarrow n-1$ to 2 by $-1$
    If $i+l > n$ then
        $S(v) \leftarrow 0$
    else
        $S(v) \leftarrow F_{i+1,i+l}(v)$
    If $i+l > n+1$ then
        $F_{i,n}(v) \leftarrow \int_{x_n-v}^{\lfloor x_i+c_k \rfloor+(x_i+v)(1-\lfloor x_i+c_k \rfloor)} V_{i+1,n}(v)\,du_i$
        $V_{i,n}(v) \leftarrow \int_{u_{i-1}}^{\lfloor x_i+c_k \rfloor+(x_i+v)(1-\lfloor x_i+c_k \rfloor)} V_{i+1,n}(v)\,du_i$
    If $i+l = n+1$ then
        $F_{i,n}(v) \leftarrow \int_{x_n-v}^{x_i+v} V_{i+1,n}(v)\,du_i$
    If $i+l < n+1$ then
        $F_{i,i+l-1}(v) \leftarrow \int_{x_{i+l-1}-v}^{x_i+v} \{V_{i+1,i+l-1}(v) + S(v)\}\,du_i$
    $S(v) \leftarrow S(v) + F_{i+1,\min\{i+l-1,n\}}(v)$
    For $j \leftarrow \min\{n-1, i+l-2\}$ to $\max\{i+1, z+2\}$ by $-1$ [Interior of $A$ matrix]
        $F_{i,j}(v) \leftarrow \int_{x_j-v}^{x_{j+1}-v} \{V_{i+1,j}(v) + S(v)\}\,du_i$
        $V_{i,j}(v) \leftarrow \int_{u_{i-1}}^{x_{j+1}-v} \{V_{i+1,j}(v) + S(v)\}\,du_i$
        $S(v) \leftarrow S(v) + F_{i+1,j}(v)$
    If $z+1 \leq i$ then
        $V_{i,i}(v) \leftarrow \int_{u_{i-1}}^{x_{i+1}-v} S(v)\,du_i$
    If $z+1 > i$ then
        $V_{i,z+1}(v) \leftarrow \int_{u_{i-1}}^{x_{z+2}-v} \{V_{i+1,z+1}(v) + S(v)\}\,du_i$
    If $z+1 < i$ then
        $F_{i,i}(v) \leftarrow \int_{x_i-v}^{x_{i+1}-v} S(v)\,du_i$
If $l = n$ then                    [Rows 2, 3, ..., $n$ of $A$ matrix completed at this point]
    $S(v) \leftarrow 0$
    $F_{1,n}(v) \leftarrow \int_{x_n-v}^{x_1+v} V_{2,n}(v)\,du_1$
else
    $S(v) \leftarrow F_{2,l+1}(v)$
If $l < n$ then
    $F_{1,l}(v) \leftarrow \int_{x_l-v}^{x_1+v} \{V_{2,l}(v) + S(v)\}\,du_1$
$S(v) \leftarrow S(v) + F_{2,l}(v)$
For $j \leftarrow \min\{n-1, l-1\}$ to $\max\{2, z+1\}$ by $-1$
    $F_{1,j}(v) \leftarrow \int_{(x_j-v)\lceil x_j-c_k \rceil}^{x_{j+1}-v} \{V_{2,j}(v) + S(v)\}\,du_1$
    $S(v) \leftarrow S(v) + F_{2,j}(v)$
If $z = 0$ then
    $F_{1,1}(v) \leftarrow \int_0^{x_2-v} S(v)\,du_1$
$P_k(v) \leftarrow 0$
For $j \leftarrow z+1$ to $l$
    $P_k(v) \leftarrow P_k(v) + F_{1,j}(v)$
$P_k(v) \leftarrow n!P_k(v)$
```

[**Phase 4:** Shift the distribution to the form of $P(D_n < y)$ by substituting $y = v + \frac{1}{2n}$.]

```
$y_0 \leftarrow v_0 + \frac{1}{2n}$
For $k \leftarrow 1$ to $m$                    [Loop through all subintervals of support]
    $y_i \leftarrow v_k + \frac{1}{2n}$
    $F_k(y) \leftarrow P_k(y - \frac{1}{2n})$
```

Return: $y_0, y_1, \ldots, y_m$ and $F_1(y), F_2(y), \ldots, F_m(y)$.

Appendix H

Simulation Code for MLEOS Approximations

As outlined in Chapter 7. the following Maple code was used to generate the MLEOS versus MLE estimations for θ from the Uniform(0. θ) distribution. The procedure `SecSolve` is a generic secant-method for finding numerical solutions to an equation. This procedure is faster in finding solutions to the MLEOS equation than Maple's `fsolve` command. The procedure **`MaxInitVals`** is a preparatory procedure for `SecSolve` that returns an interval where a local maximum can be found. It is merely a grid-search of an interval to find two points $x_1 < x_2$ such that $f'(x_1) > 0 > f'(x_2)$. which is a condition for a local maximum on $f(x)$ on $[x_1, x_2]$.

```
restart;
read(`d:/research/probmap/probproc.map`);
read(`d:/research/secant.map`);
read(`d:/research/maxini.map`);
```

```
ntest := 1000; nsample := 25; Digits := 10;
Seed := readlib(randomize)();
randomize(8218);
outpt := array(1 .. ntest);
thisdist := UniformRV(0, b);
bsample := 1;
for i from 1 to nsample do
  fnow[i] := OrderStat(thisdist, nsample, i):
od:
varsample := array(1 .. nsample);
wincnt := 0;
for k from 1 to ntest do
  print(kisnow, k);
  for i from 1 to nsample do
    varsample[i] := evalf(VarUni(0.0, bsample));
  od;
  varlist := sort([seq(varsample[i], i = 1 .. nsample)]);
  bold := varlist[nsample];
  lls := 0;
  for i from 1 to nsample do
    lls := lls + ln(fnow[i][1][1](varlist[i])):
  od:
  llsp := unapply((diff(lls, b)), b):
  bnew := SecSolve(llsp, 0.998, 0.996):
  if (bnew = bad) then
    ini := MaxInitVals(llsp, 0.02, 2.2, 5);
    print(`initial guesses are `, ini[1], ini[2]);
    bnew := SecSolve(llsp, ini[1], ini[2]);
  fi;
  if (bnew = bad) then
  bnew := varlist[nsample];
  elif (abs(bnew - 1) >  abs(1 - varlist[nsample])) then
    bnew := varlist[nsample];
  else
    wincnt := wincnt + 1; print(wincnt);
  fi;
  outpt[k] := [bnew, bold, bold * (nsample + 1) / nsample];
od:
sum1 := 0; sum2 := 0; sum3 := 0;
var1 := 0; var2 := 0; var3 := 0;
skipped := 0;
for i from 1 to ntest do
  if(nops(outpt[i]) = 3) then
```

```
      sum1 := sum1 + outpt[i][1];
      var1 := var1 + (outpt[i][1] - bsample ) ^ 2;
      sum2 := sum2 + outpt[i][2];
      var2 := var2 + (outpt[i][2] - bsample ) ^ 2;
      sum3 := sum3 + outpt[i][3];
      var3 := var3 + (outpt[i][3] - bsample ) ^ 2;
    else
      skipped := skipped + 1;
    fi;
  od:
  print('size of each sample = ', nsample,
      'number of samples = ', ntest);
  avenew := sum1 / (ntest - skipped);
  aveold := sum2 / (ntest - skipped);
  avgadj := sum3 / (ntest - skipped);
  varnew := var1 / (ntest - skipped);
  varold := var2 / (ntest - skipped);
  varadj := var3 / (ntest - skipped);
  print('had to skip', skipped, 'win percentage =',
      evalf(wincnt / (ntest - skipped)));
```

Bibliography

[1] Balakrishnan, N. and W. W. S. Chen, 1997. *CRC Handbook of Tables for Order Statistics from Inverse Gaussian Distributions with Applications.* New York: CRC.

[2] Barlow, R. E. and F. Proschan, 1981. *Statistical Theory of Reliability and Life Testing.* Silver Spring, Md.: To Begin With.

[3] Barr, D. R., 1994. Personal communication.

[4] Barr, D. R. and T. Davidson, 1973. "A Kolmogorov–Smirnov Test for Censored Samples." *Technometrics.* Volume 15, pp. 739–757.

[5] Barr, D. R. and P. W. Zehna, 1983. *Probability: Modeling Uncertainty.* Reading, Mass.: Addison–Wesley.

[6] Barr, D. R. and P. W. Zehna, 1971. *Probability.* Belmont, California: Brooks–Cole.

[7] Berger, R., 1995. Personal communication.

[8] Birnbaum, Z. W., 1952. "Numerical Tabulation of the Distribution of Kolmogorov's Statistic for Finite Sample Size." *Journal of the American Statistical Association.* Volume 47, pp. 425–441.

[9] Casella, G. and R. Berger, 1990. *Statistical Inference.* Pacific Grove, California: Wadsworth and Brooks/Cole, Inc.

[10] Chhikara, R. S. and Folks, L. S., 1989. *The Inverse Gaussian Distribution: Theory, Methodology and Applications.* New York: Marcel Dekker.

[11] Cox, D. R. and D. Oakes, 1984. *Analysis of Survival Data.* New York: Chapman and Hall.

[12] Cox. D. R. and E. J. Snell. 1981. *Applied Statistics.* New York: Chapman and Hall.

[13] Crowder. M. J., A. C. Kimber. R. L. Smith and T. J. Sweeting. 1991. *Statistical Analysis of Reliability Data.* New York: Chapman and Hall.

[14] D'Agostino. R. B. and M. A. Stephens. 1986. *Goodness-of-Fit Techniques.* New York: Marcel Dekker.

[15] David. H. A., 1981. *Order Statistics.* Second edition. New York: John Wiley and Sons.

[16] David. H. A., 1993. "A Note on Order Statistics for Dependent Variables." *Journal of the American Statistical Association.* Volume 47. pp. 198–199.

[17] Devroye. L., 1996. "Random Variate Generation in One Line of Code." from *Proceedings of the 1996 Winter Simulation Conference.* J. Charnes. D. Morrice, D. Brunner. and J. Swain. eds. Coronado, California: Institute of Electrical and Electronics Engineers. pp. 265–272.

[18] Evans. M., N. Hastings and B. Peacock, 1993. *Statistical Distributions.* Second edition. New York: John Wiley and Sons.

[19] Freund. J., 1992. *Mathematical Statistics.* Fifth edition. Englewood Cliffs. New Jersey: Prentice–Hall.

[20] Gehan. E. A., 1965. "A Generalized Wilcoxon Test for Comparing Arbitrarily Singly-Censored Samples." *Biometrika.* Volume 52. Parts 1 and 2. pp. 203–223.

[21] Harter. H. L. and N. Balakrishnan. 1996. *CRC Handbook of Tables for the Use of Order Statistics in Estimation.* New York: CRC.

[22] Hogg. R. V. and A. T. Craig. 1995. *Mathematical Statistics.* Fifth edition. Englewood Cliffs. New Jersey: Prentice–Hall.

[23] Johnson. N. L., S. Kotz, and N. Balakrishnan. 1995. *Continuous Univariate Distributions. Volume 2,* Second edition. New York: John Wiley and Sons.

[24] Johnson, R. A. and D. W. Wichern, 1992. *Applied Multivariate Statistical Analysis.* Third edition. Englewood Cliffs, New Jersey: Prentice–Hall.

[25] Kececioglu. D., 1991. *Reliability Engineering Handbook. Volume I.* Englewood Cliffs. New Jersey: Prentice–Hall.

[26] Kendall. M. G., and A. Stuart, 1963. *The Advanced Theory of Statistics: Volume II Inference and Relationship.* New York: Hafner Publishing.

[27] Knuth, D. E., 1981. *The Art of Computer Programming,* Second edition. Reading, Mass.: Addison–Wesley.

[28] Kuiper, N. H., 1960. "Tests Concerning Random Points on a Circle." *Proc. Koninkljke Nederlandse Akadamie van Wetenschappen, Series A*. Volume 63. pp. 38–47.

[29] Larsen, R. J. and M. L. Marx, 1986. *An Introduction to Mathematical Statistics and Its Applications*, Second edition. Englewood Cliffs, New Jersey: Prentice–Hall.

[30] Law, A. M. and W. D. Kelton, 1991. *Simulation Modeling and Analysis*. Second edition. New York: McGraw–Hill.

[31] Lee, E. T., 1992. *Statistical Methods for Survival Data Analysis*, Second edition. New York: John Wiley and Sons.

[32] Lee, S., J. R. Wilson, and M. M. Crawford, 1991. "Modeling and Simulation of a Nonhomogeneous Poisson Process with Cyclic Features." *Communications in Statistics - Simulation and Computation*. Volume 20, pp. 777–809.

[33] Leemis, L., 1995. *Probabilistic Models and Statistical Methods in Reliability*. Englewood Cliffs, New Jersey: Prentice–Hall.

[34] Lehmann, E. L., 1986. *Testing Statistical Hypothesis*, Second edition. New York: John Wiley and Sons.

[35] Lieblein, J. and M. Zelen, 1956. "Statistical Investigation of the Fatigue Life of Deep-Groove Ball Bearings." *Journal of Research of the National Bureau of Standards*. Volume 57, pp. 273–316.

[36] Lilliefors, H. W., 1967. "On the Kolmogorov–Smirnov Test for Normality with Mean and Variance Unknown." *Journal of the American Statistical Association*. Volume 62. pp. 399–402.

[37] Lilliefors, H. W., 1969. "On the Kolmogorov–Smirnov Test for the Exponential Distribution with Mean Unknown." *Journal of the American Statistical Association*. Volume 64. pp. 387–389.

[38] Miller, L. H., 1956. "Table of Percentage Points of Kolmogorov Statistics." *Journal of the American Statistical Association*. Volume 51. pp. 111–121.

[39] Moore, D. S. and G. P. McCabe, 1993. *Introduction to the Practice of Statistics*, Second edition. New York: W. H. Freeman and Company.

[40] Owen, D. B., 1962. *Handbook of Statistical Tables*. Reading, Mass.: Addison–Wesley.

[41] Park, S. and L. M. Leemis, 1997. *Discrete-event Simulation: A First Course*. The College of William & Mary.

[42] Port, S. C., 1994. *Theoretical Probability for Applications*. New York: John Wiley and Sons.

[43] Rohatgi, V. K., 1976. *An Introduction to Probability Theory and Mathematical Statistics*. New York: John Wiley and Sons.

[44] Ross, S. M., 1993. *Introduction to Probability Models*. Fifth edition. Boston: Academy Press.

[45] Sahner, R. A. and K. S. Trivedi, 1993. "A Software Tool for Learning About Stochastic Models." *IEEE Transactions on Education*. Volume 36. pp. 56–61.

[46] Sarhan A. E. and B. G. Greenberg, 1962. *Contributions to Order Statistics*. New York: John Wiley and Sons.

[47] Stephens, M. A., 1965. "The Goodness-of-fit Statistics: Distributions and Significance Points." *Biometrika*. Volume 52. pp. 309–322.

[48] Stephens, M. A., 1986. "Tests Based on EDF Statistics." from *Goodness-of-fit Test Techniques*. R. B. D'Agostino and M. A. Stephens, eds. New York: Marcel Dekker. pp. 97–193.

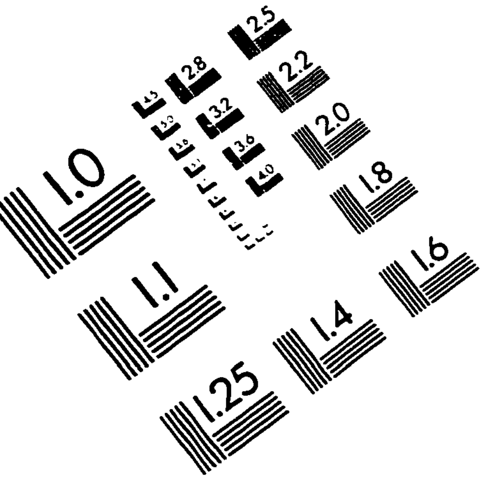
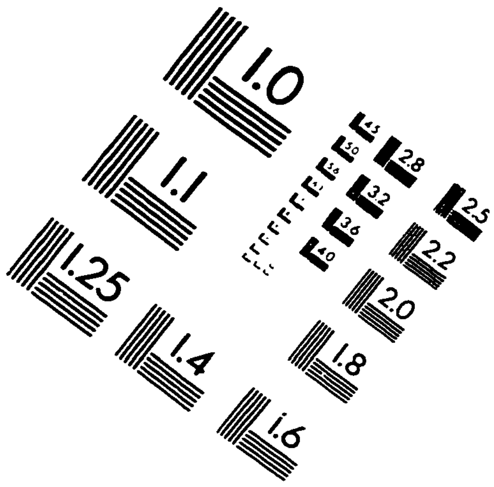
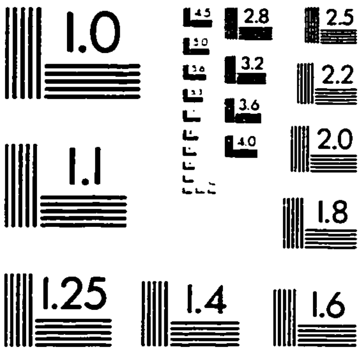
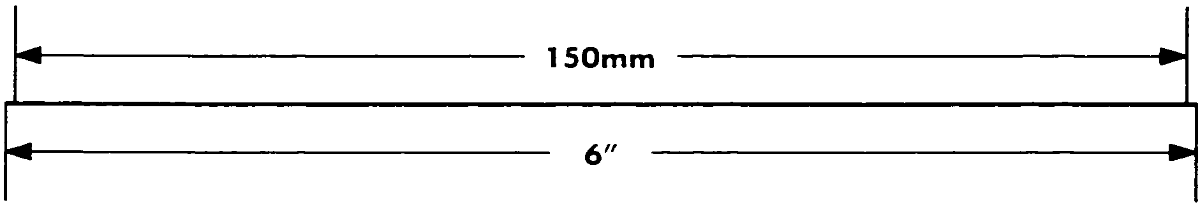
[49] Tietjen, G. L., 1986. "The Analysis and Detection of Outliers." from *Goodness-of-fit Techniques*. R. B. D'Agostino and M. A. Stephens, eds. New York: Marcel Dekker. pp. 497–522.

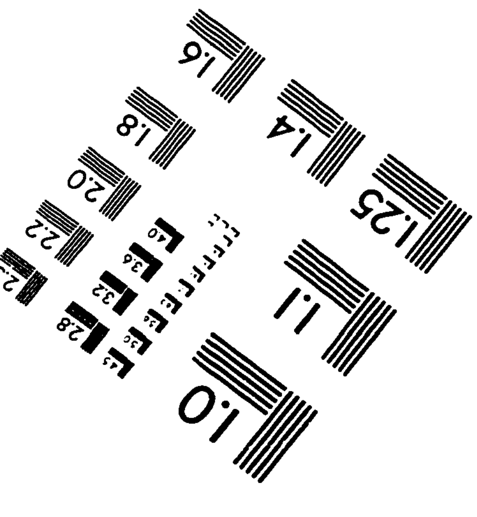
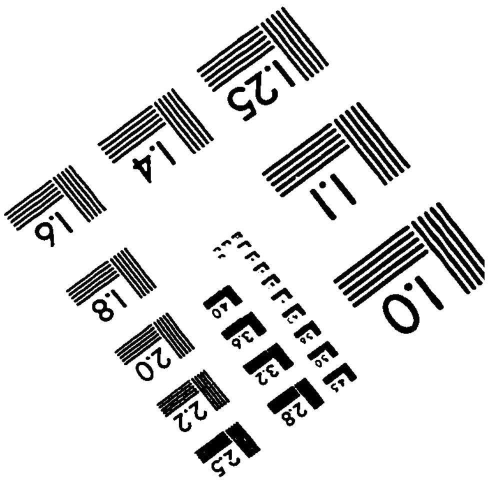
[50] Trivedi, K. S., 1997. Personal communication.

Vita

Andrew Gordon Glen is a Major in the United States Army, currently assigned as an assistant professor in the Department of Systems Engineering at the United States Military Academy in West Point, New York. He was born on February 22, 1962 in Denville, New Jersey. He holds a Bachelors of Science degree, graduated and commissioned in the Field Artillery by the United States Military Academy in 1984. He also holds a Masters of Science degree from the College of William & Mary, graduated in 1995. He is a career Army officer of 14 years who has served numerous overseas tours, to include combat duty in the Persian Gulf. His military awards include the Bronze Star and the Meritorious Service Medal with Oak Leaf Cluster. He is married to the former Lisa Mary Stewart, now also a Major in the Army. The Glens have three daughters, Andrea, Rebecca, and Mary.

IMAGE EVALUATION
TEST TARGET (QA-3)

APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989