Chapter 1

One Dimension

1.1 One Dimension

This section covers the computing of functions of a single random variable. The following serve as motivating examples,

1. X^2
2. $log(X)$
3. $\frac{1}{X}$
4. $\frac{1}{log(X^2-X)}$
5. $[X-k]^+ - [X-k]^- \equiv X-k$

An interesting feature of the last item, known as Put-Call Parity [8], is that it seems to recover lost information through truncation. If $Z_+ := [X-k]^+$ and $Z_- := [X-k]^-$ then Z_+ and Z_- are maximally correlated. In the language of quantum mechanics the two Z's are *maximally entangled*.

`find reference` TO DO

As a naming convenience it is assumed that X represents a *basic random variable*. A basic random variable is on that is not related to any other random variable within RICO by an explicit function. Conversely the label Z refers to a random variable that is functionally related to some X via an explicit function f such that $Z = f(X)$. When more than one function of X is needed they will be distinguished by subscripting. Since a random variable is defined by its associated cumulative density function RICO requires random variables to be associated with the following computable expressions

$$P(Z < k) \qquad\qquad P(Z = k) \tag{1.1}$$

for any constant k. In order to incorporate functions of random variables into an algorithmic context is it also necessary to be able to compute

$$P(Z_1 < Z_2) \qquad \text{where} \qquad Z_1 = f_1(X),\ Z_2 = f_2(X)$$

Notice that $P(Z_1 < Z_2) = P(Z_1 - Z_2 < 0)$ and that $Z_1 - Z_2$ is itself a random variable. The computable expressions in (1.1) are sufficient to compute $P(Z_1 < Z_2)$ which may arise in conditional branching statements such as $if(Z_1 < Z_2)\{...\}$. Both branches of an *if* statement may be followed albeit with different associated probabilities. Conditional branching statements will be discussed in a later section.

1.1.1 Plotting $Z = f(X)$

A *continuous* random variable has the property that

$$P(X = k) = 0 \ \forall\ k \in \mathcal{D}(X) \tag{1.2}$$

and it assumed by RICO that a continuous random variable has an associated probability density function, $\rho(X)$. The expression $\mathcal{D}(X)$ refers to the support of X which in the continuous case is the domain of the associated probability density function.

A *discrete* random variable has discrete support such that

$$P(X = k_i) = p_i \ \forall\ k_i \in K = \{k_1, k_2, \ldots, k_n\} \tag{1.3}$$

In general a random variable X may be a mixture of continuous and discretely ditributed probability. In RICO the continuous and discrete aspects of a random variable are represented separately. The continuous aspect of a random variable is more numerically challenging and will be the focus of this section as a special case.

Notice that the discrete aspect of a random variable cannot be ignored even if an original X is purely continuous. Consider the motivating example, $Z = [X - k]^+$ in (5), transforms a continuous X into a *mixed* continuous/discrete Z.

In the next section it is assumed that both X and $Z = f(X)$ are continuous random variables.

Plotting Continuous $Z = f(X)$

To plot the probability density function of $Z = f(X)$ it is assumed that a software module will be employed to render the actual graph for the user and that the module requires an array of pairs of points,

$$\{(z_i, h_i)\}_{i=1}^{n} \tag{1.4}$$

where z_i is a point in the support of Z and $h_i = \rho(z_i)$, the probability density of Z at z_i. The associated probability density function of Z is denoted $\rho(Z)$.

To first approximation the values of each h_i are found by choosing a set of partition endpoints

$$(z_i^p)_{i=1}^{n} \subset \mathcal{D}(Z)$$

so that

$$z_i \in (z_i^p, z_{i+1}^p)$$

and the h_i are approximated as

$$h_i = \frac{p_i}{z_{i+1}^p - z_i^p} \qquad \text{where } p_i = P(z_i^p < Z < z_{i+1}^p) \qquad i = 1, \ldots, n$$

A *partition of* Z is understood to be a partition of the range of $f(X)$.

The relationship between the partition elements is shown in figure 1.1. Assuming $n \geq 2$ the z_i^p partition points are computed in RICO as midpoints of the $(z_i)_{i=1}^{n}$ array

$$z_i^p = \begin{cases} z_1 - \frac{1}{2}(z_1 + z_2) & \text{if } i = 1 \\ \frac{1}{2}(z_{i-1} + z_i) & \text{if } 1 < i \leq n \\ z_n + \frac{1}{2}(z_{n-1} + z_n) & \text{if } i = n+1 \end{cases}$$

Let the associated partition of Z be

$$Z^p = (-\infty, z_1^p, z_2^p, ..., z_{n+1}^p, \infty)$$

Since Z is a purely continuous random variable the *partition elements* of Z^p are assumed in RICO to be open intervals. The missing endpoints of the partition elements form a set of probability zero. Let the partition elements of Z^p and associated probability values be indexed as

$$\begin{aligned} Z_0^p &= (-\infty, z_1^p) & p_0 &= P(Z_0^p) & & \\ Z_i^p &= (z_i^p, z_{i+1}^p) & p_i &= P(Z_i^p) & & i = 1, \ldots, n \\ Z_{n+1}^p &= (z_{n+1}^p, \infty) & p_{n+1} &= P(Z_{n+1}^p) & & \end{aligned}$$

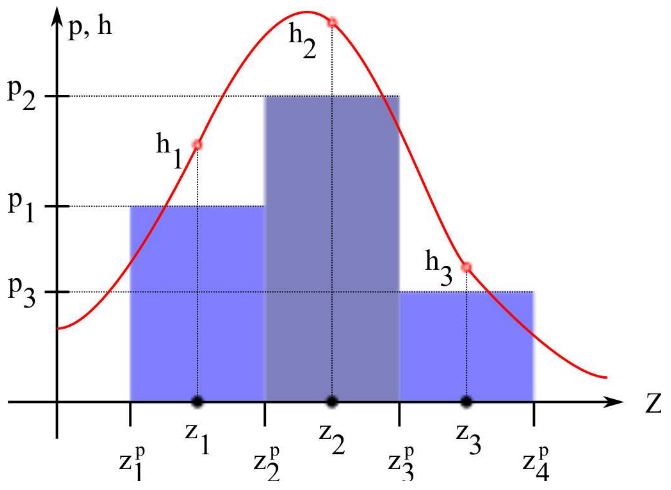

Figure 1.1: Three point Partition of Z

Numerical Considerations

Numerically, real numbers are represented by a finite number of floating point values. Let Ω be the largest representable floating point value. Similarly let ι be the smallest positive floating point value. In RICO, a constant called ω is defined as

$$\omega = min\left\{\Omega, \frac{1}{\iota}\right\} \tag{1.5}$$

and let ϵ be the smallest numerically representable positive probability value. One purpose for defining ϵ is so that so-called *thin tailed* probability distribution such as the Gaussian may be represented using a finite support interval as in

$$\mathcal{D}(X) = (X_{min}, X_{max})$$

where $-X_{min} = X_{max} \approx 10$, depending on ϵ. The X_{min} and X_{max} satisfy the following relations

$$P(X < X_{min}) \leq \epsilon \qquad\qquad P(X_{max} < X) \leq \epsilon$$

In particular all functions of a random variable have domain and range in $(-\omega, \omega)^2$ as represented visually by the shaded region in figure 1.2.

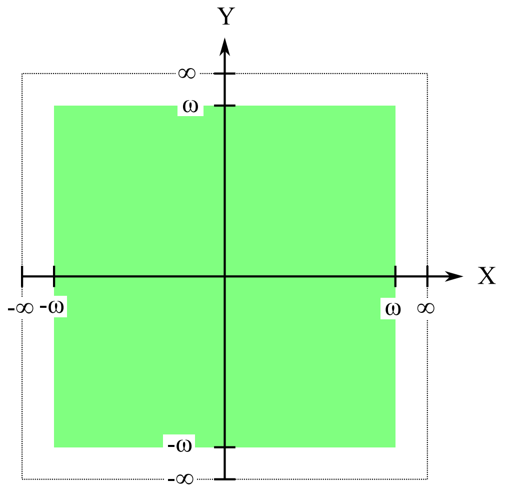

Figure 1.2: Fundamental Numeric Domain

Functions of Random Variables

In RICO there are several types of functions; *primitive*, *composite* and *piecewise monotonic*. Both primitive and composite functions are described elsewhere. An relevant feature of a primitive function such as $x^2, log(x)$, etc. is that it has a *primitive domain*. A primitive domain is a traditional mathematic domain. For example the primitive domain of x^2 is the interval $(-\infty, \infty)$ and for $log(x)$ the primitive domain is $(0, \infty)$. And example of a composite function is $x^2 - x$, a function composed of other composite functions, primitive functions and RICO-supported operations such as addition, subtraction, multiplication, etc.

In RICO, a piecewise monotonic function $f(X)$ of a random variable X is set of *piecewise monotonic function elements*. A piecewise monotonic function element is a domain interval and either another piecewise monotonic function element, a composite function or a primitive function. The set of piecewise monotonic function domain intervals form a non-intersecting set denoted $\mathcal{D}(Z)$ where $Z = f(X)$. The implied partition of the domain space is a set of open intervals containins $\mathcal{D}(Z)$ and denoted Z^p. The Z^p set is described above in the context plotting $f(X)$ and is defined here via

$$\mathcal{D}(Z)_j = z_i^p \in Z^p \qquad \text{for some } i \in 1, \cdots, n \text{ and } j \in 1, \cdots, m$$

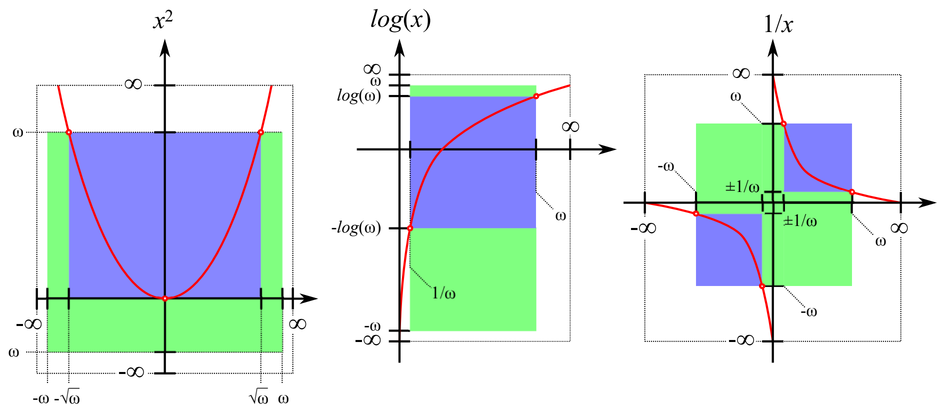

Figure 1.3: Refined Domains of Primitive Functions

For example, if $f(X) = [X - k]^+$ then

$$\begin{array}{ll} \mathcal{D}(Z)_1 = (-\infty, k), & \mathcal{D}(Z)_2 = (k, \infty) \\ f_1(x) = 0, & f_2(x) = x \end{array}$$

where f_1 and f_2 are primitive functions.

Notice that the recursive nature of the definition of a piecewise monotonic function forms a tree structure denoted $\mathcal{T}(f)$. The leaf nodes of $\mathcal{T}(f)$ are piecewise monotonic elements whose associated functions are either composite of primitive. A critical aspect of the definition of a piecewise monotonic function in RICO is that leaf node functions are monotonic over their associated domain interval. For example the primitive function $f(x) = x^2$ has the associated primitive domain $(-\infty, \infty)$, but the piecewise monotonic function $f(x) = x^2$ is composed of two elements,

$$f(x) = \begin{cases} x^2 \text{ if } x \in (-\sqrt{\omega}, 0) \\ x^2 \text{ if } x \in (0, \sqrt{\omega}) \end{cases} \tag{1.6}$$

where the numerical considerations described in the previous section are taken into account.

Piecewise Monotonic Functions of Random Variables

For the purpose of plotting an other analysis such as computing $P(Z < k)$ for $Z = f(X)$ a composite f must be transformed into a piecewise monotonic f. The

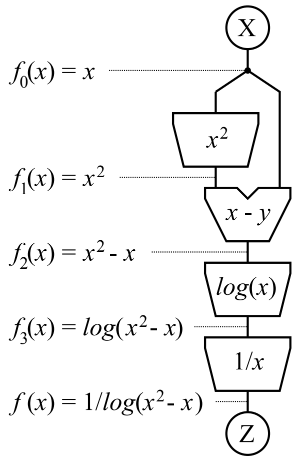

Figure 1.4: Parse Tree of $Z = 1/log(X^2 - X)$

transformation process is referred to as *refinment*. Function refinment serves two purposes. The first is to ensure that the numerical considerations are respected such that the associated function evaluates to a representable floating point value for every element of the domain. The second purpose is to ensure monotonicity within each domain interval.

The refinment for several primitive functions is depicted in figure 1.3 As a guiding example of the refinment process suppose that

$$Z = f(X) = \frac{1}{log(X^2 - X)}$$

where

$$X \sim Normal(0, 1) \text{ and } \mathcal{D}(X) = (-10, 10)$$

The parse tree of f is a composition of primitive functions, composite function and arithmetic operator components is shown in figure 1.4. Transforming the composite f into the piecewise monotonic f is a sequention process that begins at the input X. Several composite function components are identified in figure 1.4 including the indentity function $f_0(x) = x$.

To begin, note that the domain of X is the interval $(-10, 10)$ which is assigned to be the domain of f_0. The next step is to consider the effect of f_0 of the adjacent node x^2 and the y-input of the operation $x - y$. Assume that $f_1(x)$ is refined in isolation of the containing $f(x)$ so that its initial definition is given in equation

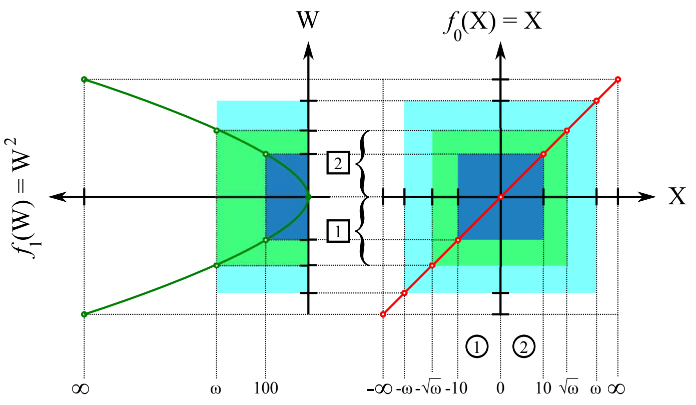

Figure 1.5: Refinement $f_0(X)$ by $f_1(X)$

1.6. The refinment process is to take the range of f_0, $(-10, 10)$, and intersect it with the domain of f_1 into range components $\{(-10, 0), (0, 10)\}$ and find the pre-image under f_0 which is also $\{(-10, 0), (0, 10)\}$. This process is shown in figure 1.5 where $\boxed{1}$ = $(-\sqrt{\omega}, 0)$, $\boxed{2}$ = $(0, \sqrt{\omega})$, ① = $(-10, 0)$, ② = $(0, 10)$.

In isolation the operation $x - y$ accepts any input in $(-\omega, \omega)$ so no further refinement is needed at this stage. Similarly no further refinement of f_2 is imposed by $x - y$. The domain of f_2 is then the set {①, ②} in figure 1.5.

The domain of f_2 in the parse tree 1.4 is refined by the domain of the primitive $log(x)$. Because f_2 involves a 2×1-dimensional transform, namely subtraction, and since the two operands are maximally entangled, the domain is less clearly defined and must be identified numerically. In general any value can be achieved by the transform $x - y$. Referring to the figure 1.6 the domain of $log(x)$ is $\boxed{1}$ = $(1/\omega, \omega)$. The domain of $f_2(X)$ is notated as before,

$$\begin{aligned}\mathcal{D}(f_2) &= \mathcal{D}(f_1) \cap f_2^{-1}(\mathcal{D}(log)) \\ &= (-10, 10) \cap \big((-\sqrt{\omega}, x_1) \cup (x_4, \sqrt{\omega})\big) \\ &= (-10, x_1) \cup (x_2, 10)\end{aligned}$$

where

$$x_1 = min\left\{-f_2^{-1}(1/\omega), -\frac{1}{\omega}\right\} \qquad x_2 = max\left\{+f_2^{-1}(1/\omega), 1 + \frac{1}{\omega}\right\}$$

Numerical considerations dictate that since $f_2^{-1}(1/\omega) < 1/\omega$ the nearest x-value within each domain interval must be found. At this stage the function f_2 is refined

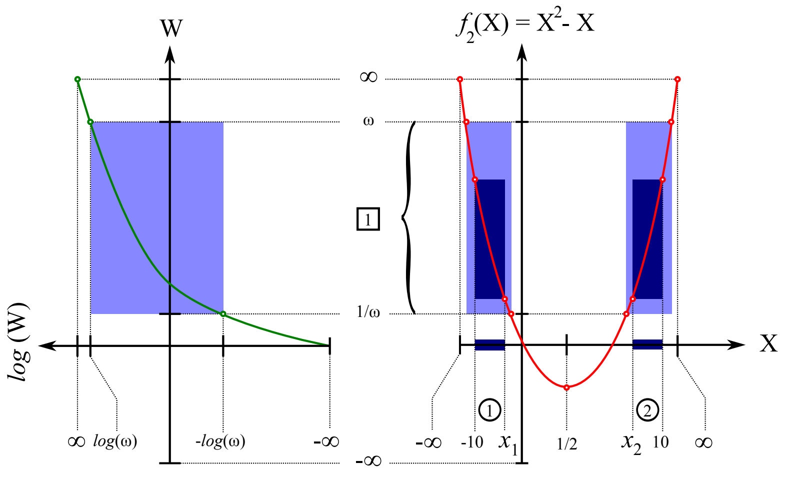

Figure 1.6: Domain of $f_2(X)$

by $log(x)$ and piecewise defined as

$$f_2(x) = \begin{cases} x^2 - x \text{ if } x \in (-10, -\frac{1}{\omega}) \\ x^2 - x \text{ if } x \in (1 + \frac{1}{\omega}, 10) \end{cases}$$

The last step of domain refinement in this example is shown in figure 1.7. Recall from 1.3 that the numerical domain of $1/x$ requires a small exclusion interval $C = (-1/\omega, 1/\omega)$. In isolation the primitive reciprocal function is defined on the piecewise domain $\{(-\omega, -1/\omega), (1/\omega, \omega)\}$. In figure 1.7 the domains are $\boxed{1}$ and $\boxed{2}$ respectively. The pre-images under f_3 are then intersected with the domain of refined f_2 yielding the refined domain of f_3 as

$$\begin{aligned} \mathcal{D}(f_3) &= \mathcal{D}(f_1(X)) \cap f_3^{-1}(\{(-\omega, -1/\omega), (1/\omega, \omega)\}) \\ &= (-10, x_3) \cup (x_4, x_1) \cup (x_2, x_5) \cup (x_6, 10) \end{aligned}$$

In figure 1.7 the refined domain of f_3 is shown as ①, . . . , ④. Notice that the intervals (x_3, x_4) and (x_5, x_6) were excluded from the the refined domain of f_2 and that they are very small and thus likely of negligible probability.

Notice that since f does not feed into any other function elements in the example parse tree 1.4 requires no further refinement. The final function of $f(X)$ is shown in figure 1.8 with identified domain { ①, ②, ③, ④ }.

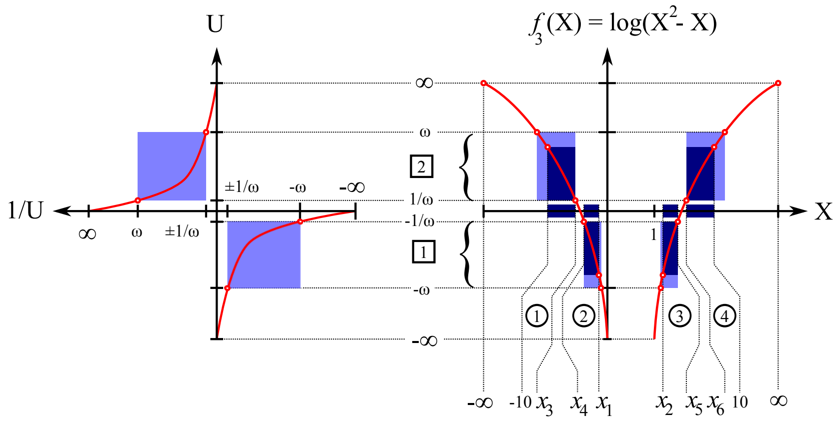

Figure 1.7: Domain of $f_3(X)$

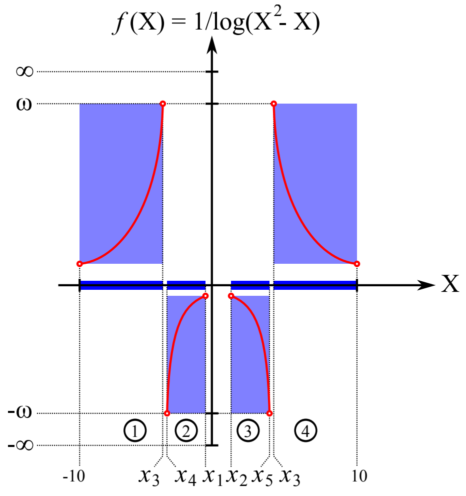

Figure 1.8: $Z = f(X)$ with Domain

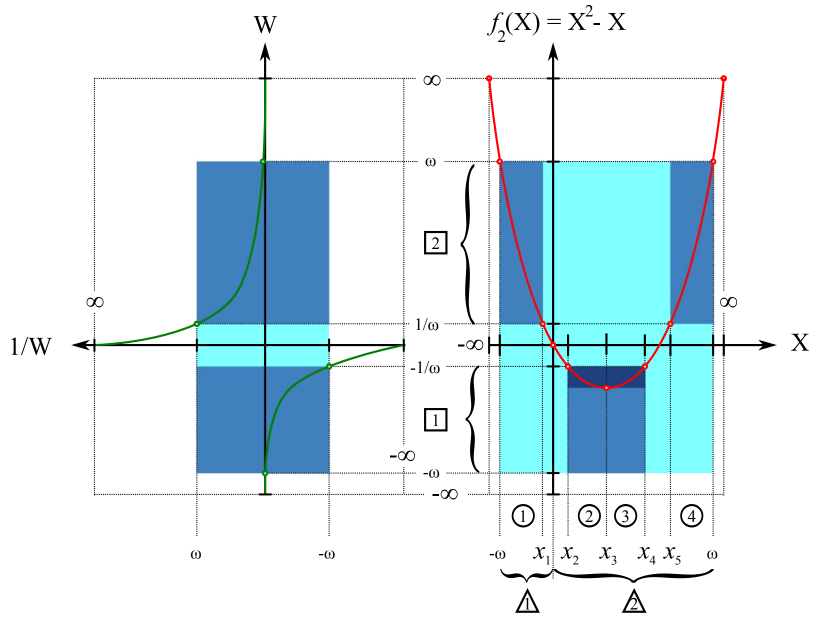

Figure 1.9: Refining $\frac{1}{x^2-x}$

Combining Monotonic Functions of Random Variables

The previous example revealed some important issues in computing functions of random variables. Breaking a function into non-overlapping monotonic fragments begs the question of how to locate interval endpoints. Domain refinement by intersection brought some numerical considerations when values became too small to represent as floating point values. The composition of two bijective functions is a bijective function, but the the same cannot be said for the sum or project of two bijections. The issue of refining a function that is the sum or product of bijections is addressed in this sub-section.

Consider the example,

$$f(x) = \frac{1}{x^2 - x}$$

Since $f(x)$ contains x^2 the initial domain set is $\{(-\omega, 0), (0, \omega)\}$ represented in figure 1.9 by $\triangle\!\!\!\!\!\;1$ and $\triangle\!\!\!\!\!\;2$ respectively. Recalling the fundamental domain set of $1/x$ from figure 1.3 as $\{(-\omega, -1/\omega), (1/\omega, \omega)\}$ as $\boxed{1}$ and $\boxed{2}$ respectively.

Continuing with figure 1.9, the refined domain element $\textcircled{1}$ is found by intersecting the pre-image of $\boxed{2}$ under f given domain element $\triangle 1$. Finding the refined domain element $\textcircled{4}$ seems just a straightforward; the pre-image under f given domain $\triangle 2$ intersected with $\triangle 2$, but only because f happened to be monotonic in $\textcircled{4}$. Notice that f over domain element $\triangle 2$ is not monotonic. What is required is the intermediate step of refining each original domain element ($\triangle 1$ and $\triangle 2$ in this example) so that f is monotonic over each domain element before further refining f against $1/x$.

Chapter 2

Introduction

2.1 Introduction

The purpose of the work presented herein is to create a means of introducing uncertainty into deterministic models with performance superior to stand-alone Monte Carlo-based sampling techniques.

In the course of developing this material several spreadsheet-based models are analyzed. (See Appendix). A situation believed to be common is that spreadsheet-based models tend to be resistent to structural changes and their sensitivity to inputs is difficult to measure without external tools.

A remedy of the spreadsheet-based modeling issue is to reverse-engineer the spreadsheet to discover the specific model and re-implement this model in a traditional programming environment. This re-implementation process itself only addresses the issue of model brittleness and not the sensitivity issue. The work discussed in this paper recommends implementing a class of models such as those that may be implemented in a spreadsheet environment into a special environment. The environment proposed is called *RICO*, an achronym for *Random input, Correlated output*.

A RICO programming environment allows the programmatic manipulation of random variables as first class computing objects. The practical upshot is that if a given model accepts numeric data, that same model implemented in a RICO environment can substitute any input value with a random variable thereby allowing the study of model input sensisitivity and model response to input uncertainty. When multiple model inputs are replaced with random variables versions of these variables may become correlated within the model even if independent initially.

Numeric model outputs may then be a joint distribution of correlated random variables.

A RICO programming environment is not a specific software product, but a specification for constructing a software programming environment. While developing RICO a collection of software modules are constructed and used to produce many of the numeric and graphical results discussed in this paper. Collectively these software modules consitute a *reference* implementation of RICO. Any code snippets shown all run in at least one of the software modules within the RICO reference implementation.

Not limited to models implementable within a spreadsheet-based environment, RICO defines a number of basic mathematical operations and program flow control statements that are common to a wide variety of models. These operations include addition, subtraction, multiplication, division, exponentiation (both x^y and $exp(x)$) and logarithm. Program flow control statements include conditional statements such as $IF(X \leq Y)$ where either or both X and Y are optionally random variables. Interestingly, if at least one of X and Y is a random variable the conditional IF statement will take *both* not exclusively one or the other of the two implied code paths.

The class of models implementable in a RICO environment as large and several examples are explored. Some examples detailed below energy policy analysis, business cost savings analysis and geometric Black Scholes pricing. Important modeling components include constrained optimization and linear algebra involving random variables.

```
Find reference (Dineen?) likening a stochastic
   process to a sequence of random variables
```

TO DO

2.1.1 A Basic Model

At its heart a RICO model is a function of some number of inputs that produces some number of output symbolically represented in figure 2.1.

To introduce uncertainty into the model a random variable is identified for one or more inputs. As an example suppose a log-normal looking random variable X such as shown in figure 2.2.

Without loss of generality suppose a basic model with a number of input variables two of which (X and Y) are random and one output value of interest, Z. The Monte Carlo method of analysis requires a random sample for each

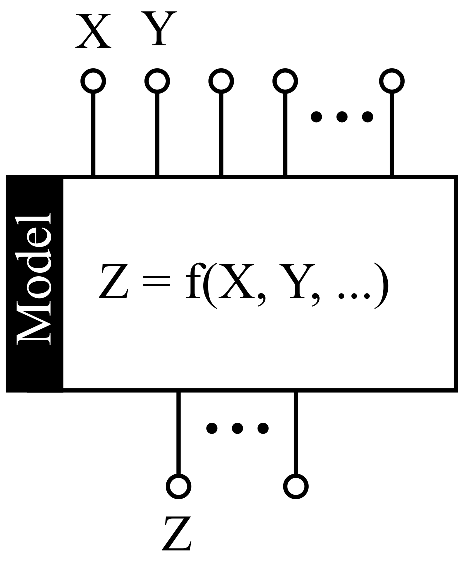

Figure 2.1: A Basic Model

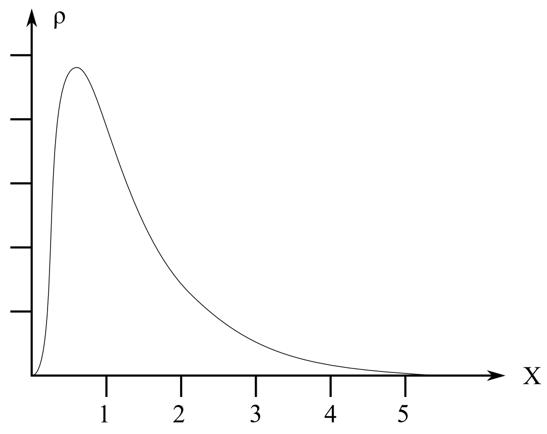

Figure 2.2: An Input Random Variable

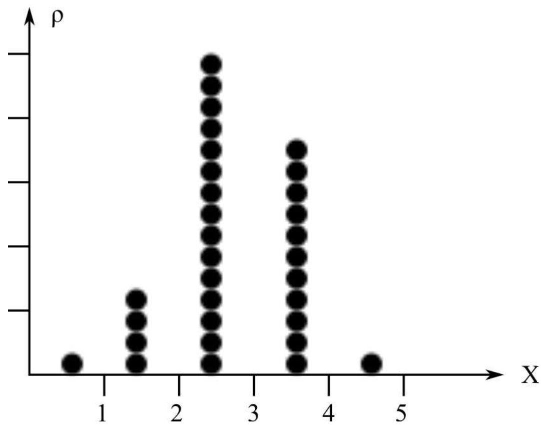

Figure 2.3: Example Output Histogram

random variable is generated. Suppose the random sample for X is denoted $x = (x_1, x_2, ..., x_n)$ and similarly for Y, $y = (y_1, y_2, ..., y_n)$. The model is run n times using the input values x_i and y_i in place of the X and Y inputs respectively for $i \in 1, ..., n$ generating n output values for Z called $z = (z_1, ..., z_n)$.

To understand the output values $z = (z_1, ..., z_n)$ generated by an n-run Monte Carlo method a partition of the space of output values is created by some means and the individual z_i values are counted into *bins*, that is, partition intervals. A possible result with unit-interval bins is shown in figure 2.3

A common exercise in statistics

`reference` TO DO

is to postulate a familiy of probability distributions and fit the 'best' one to the output sample z. An example result is shown in figure 2.4

A serious concern is that in the context of an algorithmic model it is challenging to find a family of probability distributions from which to select a 'best' fit for the observed output sample, z. If the output is unimodal a subset of the extensive exponential family may be chosen, but for multi-modal output the choice is less clear.

The alternative offered by RICO is a hybrid approach. The RICO approach is to maintain a symbolic description of input and internal model variables where possible. When a symbolic description is not possible for an internal random variable, a numberic description is used. There is a practical limit on the number

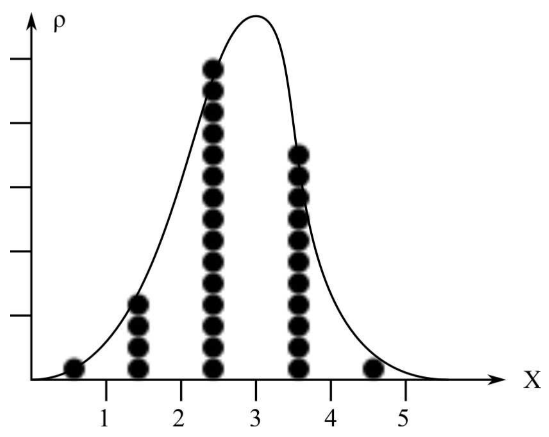

Figure 2.4: Example Output Histogram - Fitted

of dimensions used to describe a numeric random variable and when this limit is surpassed the final techniques applied is traditional Monte Carlo sampling.

A significant advantage that the RICO approach offers is symbolic representation of random variable tails. It is taken as axiomatic that the tail of a random variable is that region that cannot be reliably sampled. For random variables with so-called *thin-tailed* probability distributions, failure to adequately sample from the tails may be inconsequential, but *fat-tailed* distributions such as the Cauchy distribution have a significant amount of probability mass in the tails and this region of the support space cannot be safely ignored. Consider, for example, that the Cauchy distribution, like the St. Petersburg lottery, has no finite expected value.

`include at least one reference such as Tanner [17]` TO DO

Chapter 3

Numeric Computation

3.1 Numeric Representation

Given a sample of a random variable and a partition of the support space as in figure 3.1 a histogram forms a natural summary. In RICO, a method of representing a one dimensional continuous random variable numerically is suggested in figure 3.2. Since random variables may be both discrete and continuously distributed at the same time a pair of parallel arrays is used programmatically,

$$X \sim \begin{cases} X_c = \begin{cases} (x_0 = -\infty, x_1, x_2, ..., x_n, x_{n+1} = \infty) \\ (p_0, p_1, ..., p_n, 0) \end{cases} \\ X_d = \begin{cases} (y_1, y_2, ..., x_m) \\ (q_1, q_2, ..., q_m) \end{cases} \end{cases}$$

where

$$-\infty < x_1 < ... < x_n < \infty$$

The endpoints for the continuously distributed portion of X, X_c, are assumed to be $\pm\infty$ with n partition endpoints between. The p_i values are defined as

$$p_i := \begin{cases} P(x_i < X_c < x_{i+1}) & \text{if } i \in \{1, ..., n-1\} \\ P(X_c < x_1) & \text{if } i = 0 \\ P(x_n < X_c) & \text{if } i = n+1 \end{cases}$$

and the q_j values are defined as

$$q_j := P(X_d = y_j) \text{ for } j \in \{1, ..., m\}$$

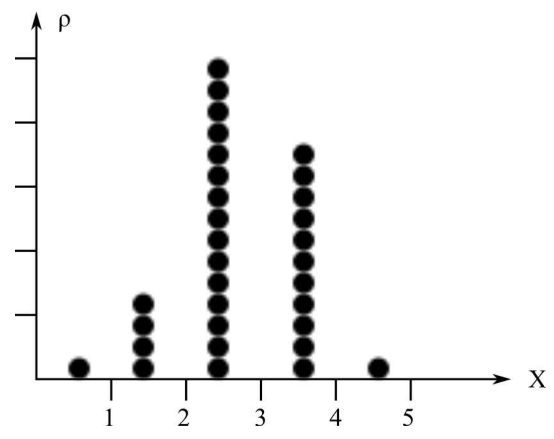

Figure 3.1: Example Sample

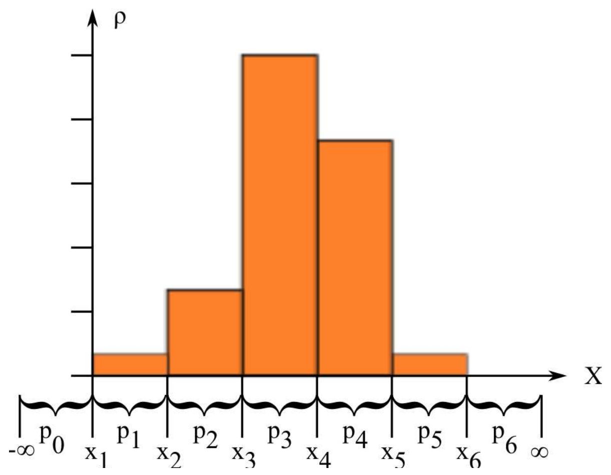

Figure 3.2: 1D Numeric Random Variable

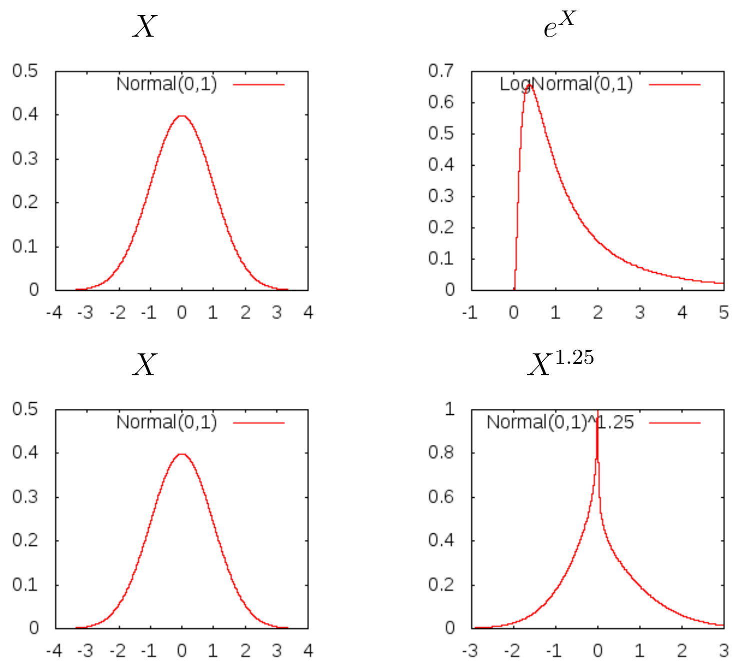

Table 3.1: Standard Normal X, e^X, $X^{1.25}$

For an example of RICO in action, let $X \sim N(0,1)$ numerically represented. New random variables such as e^X and $X^{1.25}$ can be created and graphed, see table 3.1. The code used to generate the graphs is

```
X = NormalNumeric(0,1,1000)
Plot(exp(X))
Plot(X**1.25)
```

Illustrative example of multiplication of two piecewise continuous random variables X and Y. Referring to figure 3.3, the following conditions hold for X and Y,

$$P(1 < X < 2) = 0.2 \qquad P(2 < X < 3) = 0.8$$
$$P(4 < Y < 5) = 0.3 \qquad P(5 < Y < 6) = 0.7$$

and the probability densities are super-imposed on the joint density distribution of (X, Y). For illustrative purposes a partition of the XY random variables is

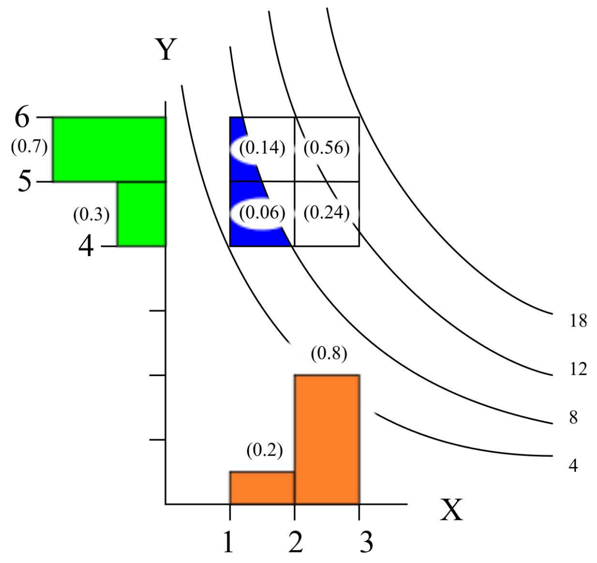

Figure 3.3: Piecewise Continuous X times Y

chosen as $(4, 8, 12, 18)$ to coincide with the corners of the non-zero probability density region of (X, Y). Each partition endpoint $\{4, 8, 12, 18\}$ corresponds to an iso-probability level curve identified in the figure 3.3. Notice that in the case of multiplication the level curves are hyperbolas. To compute the numeric random variable XY given the partition $(4, 8, 12, 18)$, calculate the area between level curves within joint probability rectangles. In particular, to compute $P(4 < XY < 8)$ one must find the fractional area of each of the two shaded rectangles multiplied by the probability contained in each such rectangle. The probability of the two shaded rectangles is $0.14 + 0.06 = 0.2$. To compute the probability of the shaded area using RICO involves forcing the particular partition shown in the figure as follows,

```
X = ContinuousNumeric((1,2,3),(0.2,0.8,0))
Y = ContinuousNumeric((4,5,6),(0.3,0.7,0))
Y.getNumericRandomVariable().force_partition((4,8,12,18))
X*Y
#()
#(4.0(0.11130904824004995)8.0(0.3565452412002493)
   12.0(0.5321457105597007)18.0,)
```

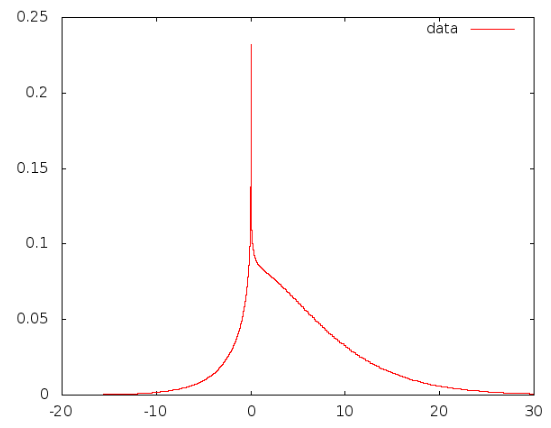

Figure 3.4: N(5,3) x N(1,1)

According to RICO the results of the above numerical example are

$$P(4 < XY < 8) \approx 11\% \quad P(8 < XY < 12) \approx 36\% \quad P(12 < XY < 18) \approx 53\%$$

A larger example of multiplication is XY where $X \sim N(5,3), Y \sim N(1,1)$. The listing follows as the plot is shown in figure 3.4.

```
X = Normal(5,3)
Y = Normal(1,1)
XY = X*Y
Plot().xrange(-20,30).plot(XY).show()
```

Similarly, division of two numeric random variables, this time in one line of code with the result shown in figure 3.5 which happens to be multi-modal.

```
Plot().xrange(-20,30).plot(Normal(7,3)/Normal(1,1)).show()
```

3.2 Correlation

Consider the example in figure 3.2 without preperatory remarks.

The issue in figure 3.2 is that RICO needs to understand that two references to the same underlying variable are 100% correlated. When RICO is asked to

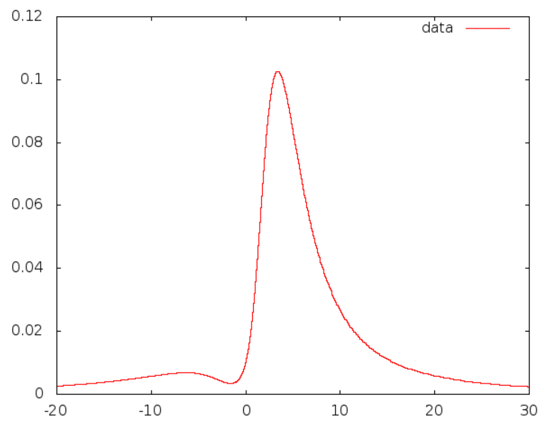

Figure 3.5: N(7,3) / N(1,1)

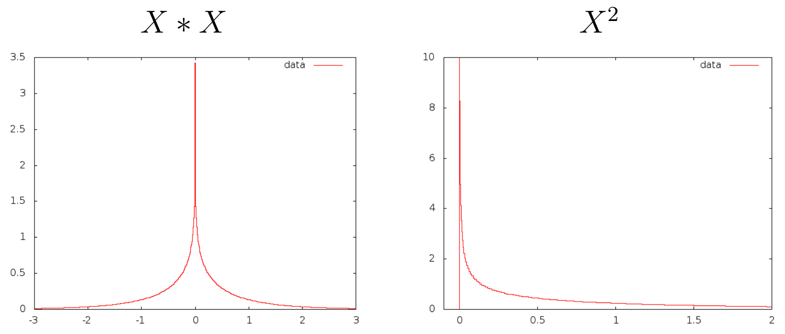

Table 3.2: Standard Normal $X * X$ versus X^2

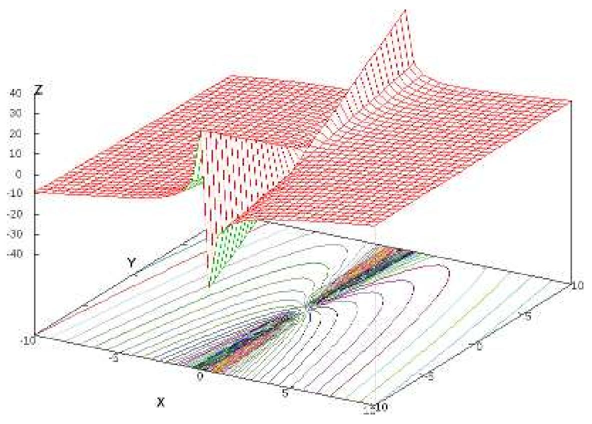

Figure 3.6: X + Y/X

compute X^2 there is no confusion, but $X * X$, without correlation tracking appears as the product of two independent copies of X.

Within the context of an algorithmic model with some random variable inputs intermediate variables are created and mixed with other intermediate variables such that partially correlated expressions are possible,

$$X + XY$$
$$X + Y/X$$

Assuming that $X \perp Y$ the multiplication XY is between independent random variables and is computable is detailed in the previous section. The sum in $X + XY$ is not between independent random variables. Fortunately RICO is equipped with a symbolic processing engine so that the expression $X + XY$ will be factored into $X(1 + Y)$. This expression is computable as a sequence of operations. The expression $1 + Y$ is independent of X so the product $X(1 + Y)$ is between independent random variables.

The expression $X + Y/X$ is not decomposible into a sequence of operations between independent random variables. In such a case the iso-probability level curves are more complicated than the hyperbolas found in the multiplication case. The level curves for a particular partition of $Z = X + Y/X$ are shown in figure 3.6.

In the case of $Z = X + Y/X$, the partitions of X and Y for a partition of the joint (X, Y) space. A subset of the iso-probability level curves associated with the partition of Z are approximated within each (X, Y)-partition rectangle bounded by (x_0, y_0) and (x_1, y_1), relatively indexed, by approximating the supported probability surface f with a bilinear function of X and Y as

$$f(x, y) = axy + bx + cy + d$$

where the coefficients (a, b, c, d) are

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} x_0y_0 & x_0 & y_0 & 1 \\ x_1y_0 & x_1 & y_0 & 1 \\ x_0y_1 & x_0 & y_1 & 1 \\ x_1y_1 & x_1 & y_1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} f(x_0, y_0) \\ f(x_1, y_0) \\ f(x_0, y_1) \\ f(x_1, y_1) \end{pmatrix}$$

An iso-probability level curve is then a function $y(x|z)$ for a given partition endpoint z as

$$y(x|z) = \frac{z - d - bx}{ax + c}$$

A necessary step is to compute the fraction of each (X, Y)-partition rectangle intersected each given Z-partition interval. The bilinear approximation admits the following closed form solution,

$$\int y(x|z)\ dx = \frac{log(ax + x)(az - ad + bc) - abx}{a^2} + const$$

As a numerical example suppose X has a partition element bounded by $\{1, 3\}$ and similarly Y has a partition element bounded by $\{-10, 10\}$. If Z has a partition element bounded by $\{0, 1\}$ then figure 3.7 shows that appoximately 8.25% of the probability represented by this partition rectangle is allocated to the $(0, 1)$ partition element of Z. The resulting probability distribution for Z is shown in figure 3.8.

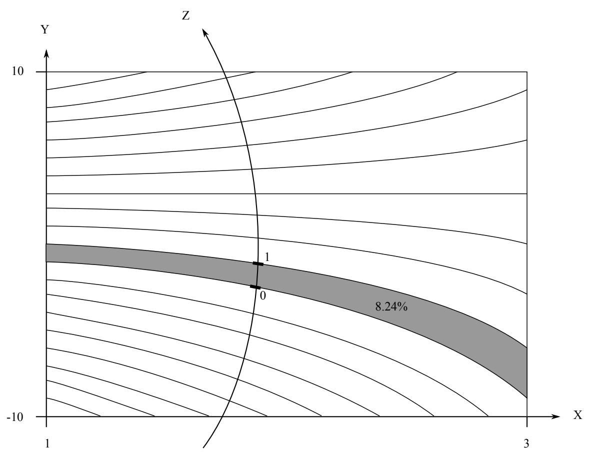

Figure 3.7: A partition element of X + Y/X with level curves

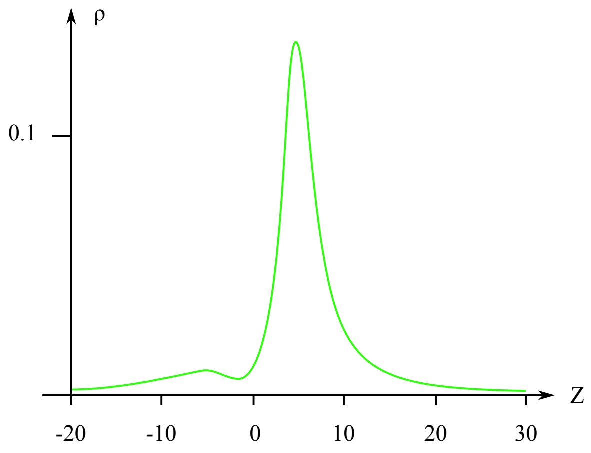

Figure 3.8: $X + Y/X$ where $X \sim N(5, 3), Y \sim N(1, 1)$

Chapter 4

Correlated Operations

Given a random variable A and two real-valued functions f and g such that $X = f(A)$ and $Y = g(A)$ let $Z = h(X,Y)$ where h is some real-value function from $\mathbb{R}^2$. The function $h(x,y) = x + y$ is of particular interest.

Since A may be a mixed random variable the development will by assuming A is discrete, then continuous and finally a general mixed random variable. In all cases X and Y are, by design, 100% correlated through A.

4.0.1 Discrete Operations on Correlated Random Variables

Suppose that,

$$A = ((a_1, a_2, ..., a_n), (p_1, p_2, ..., p_n))$$

where $Pr[A = a_i] = p_i$ for any $i \in 1...n$, that is, A is a discrete random variable. Consequently,

$$\begin{aligned} X &= ((x_1, x_2, ..., x_n), (p_1, p_2, ..., p_n)) \\ Y &= ((y_1, y_2, ..., y_n), (p_1, p_2, ..., p_n)) \end{aligned}$$

where $x_i = f(a_i)$ and $y_i = g(a_i)$ for each i. Notice that duplicate values of x_i are possible. If it happens that $x_i < x_{i+1}$ for each $i \in 1...n-1$ then X is said to be in *proper form* and similarly for A and Y. To emphasize that X and Y are derived in a pointwise order-preserving manner they may be written in *synchronous* form,

$$X = (x_1, x_2, ..., x_n)$$
$$Y = (y_1, y_2, ..., y_n)$$

where the probability values associated with each x_i and y_i are found in A. The joint probability distribution of X and Y is itself a random variable called XY. Stated in syncrhonous form,

$$XY = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$$

A new random variable $Z = h(X, Y)$ is stated in synchronous form with respect to A as,

$$Z = (h(x_1, y_1), h(x_2, y_2), ..., h(x_n, y_n))$$

To restate Z in proper form requires two steps. The first is to remove duplicates form the range of Z,

$$\mathbf{R}(Z) = \{h(x_i, y_i)\}_{i \in 1..n}$$

The second step is to find the probability associated with each element of the range of Z. Assuming the following proper form of Z as,

$$Z = ((z_1, z_2, ..., z_m), (q_1, q_2, ..., q_m))$$

where $m \leq n$ then,

$$z_j \in \mathbf{R}(Z), \text{ ordered ascending}$$
$$q_j = \sum_{i | z_j = h(x_i, y_i)} p_i$$

For example suppose,

$$\begin{aligned} A &= ((a_1, a_2, a_3), (p_1, p_2, p_3)) \\ X &= (1, 2, 3) \\ Y &= (1, 3, 2) \end{aligned}$$

The joint random variable XY in synchronous form is,

$$XY = ((1, 1), (2, 3), (3, 2))$$

Suppose $Z = h(X, Y)$ where $h(x, y) = x + y$. Then Z in synchronous form with respect to A is,

$$Z = (2, 5, 5)$$

To find the proper form of Z the range is first determined,

$$\mathbf{R}(Z) = \{2, 5\}$$

the proper form of Z is stated as,

$$Z = ((z_1, z_2), (q_1, q_2))$$

where

$$\begin{aligned} q_1 &= p_1 \\ q_2 &= p_2 + p_3 \end{aligned}$$

since $Z = 5$ is the set $\{(x_2, y_2) = (2, 3), (x_3, y_3) = (3, 2)\}$. Notice that the process of finding Z in proper form is that of integrating iso-value subsets of the joint XY range, the domain of Z.

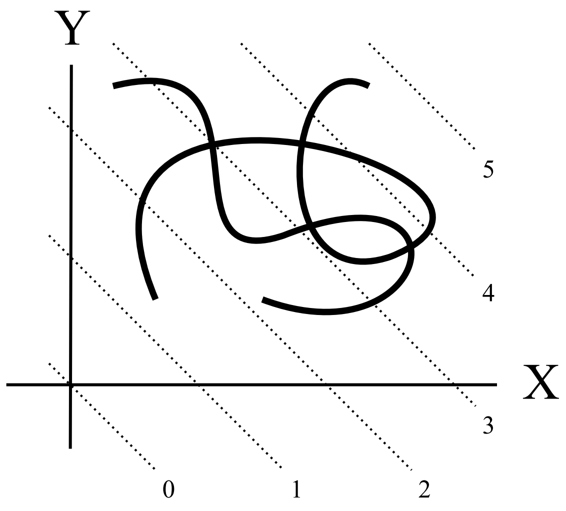

Figure 4.1: Distribution of Continuous XY

4.0.2 Continous Operations on Correlated Random Variables

Suppose A is a real-valued continous random variable. Let P be the probability density function associate with A and write $A \sim P$. The random variables $X = f(A)$ and $Y = g(A)$, in synchronous form, share this association with A, that is, $X\ P$ and $Y\ P$ for any real valued functions f and g. The joint random variable XY is also stated in synchronous form as $XY \sim P$. Finally, $Z = h(XY)$ for any real-valued function h from $\mathbb{R}^2$ is also stated in synchronous form as $Z \sim P$.

To form an interesting example consider that $X = f(A)$ and $Y = g(A)$ form a parametric curve in XY-space and that the probability density P is distributed along this curve. If $Z = h(XY)$ such that $h(x, y) = x+y$ then iso-value contours in XY-space appear as parallel lines with slope -1. In the figure 4.1 the disjoint curves are the single parametric (X, Y) curve and the dotted diagonal lines are the iso-value contours for addition of X and Y. The probability density P of A would appear in the figure perpendicular to the XY plane over the paremetric curve of (X, Y). From the figure it is apparent that $Pr(1 < Z < 5) = 1$. Notice that the iso-value contour labeled 5 in the figure intersects the XY curve at three places so that $Pr(Z = 4)$ is found as the sum of three probability density values in P.

Stating the synchronous form of Z with respect to A is as simple as for X and Y, that is, $Z \sim P$. Finding the proper form of Z may be a more challenging problem. The procedure for computing a numerical approximation to the proper form of Z is detailed in the dissertation [12]. Notice in particular that computing the proper form of a random variable may be avoided until and observation is required for reasons such as graphing or comparison to unrelated random variables and constant values.

4.0.3 Mixed Discrete / Continuous Operations on Correlated Random Variables

To prepare for the computation of operations on a pair of mixed discrete/continuous random variables dependent on a single common random variable, it is useful to develop the case where one operand is discrete and the other is continuous. This situation can only arise if A is not a purely discrete random variable.

Suppose that A is a continuous random variable such that $A \sim P$ as above. Suppose without loss of generality that X is a discrete random variable and that Y is a continuous random variable. A visual example of a possible joint random variable XY appears in figure 4.2. Included in the figure are the iso-value contours used to compute $Z = h(XY)$ where $h(x, y) = x + y$ as the previous continous example. Notice that this case is not fundamentally different from the previous case where X and Y are both continous. In the figure X has four unique values in its range labeled x_1, x_2, x_3, x_4. It is apparent from the figure that $Pr(1 < Z < 5) = 1$ and that Z is a continuous random variable. The procedure for computing a numerical approximation to the proper form of Z is detailed in the dissertation [12].

4.0.4 Operations on Correlated Mixed Random Variables

If random variable A is *mixed*, that is, containing both continous and discrete probability distributions it is useful to decompose it into discrete and continous components and write,

$$A = A_d \oplus A_c$$

where A_d is a discrete random variable and A_c is a continuous random variable and the $'\oplus'$ operator performs a sum of distribution functions by converting discrete probability to Dirac Delta functions. The components of A are written as,

$$A_d = ((a_1, ..., a_n), (p_1, ..., p_n))$$
$$A_c \sim Q$$

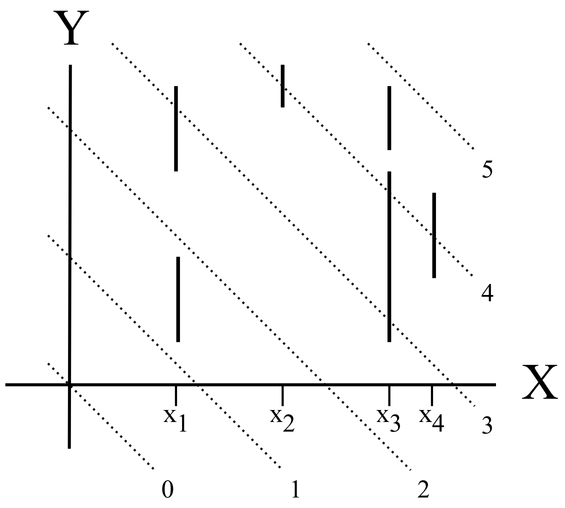

Figure 4.2: Distribution of Discrete/Continous XY

where Q is a conditional probability distribution represented by a continuous probability density function. Notice that if $d = Pr(A_d)$ then $Pr(A_c) = 1 - d$. That is,

$$d = \sum_{i=1..n} p_i$$
$$1 - d = \int_{A_c} dQ$$

where the abuse of integral notation implies that the integral is performed over the range of A_c in the usual sense. A non-trivial mixed random variable then requires that $0 < d < 1$.

Notice in particular that for special case of addition of correlated random variables the operation of addition as in $Z = X + Y$ is that of projecting the XY distribution to the diagonal as shown in figure 4.3.

As an example suppose,

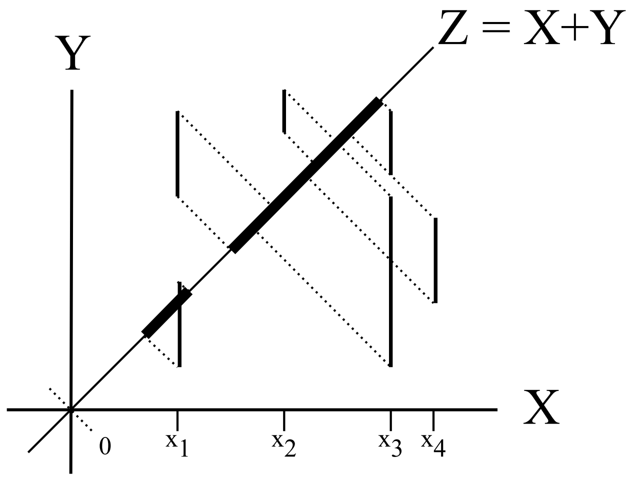

Figure 4.3: Projection of XY-space to $X+Y$-space

$$
\begin{aligned}
A &= \mathbf{U}(-1,2)\\
f(x) &= step(x) = \begin{cases} 0 & \text{if x} \leq 0,\\ 1 & \text{else}\end{cases}\\
g(y) &= |y|+1\\
X &= f(A)\\
Y &= g(A)
\end{aligned}
$$

then in proper form,

$$
\begin{aligned}
X &= ((0,1),(\frac{1}{3},\frac{2}{3}))\\
Y &= \mathbf{U}((0,1,2),(\frac{2}{3},\frac{1}{3}))
\end{aligned}
$$

where Y is *multi-uniform* requiring probabilities within partition elements to be specified. Notice that $Pr(X=0)=\frac{1}{3}$, $Pr(X=1)=\frac{2}{3}$, $Pr(0<Y<1)=\frac{2}{3}$ and $Pr(1<Y<2)=\frac{1}{3}$.

Suppose further that $Z=X+Y$. The joint XY figure 4.4 reveals the details. Noticing that the probability is uniformly distributed over the range of XY and

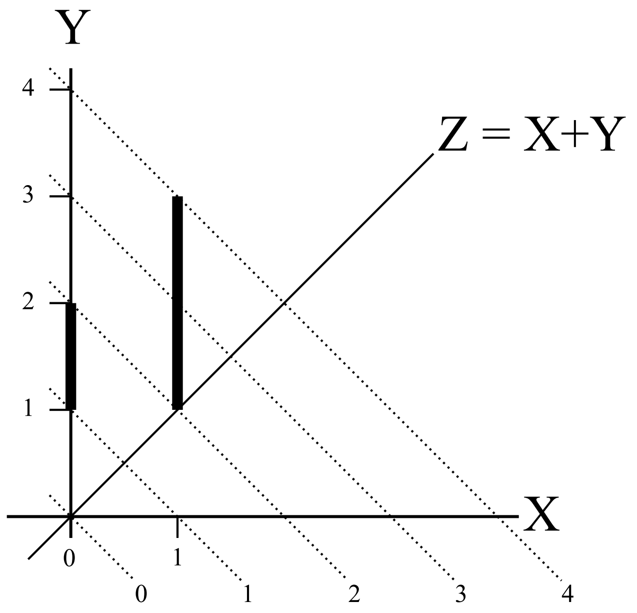

Figure 4.4: Example of Discrete/Continous XY

that the two fragments of that region do not overlap according to the iso-value contours of Z the problem is solved by inspection so that,

$$Z \sim \mathbf{U}(1,4)$$

To find this result more formally Y is conditioned on the discrete X so that,

$$Y|X = 0 \sim \mathbb{U}(1,2)$$
$$Y|X = 1 \sim \mathbb{U}(1,3)$$

then,

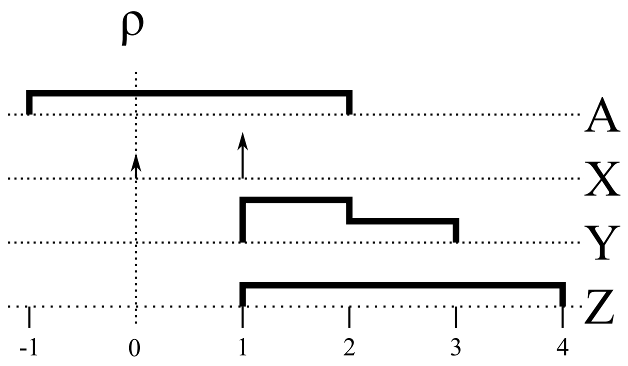

Figure 4.5: Example Discrete/Continous Distributions in Proper Form

$$\begin{aligned}
Z &= X + Y \\
&= (X + Y|X = 0) \oplus (X + Y|X = 1) \\
&= (0 + Y|X = 0) \oplus (1 + Y|X = 1) \\
&\sim \mathbf{U}(1 + 0, 2 + 0) * Pr(X = 0) + \mathbf{U}(1 + 1, 3 + 1) * Pr(X = 1) \\
&\sim \mathbf{U}(1, 2) * \frac{1}{3} + \mathbf{U}(2, 4) * \frac{2}{3} \\
&\sim \mathbf{U}(1, 4)
\end{aligned}$$

For visual convenience the proper form distributions of A, X, Y and Z are shown in figure 4.5. Notice that to compute $Z = X + Y$ from the proper forms of X and Y is more challenging than from the synchronous form of XY in figure 4.4 in part because of the otherwise unseen correlation between X and Y through A.

Chapter 5

Constrained Optimization

5.1 Tables and Chairs with Correlated Random Prices

In this final stage of the tables and chairs example consider correlated random prices. Common practice when writing about economic matters is to develop a story around the problem to help tie the elements together. This practice will be followed for this example.

Suppose that a small furniture manufacturer in Portland, Oregon wants to forecast weekly revenue. The manufacturer makes tables and chairs in a small show with a small crew. Using a forecast for demand for tables, chairs and dinette sets the manufacturer derives the likely market prices for tables and chairs. A dinette set is composed of one table and two chairs.

Figure 5.1 shows the independent random variables corresponding to forecast demand for dinette sets (the exponential curve), tables (the tall Gaussian curve) and chairs (the wide Chi-Squared curve). The vertical axis represents probability density and the horizontal axis represents demand for units (in thousands) in the Portland market.

The manufacturer believes that market price and demand for tables are related by the inverse function,

$$P_t = \frac{14 * 80}{D_t + D_d}$$

where D_t is the demand for tables alone and D_d is the demand for dinette sets. Thus $D_t + D_d$ is the total demand for tables. Similarly, the price of chairs is related to the demand for chairs by the inverse function,

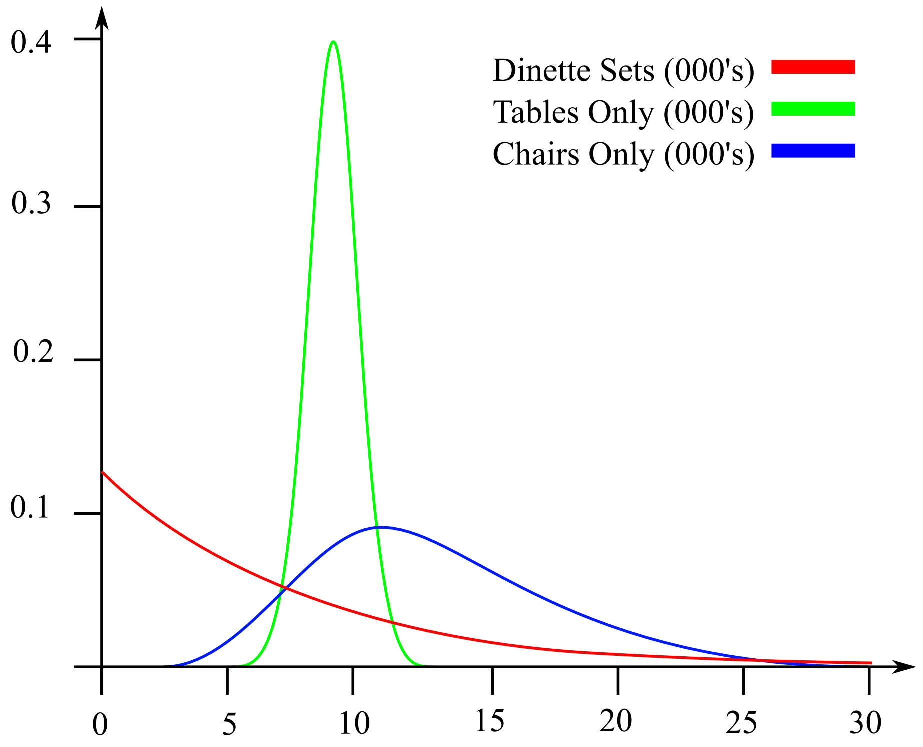

Figure 5.1: Tables, Chairs and Dinette Sets Random Variables

$$P_c = \frac{24 * 45}{D_c + 2D_d}$$

where D_c is the demand for chairs alone and again D_d is the demand for dinette sets. The sale of one dinette set implies the sale of two chairs. The actual functions are immaterial and have been contrived so that the results of this version of the tables and chairs example are comparable to previous versions.

Recognize that P_t and P_c are correlated, but do not need to materialize their joint probability distribution in order to compute revenue results.

The example is data-intensive and prototype software is used to produce numerical results. Rather than presenting the prototype, written in Python using the Numpy library, the data structures and sequence of operations will be described.

Let the input random variables be,

$$\begin{aligned} Dt &= \{DXt, DPt\} \\ Dc &= \{DXc, DPc\} \\ Dd &= \{DXd, DPd\} \end{aligned}$$

where,

$$\begin{aligned} DXt &= (DXt_1, \ldots, DXt_{Nt}) \\ DPt &= (DPt_1, \ldots, DPt_{Nt}) \\ DXc &= (DXc_1, \ldots, DXc_{Nc}) \\ DPc &= (DPc_1, \ldots, DPc_{Nc}) \\ DXd &= (DXd_1, \ldots, DXd_{Nd}) \\ DPd &= (DPd_1, \ldots, DPd_{Nd}) \end{aligned}$$

assuming that $DPt_{Nt} = DPc_{Nc} = DPd_{Nd} = 0$ as usual for the numeric random variables since probability values are between partition values. Then Nt, Nc and Nd as the number of partition endpoints for each input random variable; tables, chairs and dinette sets respectively.

Form the demand joint probability distribution DP for the input random variables by Cartesian product,

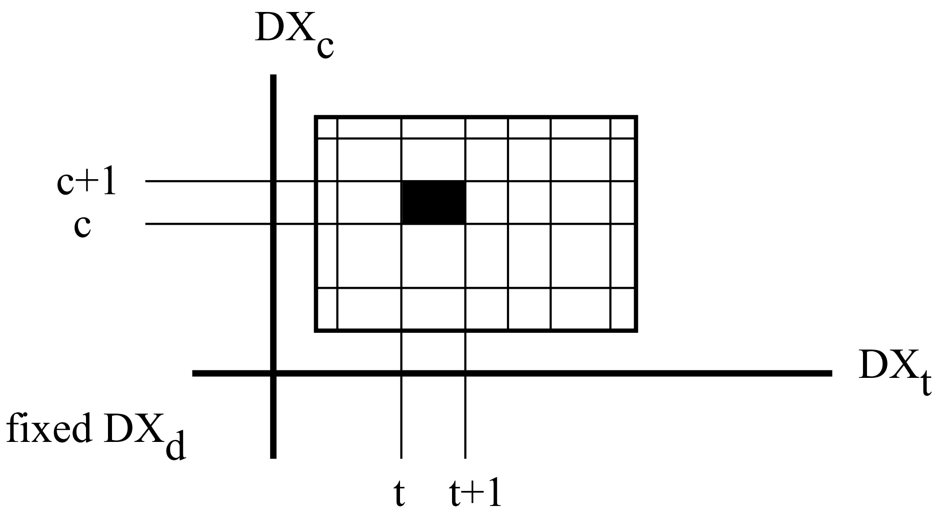

Figure 5.2: One Layer of Demand Probability Array

$$DP = DPt \times DPc \times DPd$$

Separately form parallel 3D arrays for each input demand,

$$\begin{aligned} DT &= DXt \times ones(Nc) \times ones(Nd) \\ DC &= ones(Nt) \times DXc \times ones(Nd) \\ DD &= ones(Nt) \times ones(Nc) \times DXd \end{aligned}$$

where, defining by example,

$$ones(5) = (1, 1, 1, 1, 1)$$

Throughout this example presentation 2D diagrams will tend to be preferred to represent 3D objects for clarity. In figure 5.2 the demand joint probability DP is represented for a fixed value of DXd.

In an abuse of notation, when no confusion arises, t, c and d are used as both identifiers and indices. The indices for tables, chairs and dinette set random variables are,

$$\begin{aligned} t &= 1 \dots Nt \\ c &= 1 \dots Nc \\ d &= 1 \dots Nd \end{aligned}$$

A specific point will be referred to within the demand joint probability array as $DP_{t,c,d}$ or simply DP_{tcd} where the commas are dropped for clarity. The shaded rectangle in figure 5.2 is then the demand-space rectangular block of uniform probability distribution with value DP_{tcd}.

The four 3D arrays DT, DC, DD and DP are all parallel. Create other arrays parallel to these. The reason for this parallelism is to ensure that the probability within each block is correctly tracked through each step of the computation process.

The 3D arrays for the prices of tables and chairs are then written,

$$\begin{aligned} PT &= \frac{14 * 80}{DT + DD} \\ PC &= \frac{24 * 45}{DC + 2DD} \end{aligned}$$

where the sums, $DT + DD$ and $DC + 2DD$, are computed element-wise as well as the reciprocal functions. The result is that PT and PC are 3D arrays of size $Nt * Nc * Nd$ and are parallel to the demand and demand probability arrays. Note in particular that the price arrays PT and PC are not random variables. They may be converted to random variable form as detailed below. This is possible because values of prices are located at vertices of a block of probability distribution and the value of the probability is uniformly distributed within that block in the 3D array DP.

Now that input prices are established for the manufacturer optimization may be applied to decide what combination of tables and chairs to produce. In this example the optimization is computed exhaustively and the demand space is readily paritioned into three output cases A, B and C.

Consider that each point in the 3D demand array represents a particular choice of input values that has associated with it two particular prices, one for tables and the other for chairs. It is known, for example, that if $\frac{2}{3}Pt < Pc$ then output A will be selected and similar rules apply for outputs B and C. This means that for each point in the 3D demand space a Boolean value 1 or 0 can be assigned where

1 means that point has an associated price for chairs that is larger then two-thirds that of tables. A parallel 3D array of Boolean values called a *mask* based on the optimized output selection rules results. Let,

$$\begin{aligned} MA &= \frac{2}{3}PT < PC \\ MB &= \frac{1}{4}PT < PC < \frac{2}{3}PT \\ MC &= PC < \frac{1}{4}PT \end{aligned}$$

Each output is associated with some revenue. Using the price arrays, parallel revenue arrays may be formed. Let,

$$\begin{aligned} RA &= 45PC \\ RB &= 24PC + 14PT \\ RC &= 20PT \end{aligned}$$

To convert the revenue arrays, RA, RB and RC into random variables a partition must be identified. Noticing that while the revenue arrays are parallel, the masks indicate that any given point in the array space indexed by (t, c, d) is intended to be present in exactly one revenue array. This is because the output A, B and C are mutually exclusive so, for example, the probability of producing \$1000 and \$2000 of revenue using output A can be added to the probability of producing this same range of revenue for output B and for C to arrive at a probability of producing that range of revenue regardless of output choice.

The partition used for the revenue random variable must span the range of possible revenue values and be fine where there is more revenue information and course where there is less and be so fine overall that numerical artifacts do not overwhelm the result. In this example every 23^rd point from each 175-point input demand random variable is chosen and the problem rerun on the partition values alone, not the probability values. In the Python code this amounts to a single function call since all the code is in place for the main computation. The result is are smaller versions over the same revenue arrays representing collectively a sample of the possible revenue values this example model produces using the given demand inputs. The steps are as follows,

1. Form one dimensional arrays of valid revenue values for each output.

2. Run the same process as above to generate revenue arrays and output masks. Prepend an s to the name indicating they are small versions due to the reduced partition size.

3. Concatenate the three 1D arrays into a single array called $temp$.

4. Sort the $temporary$ array and remove any duplicates.

5. append the value $-\infty$ to the start of the array and ∞ to the end. Call the result Rx.

In this case the Python code from the prototype sums up the process concisely,

$$\begin{aligned} temp &= concatenate((sRA[sMA], sRB[sMB], sRC[sMC])) \\ Rx &= concatenate(([-\infty], unique(temp), [+\infty])) \end{aligned}$$

where $sRA[sMA]$ returns a one dimensional array from an arbitrary array only for points where the corresponding point in the sMA small output mask array is a 1 and $unique()$ sorts and removes duplicates from an array.

For each (big) output array RA, RB and RC with associated masks MA, MB and MC a one dimensional array is created for the probability distribution that is parallel to the one dimensional partition array Rx.

$$\begin{aligned} Rap &= zeros(Rx) \\ Rbp &= zeros(Rx) \\ Rcp &= zeros(Rx) \end{aligned}$$

where, defining by example,

$$zeros(5) = (0, 0, 0, 0, 0)$$

The probability arrays, once filled in, will complete the formation of the output revenue random variables,

$$\begin{aligned} Ra &= \{Rx, Rap\} \\ Rb &= \{Rx, Rbp\} \\ Rc &= \{Rx, Rcp\} \end{aligned}$$

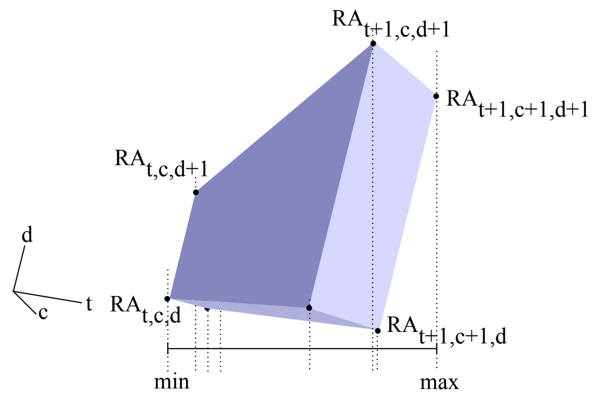

Figure 5.3: Line Projection of 3D Probability Block

The three output random variables Ra, Rb and Rc are mutually exclusive and since they share a common partition their probability values are summable to find the final output revenue random variable R,

$$R = \{Rx, Rap + Rbp + Rcp\}$$

It remains to describe how to fill in the probability arrays Rap, Rbp and Rcp. The process for Rap is the same for the others.

Given the (big) output revenue 3D array RA, its associated mask MA, the associated 3D probability array DP and the 1D revenue partition Rx the zero-valued 1D probability array Rap may be created using the following steps.

The output revenue 3D array RA together with the associated probability array DP describes a partition of the joint demand (input) space into blocks. Recall that the blocks have indices t,c and d so that the (t, c, d) block has uniform probability DP_{tcd} and eight vertices with the following revenues,

$$\begin{array}{cccc} RA_{t,c,d} & RA_{t,c,d+1} & RA_{t,c+1,d} & RA_{t,c+1,d+1} \\ RA_{t+1,c,d} & RA_{t+1,c,d+1} & RA_{t+1,c+1,d} & RA_{t+1,c+1,d+1} \end{array}$$

for some block such that $1 \leq t < Nt$, $1 \leq c < Nc$ and $1 \leq d < Nd$. If all the vertices are *valid*, that is, the associated mask value is 1 for each vertex then figure 5.3 symbolically represents one possible scenario.

The limits of the 3D block projection are the minimum and maximum revenue vertex values. That is,

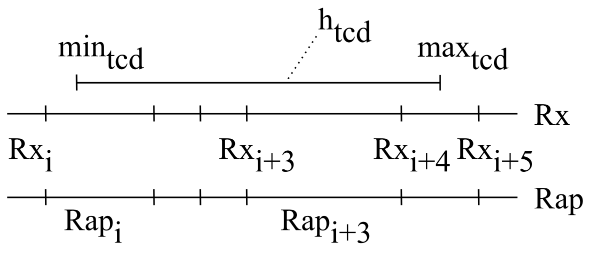

Figure 5.4: Partition Allocation of Probability Line

$$min_{tcd} = Min(RA_{t,c,d}, \dots, RA_{t+1,c+1,d+1})$$
$$max_{tcd} = Max(RA_{t,c,d}, \dots, RA_{t+1,c+1,d+1})$$

For the software prototype version of this example it is assumed that the 3D block probability $DP_{t,c,d}$ is distributed uniformly over the revenue line segment (min, max) so that the density is h_{tcd},

$$h_{tcd} = \frac{DP_{tcd}}{max_{tcd} - max_{tcd}}$$

where it is assumed that $min_{tcd} < max_{tcd}$. The special cases when min_{tcd} and max_{tcd} are equal will be detailed below. Continuing with the general case the uniform probability density h_{tcd} is allocated to the revenue probability array Rap recalling that Rap is delimited by the partition array Rx. Referring to figure 5.4,

$$\begin{aligned} Rap_i &= Rap_i + (Rx_{i+1} - min_{tcd})h_{tcd} \\ &\dots \\ Rap_{i+3} &= Rap_{i+3} + (Rx_{i+4} - Rx_{i+3})h_{tcd} \\ Rap_{i+4} &= Rap_{i+4} + (max_{tcd} - Rx_{i+4})h_{tcd} \end{aligned}$$

where it is indicated how to compute end cases as well as cases where Rx partitions are spanned by the (min_{tcd}, max_{tcd}) interval.

If the mask MA indicates that some of the vertices of the (t, c, d) block are not valid then the amount of block probability DP_{tcd} available for allocation must be reduced. For example, if 3 of 8 vertices are valid for the (t, c, d) block then the block probability is correspondingly reduced to $\frac{3}{8}DP_{tcd}$ so that the (min_{tcd}, max_{tcd}) interval probability density is,

$$h_{tcd} = \frac{3}{8}\frac{DP_{tcd}}{max_{tcd} - max_{tcd}}$$

If it happened that $min_{tcd} = max_{tcd}$ either because all the valid vertices have the same revenue value or there is only one valid vertex for the (t, c, d) block then the corresponding partition element is located for the Rap array and its value is incremented with the available probability for that block. If it happens that $min_{tcd} = max_{tcd}$ equals an Rx partition endpoint then the available block probability is halved and allocated to the adjacent partition intervals.

Perform the above operations for each output case and combine them into the full revenue random variable and show the result in figure 5.5. The horizontal axis is dollars of revenue and vertical axis is probability density as usual for random variable graphs. Notice that the median value is roughly \$2200 because of careful choice of demand inputs and the demand-to-price functional relationship.

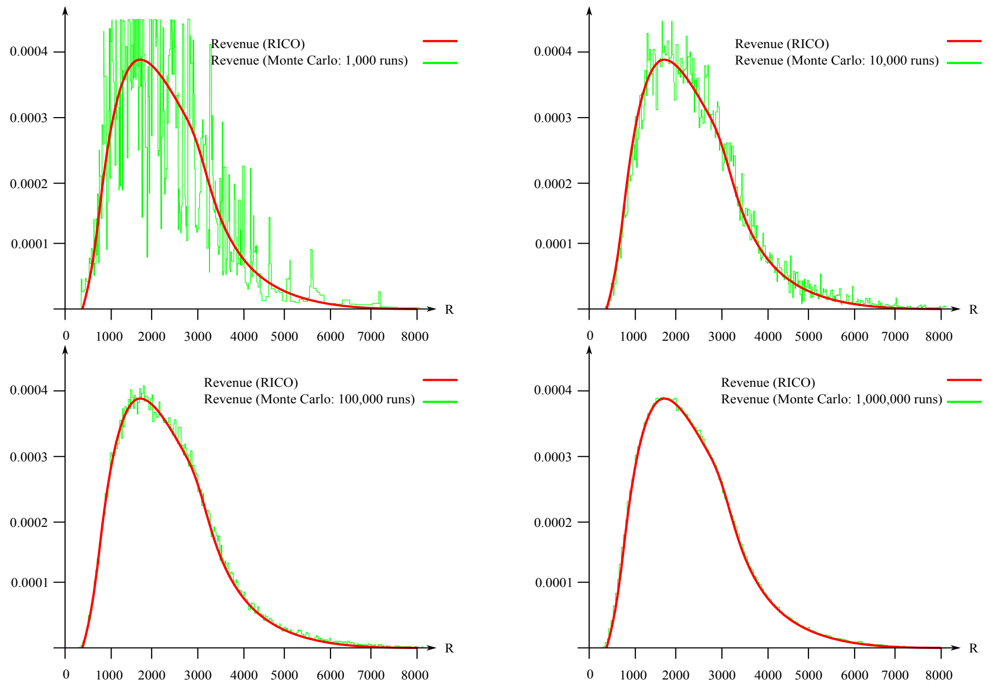

Figure 5.5: Random Revenue from Sales Combinations

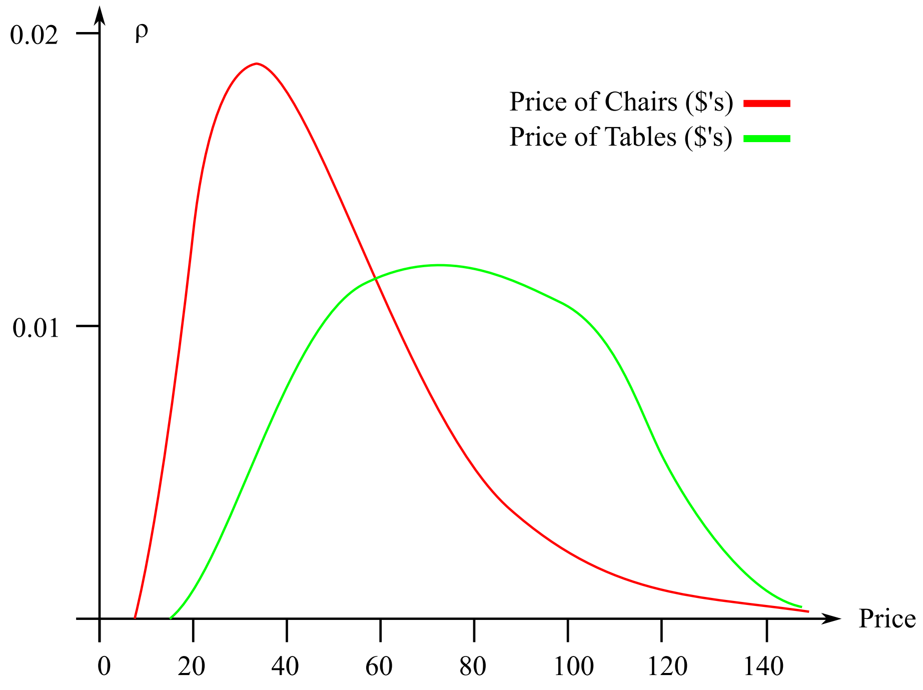

Figure 5.6: Random Variable Table and Chair Prices

Notable features of the optimized revenue in any panel of figure 5.5 is that no matter what happens with the projected demand there is a non-zero minimum revenue (about $300), a strongly likelihood of earning about $2200 and significant possibility of earning considerably more than the median $2200.

Use the machinery developed above to convert the 3D price arrays to random variables. Since there is no optimization involved in computing prices there is no need to generate masks. Since some information is available about the range of prices to expect price partitions may be chosen directly. In this case each price is partitioned into regularly spaced intervals from 0 to $150. The 3D probability arrays are projected onto the price partitions and the result for each price random variable is shown in figure 5.6. Notice that, by design, the price of chairs is has a median price of about $45 and that of tables is about $80 which corresponds with the sharp version of the tables and chairs example.

Notice that the price random variables are marginal probability distributions for a joint probability distribution are not computed. Since the price for tables and chairs are non-trivially correlated the joint distribution cannot be recovered from the marginal distributions alone as described in a standard statistics textbook such as [2].

5.1.1 Finding the Joint Price Distribution from the Demand Inputs

The reader will notice that the tables and chairs example with unknown prices is developed in preparation for the introduction of random inputs resulting in correlated prices. How to proceed with a joint distribution was described for the two prices, tables and chairs. Then the problem was solved without using, or even finding, the joint price distribution. Instead the 3D array was created to represent the three demand inputs. For this problem this technique provides directed and satisfactory results.

In this section the tables and chairs is revisited example with the same three demand input, but this time produce the joint price probability distribution. Since so much of this work is devoted to the study of correlated random variables it would be remiss not to include at least one example of same.

The problem with three 3D demand arrays, DT, DC and DD is now revisted. The associated 3D probability array DP is also available. Using the same formulas as before for finding the (correlated) prices of tables and chairs the two 3D price arrays PT and PC respectively follow.

Each 3D price array may be projected onto a price partition and produced random variable representations of the two prices in figure 5.6. Choose the same price partitions as before, evenly space intervals from \$0 to \$150. This choice allows for comparable results with those obtained previously.

The two price partitions, for PT and PC, describe a 2D partition of the (Pc, Pt)-space. If the number of points in each price partition is Np then a 2D array of size Np^2 is created and initialized with zero values.

Notice that the two 3D price arrays PT and PC together describe a 3D lattice of pairs of prices at each vertex surrounding a uniform distribution of probability described by the 3D probability array DP. The 8 vertices of each probability block, each containing the two price values, are projected onto the two dimensional (Pc, Pt)-space. This the 2D analog of the 1D procedure for finding each marginal price random variable by projecting each price block for PT or PC onto the corresponding one dimensional price line. Figure 5.7 shows an example of price block vertex projection.

Since the cluster of vertex projections in figure 5.7 indicate the projection of the 3D price probability block onto the 2D joint price partition the block proba-

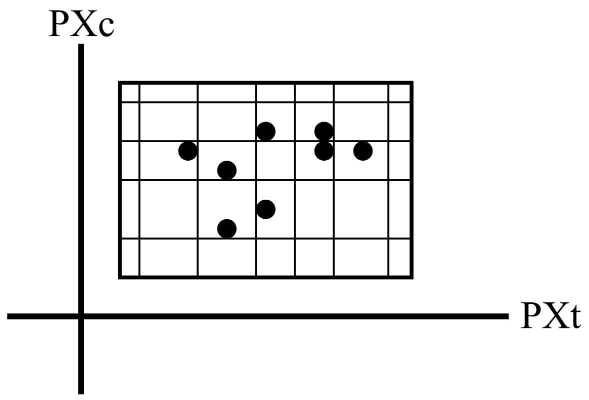

Figure 5.7: Joint Price Partition with Block Vertex Projections

bility must be allocated accordingly.

Assuming that the cluster of vertex projections represent the limits of the block projection the convex hull of these points may be found using an algorithm such as the Graham Scan as described in a textbook of computer algorithms such as Corman [7]. Assume the block probability is distributed uniformly over the interior region of the convex hull and apportion it accordingly to the partition rectangles of the 2D price distribution, called JP. Figure 5.8 shows the convex hull of the projected vertices. The heavy outline of joint price rectangles shows the limits of affected rectangles. Let p be the probability of the projected block and a the area of the convexregion, then $h = p/a$ is the probability density. The portion of probability allocated to any given rectangle in the outlined region is h times the area of the rectangle intersecting the convex region.

The tables and chairs example has over 5 million blocks to project so a complex computation of multiple rectangle intersections with convex regions associated with each block is avoided for the prototype code. A simpler, though less accurate, approach is shown in figure 5.9. The heavy outline bounding box represents the Pt and Pc partition limits bounding the block vertex cluster. The shaded inner rectangle represents the rectangular limits of the cluster points. Calculate the probability density of the block probability as if distributed uniformly over the inner shaded rectangle and distribute this by area over each intersecting price rectangle.

The results of the calculations of the prototype code for the joint probability distribution of the two correlated prices is shown in figure 5.10. Be aware that the origin is located in the upper-left corner of the graph. The x and y axis are prices

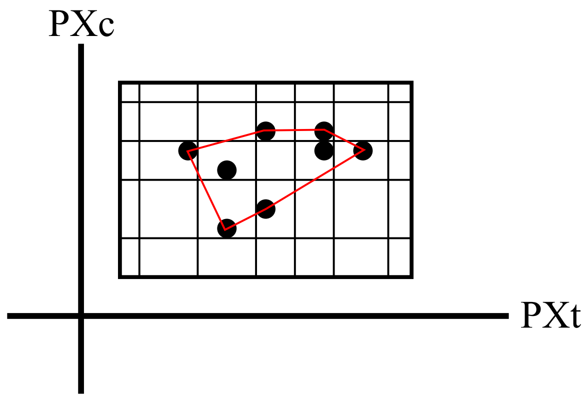

Figure 5.8: Joint Price Partition with Convex Block Projection

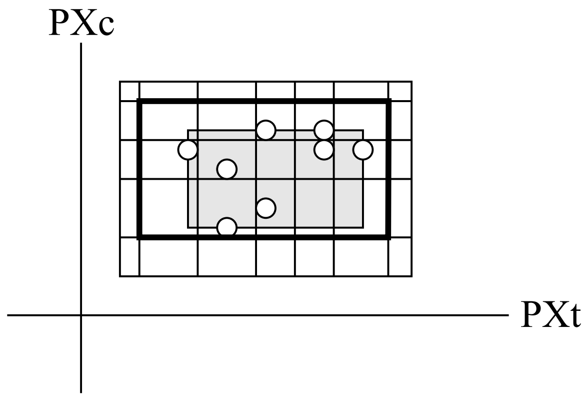

Figure 5.9: Joint Price Partition with Rectangular Block Projection

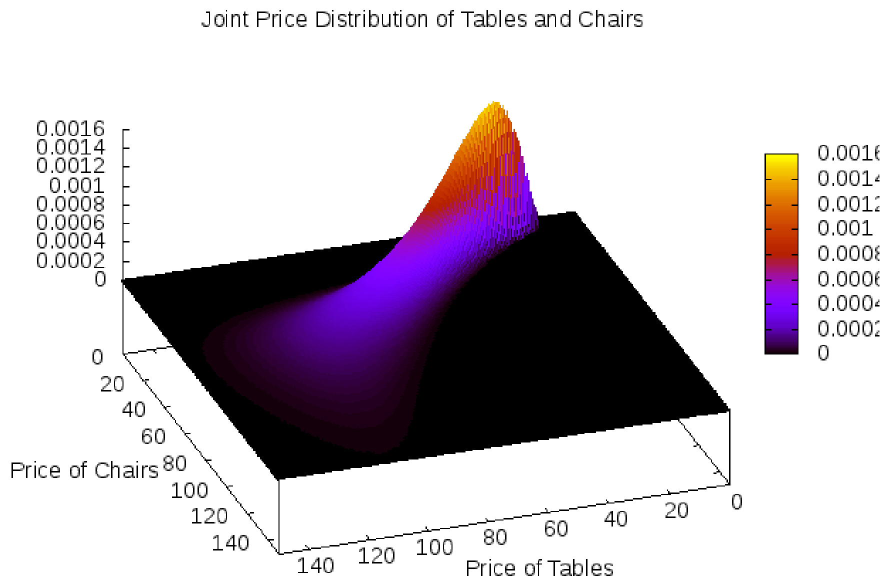

Figure 5.10: Joint Probability Distribution of Table and Chair Prices

of tables and chairs respectively and the vertical axis is probability density.

An top view of the joint probability price distribution is shown figure 5.11. Compare this figure to the original suggestion of the joint probability distribution of prices in figure B.3.

It is interesting to note that the marginal random variable prices computed with the prototype code are identical to the random variable price computed previously and show in figure 5.6.

With the joint distribution of prices in hand the question of how to compute the probability of each branch in the simplex graph (see figure B.2) is addressable. The probabilities of the branch conditional expressions are of particular interest,

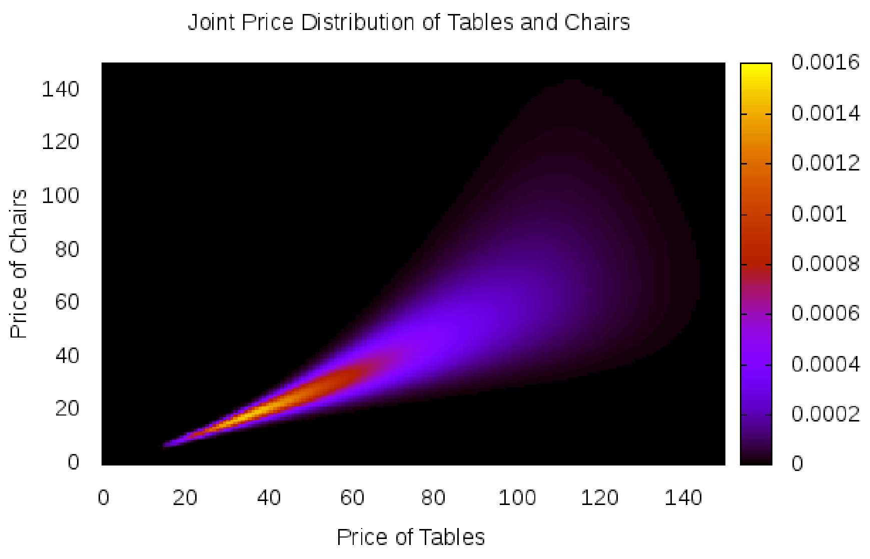

Figure 5.11: Joint Probability Distribution of Table and Chair Prices (Top View)

$$\begin{aligned}
&p_t < p_c \\
&\frac{2}{3}p_t < p_c \\
&\frac{2}{3}p_t < p_c < p_t \\
&\frac{1}{4}p_t < p_c \\
&\frac{1}{4}p_t < p_c < \frac{2}{3}p_t
\end{aligned}$$

Proceeding as before, but starting with the jointly distributed prices and their partitions P_t and P_c 2D arrays PT_2 and PT_2 are created parallel to the 2D joint probability distribution array JP. Taking the last inequality above as an example each expression is devided by p_t to find,

$$\frac{1}{4} < \frac{p_c}{p_t} < \frac{2}{3}$$

In the prototype the 2D array expression Qtc is formed as,

$$Qtc = \frac{PC_2}{PT_2}$$

as the element-wise quotient of the two 2D price arrays. Notice that Qtc together with the JP form an improper form two-dimensional random variable, a non-standard usage of the expression. If a 1D partition is chosen for Qtc the $\{Qtc, JP\}$ pair can be projected onto this partition and find a proper-form random variable, called qtc. This case is addressable by choosing the special partition,

$$Xqtc = (-\infty, 0, \frac{1}{4}, \frac{2}{3}, 1, \infty)$$

The result is,

$$Pqtc = (0, 0.0001012, 0.7206, 0.2388, 0.03024, 0)$$

0011	x_c	x_t	s_W	s_L	b
s_W	5	20	1	0	b_W
s_L	10	15	0	1	b_L
Revenue	p_c	p_t	0	0	

Table 5.1: Tables and Chairs Simplex Tableau for Unknown Prices and Resources

1001	x_c	x_t	s_W	s_L	b
s_W	1	4	$\frac{1}{5}$	0	$\frac{b_W}{5}$
s_L	0	-25	-2	1	$b_L - 2b_W$
Revenue	p_c	p_t	0	0	

Table 5.2: Tableau for Unknown Prices and Resources, State 1001

Combining these into the random variable Q for convenience as,

$$Q = \{Xqtc, Pqtc\}$$

These probability values tell us probability of each simplex directed graph branch and therefore the probability of each result. For example, referring to figure B.2, the probability of taking the first left directed edge under the condition that $p_t < p_c$ is $\mathbb{P}(1 < Q) = 0.03024$. Similarly the probability of reaching result B is $\mathbb{P}(\frac{1}{4} < Q < \frac{2}{3}) = 0.7206$.

5.1.2 Tables and Chairs with Unknowns Prices and Resources

Allowing prices and resources to be described by correlated random variables the impact on the example is to increase the number of branches from each simplex algorithm states and an increase in the number of states. The simplex tableau for unknown prices and resources is shown in table 5.1.

The directed graph for the tables and chairs example with unknown prices and resources is shown in figure 5.12. Notice that there are only $\mathbb{C}(4, 2) = 6$ possible node states in this example. Notice also that there are 5 possible terminal states; manufacture of only tables or only chair limited by either wood resource or labor resource and also the mixed case.

While the conditions present when entering a state are significant, the tableau in each state is denumerable as. Refer to tables 5.2, 5.3, 5.4, 5.5 and 5.6.

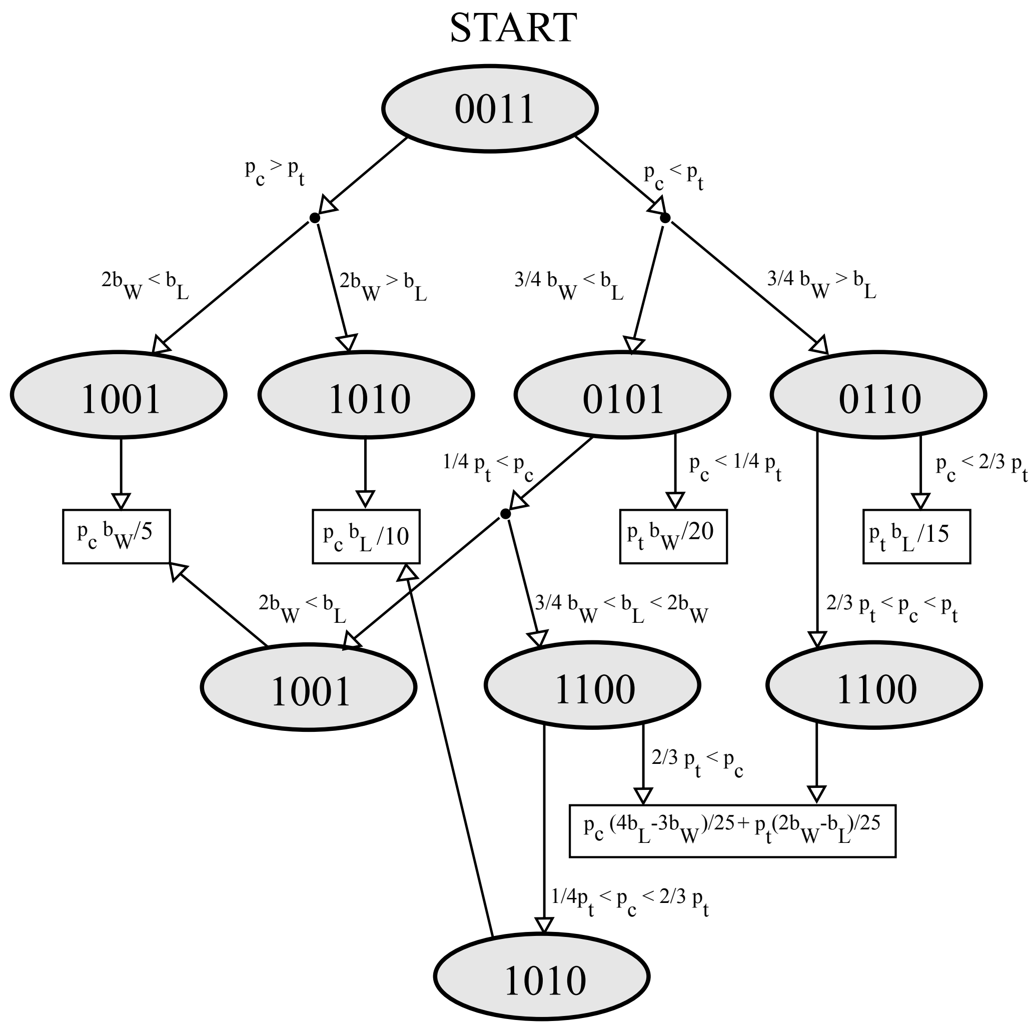

Figure 5.12: Directed Graph for Tables and Chairs with Unknown Prices and Resources

1010	x_c	x_t	s_W	s_L	b
s_W	1	$\frac{3}{2}$	0	$\frac{1}{10}$	$\frac{b_L}{10}$
s_L	0	-25	1	-$\frac{1}{2}$	$b_W - \frac{b_L}{2}$
Revenue	p_c	p_t	0	0	

Table 5.3: Tableau for Unknown Prices and Resources, State 1010

0101	x_c	x_t	s_W	s_L	b
s_W	$\frac{1}{4}$	1	$\frac{1}{20}$	0	$\frac{b_W}{20}$
s_L	$\frac{25}{4}$	0	-$\frac{3}{4}$	1	$b_L - \frac{3}{4}b_W$
Revenue	p_c	p_t	0	0	

Table 5.4: Tableau for Unknown Prices and Resources, State 0101

0110	x_c	x_t	s_W	s_L	b
s_W	$\frac{2}{3}$	1	0	$\frac{2}{30}$	$\frac{b_L}{15}$
s_L	-$\frac{25}{3}$	0	1	-$\frac{4}{3}$	$b_W - \frac{4}{3}b_L$
Revenue	p_c	p_t	0	0	

Table 5.5: Tableau for Unknown Prices and Resources, State 0110

Starting in state 0011 and following the simplex two-phase decision and first compare the two prices p_c and p_t. Assuming for clarity, as before, that since it is intended to replace p_c and p_t with continuous random variables the probability of equality is zero. In practice the possibility of equality must be checked. The initial comparisons are,

$$\begin{aligned} &argmax(p_c, p_t) \\ argmin(\frac{b_W}{5}, \frac{b_L}{10}| > 0 \text{ and } &p_c > p_t) \\ argmin(\frac{b_W}{20}, \frac{b_L}{15}| > 0 \text{ and } &p_c < p_t) \end{aligned}$$

where the > 0 condition refers to the requirement that each operand be positive else it is disqualified from the comparison.

In many cases the first or second phase of the simplex algorithm decision is disqualified since it is either non-positive or contradicts a previous assumption.

Since simplex states may be re-entered a computer algorithm can take advan-

1100	x_c	x_t	s_W	s_L	b
s_W	1	0	-$\frac{3}{25}$	$\frac{4}{25}$	$\frac{4b_L - 3b_W}{25}$
s_L	0	1	$\frac{2}{25}$	-$\frac{1}{25}$	$\frac{2b_W - b_L}{25}$
Revenue	p_c	p_t	0	0	

Table 5.6: Tableau for Unknown Prices and Resources, State 1100

tage of this possibility and cache, rather than recompute, certain elements such as the simplex tableau. Notice the comparison of linear combinations of either price or resource variables with zero, at each decision potin, in the sense that the expression $a < b$ can be rewritten as $0 < b - a$. Recall from the previous version of the tables and chairs example that each conditional statement results in a filter on the input space so that the simplex algorithm may be viewed as a filtration process. The task of the RICO modeling environment is to determine what portion of the input space passes through each facet of the simplex filtration process to a terminal node and with what probability.

5.2 Beyond the Tables and Chairs Example

In the tables and chairs example correlated random inputs for prices were chosen and it was argued that correlated random inputs could have been chosen for resources instead. Notice in the simplex algorithm the transition decision from one state to the other involves computing the maximum positive revenue impact in the case of prices and then the minimum positive resource impact. These two choices correspond to the two facets of a table pivot as explained above.

It remains to be investigated what happens to this example when some or all of the values of A are unknown. In this example the significance of unknown A values is that the manufacturer is unsure how many resources are consumed by each product.

As explained by Bellman [1], the number of solution states, not to mention the number of internal simplex algorithm states, becomes computationally intractable even for modest problems. Notice that if the problem has 100 variables and 100 constraints then the number of simplex states is at least $\mathbb{C}\approx(200, 100) = 9.05 \times 10^{58}$. The reference implementation of the AB32 model has many hundreds of variables and several thousand constraint equations.

A way to proceed is for the RICO modeling environment to partially explore the simplex directed graph. The transition from one state to the next using the simplex algorithm involves finding the maximum of a set of linear combinations of prices, in the context of the tables and chairs example, followed by finding the minimum of a set of linear combinations of resource limits assuming the A values are fixed. Each *choice element* is a linear combination of random variables which are themselves random variables. Assuming that there are three choice elements denoted X, Y and Z then the decision,

$$argmax\{X, Y, Z\}$$

results in three probability values,

$$\begin{gathered}\mathbb{P}(X < \{Y, Z\})\\ \mathbb{P}(Y < \{X, Z\})\\ \mathbb{P}(Z < \{X, Y\})\end{gathered}$$

In this case the simplex algorithm state has three initial branches corresponding to the first decision of the pivot element. These probabilities may be computed explicitly and rather than create a directed graph with all choices listed as done above with the tables and chairs directed graph, choose only the most highest probability transition. In this manner a terminal node is reached as in the sharp version of the simplex algorithm.

Since it's possible to assign probability values to each transition edge in the directed graph a choice algorithm may be applied to explore other paths based on their likelihood of occurrence. As long as the directed graph remains at least partly unexplored, it is suspected that this is the case in general, then the random variable results will not have *full probability*, that is, their probability values will sum to less than one. The proximity of the probability sum of a random variable result to unity can be used as a criterion for algorithm termination. That is, if the random variable result is deemed near enough to completion the algorithm can terminate its exploration of the simplex directed graph.

Chapter 6

Black Scholes Construction

6.1 Black Scholes Construction

Consider a financial security such as a stock S with current market price S_0. At some future time T let the price of S be represented by a random variable S_T whose probability distribution is indicated in figure 6.1. Here, no presumption is yet made about the kind of distribution for S_T.

Following Dineen [8] suppose now that S_T satisfies the so-called *no-arbitrage requirement* $S_0 = \mathbb{E}[S_T]$. Suppose an investor, Ivan, holds one unit of stock S as the sole content of his portfolio of assets. Initially his wealth W is then just,

$$W_0 = S_0$$

and at time T his wealth is expressed as,

$$W_T = S_T$$

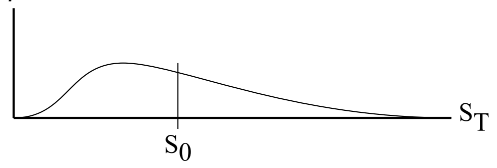

Figure 6.1: Distribution of S_T

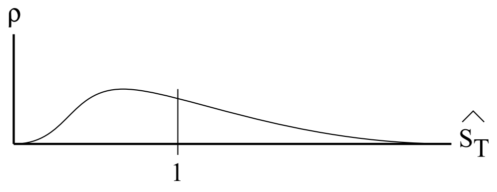

Figure 6.2: Distribution of $\hat{S_T}$

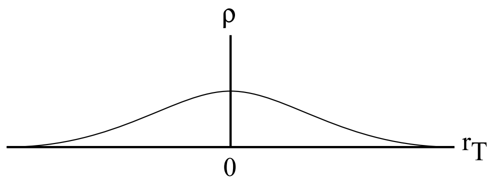

Figure 6.3: Distribution of r_T

Expressing Ivans future wealth in terms of his current wealth and a continuously compounding growth rate r one writes

$$W_T = W_0 e^{r_T}$$

where

$$r_T = log(\frac{S_T}{S_0}).$$

Referring to $\hat{S_T} = S_T/S_0$ as the *normalized* stock price rescale the probability distribution S_T to $\hat{S_T}$ so that the mean of $\hat{S_T}$ is one as depicted in figure 6.2.

Since r_T is the log of $\hat{S_T}$, its mean is zero as illustrated in figure 6.3. Notice that if $\hat{S_T}$ is represented by a LogNormal random variable then r_T is represented by a Normal random variable with zero mean.

If instead of holding a unit of stock S, the investor Ivan holds a European-style call option C based on stock S with strike price K exercisable at time T. The value of C at time T is given by

$$C_T = [S_T - K]^+$$

similarly, the related put-option P with the same parameters as S has value P_T at time T with

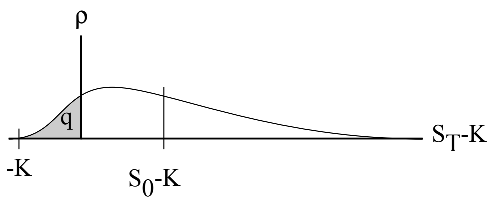

Figure 6.4: Distribution of $S_T - K$

$$P_T = [S_T - K]^-$$

It is helpful to notice that $C_T, P_T \geq 0$. For ease of calculation, the assumption is made that upon maturity all options are settled in cash and that there is no requirement that the *underlying* stock be transferred between parties. A further assumption made here is that all cash values are stated in current (t = 0) dollars. Please note that this runs counter to the not uncommon practice in financial texts, e.g. [8] to include the effect of time-value-of-money in calculations. While the current dollars assumption is equivalent to zero interest rate, the understanding is that all cash may be restated in any chosen year denomination.

To find the probability distribution over the mature value C_T of the call option C at time T with the underlying stock S and strike price K, a sequence of transformations of S_T are necessary. The first transformation is to subtract the strike price K from the distribution S_T. The resulting distribution is illustrated in figure 6.4. The key feature is represented by the shaded region in figure 6.4 is the probability q, such that $q = Pr\{S_T - K \leq 0\}$.

The second transformation, the probability distribution of $[S_T - K]^+$ is illustrated in figure 6.5. Notice that the probability q is now concentrated discretely at zero and represented by a vertical arrow labeled q. Since the remaining portion of the distribution is continuously distributed, the vertical axis still represents probability density. Notice further that the mean μ_{C_T} of this mixed, continuous/discrete probability distribution, is indicated at some positive location in figures. Notice that as $C_T = [S_T - K]^+$,

$$\mu_{C_T} = \mathbb{E}[[S_T - K]^+] = \mathbb{E}[C_T]$$

The significance of μ_{C_T} in figure 6.5 is that this price, based on the no-arbitrage principle, an investor can expect to pay today $(t = 0)$ for a call of this

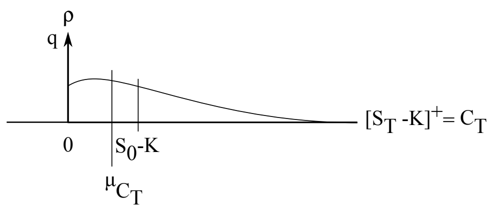

Figure 6.5: Distribution of $[S_T - K]^+$

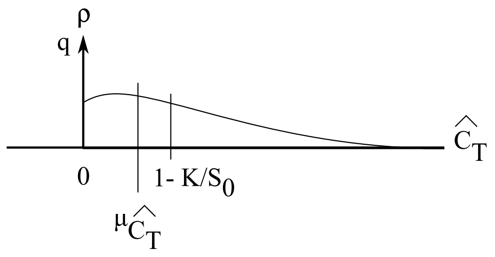

Figure 6.6: Distribution of $\hat{C_T}$

type when S_T represents the commonly accepted probability distribution of the value of stock S at time T. That is,

$$C_0 = \mu_{C_T} = \mathbb{E}[C_T]$$

Normalizing the distribution of the mature call option with respect to the initial value of the underlying S_0 gives the distribution for $\hat{C_T}$, the normalized call option price, illustrated in figure 6.6.

Using the same strike price K and underlying stock S, a put option value at maturity is illustrated in figure 6.7 where value at maturity of the put option is,

$$P_T = \mathbb{E}[[S_T - K]^-]$$

and the no-arbitrage price of the put is,

$$P_0 = \mu_{P_T}$$

Note that the probability concentrated at zero here is $1-q = Pr\{S_T - K < 0\}$. The *put-call parity formula*,

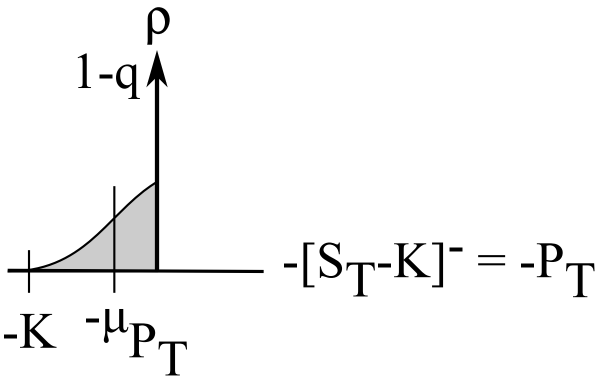

Figure 6.7: Distribution of $-[S_T - K]^-$

$$\mathbb{E}[C_T] - \mathbb{E}[P_T] = S_0 - K$$

is now obtained easily using the notation stated above, noting that

$$\begin{aligned}
\mathbb{E}[C_T] - \mathbb{E}[P_T] &= S_0 - K \\
\mathbb{E}[C_T - P_T] &= S_0 - K \\
\mathbb{E}[[S_T - K]^+ - [S_T - K]^-] &= S_0 - K \\
\mathbb{E}[S_T - K] &= S_0 - K \\
\mathbb{E}[S_T] - K &= S_0 - K \\
S_0 - K &= S_0 - K
\end{aligned}$$

as in Dineen [8], with the exception that Dineen [8] denotes as C_T the expected value of the probability distribution denoted C_T in this paper and similarly for P_T.

There is a geometric interpretation of Put-Call Parity. This can be seen in the perspective figure 6.8, where the probability density axis is perpendicular to the $C_T \times (P_T)$-space. PutCall Parity is realized in that figure as the elementary projection of the joint $C_T \times (PT)$-space to the diagonal $C_T + (P_T)$-space. The analogous algebraic calculation procedure is detailed in the appendix.

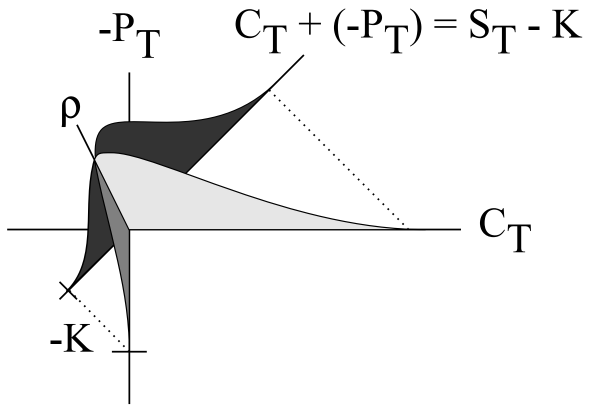

Figure 6.8: Distribution of $C_T - P_T = S_T - K$ in Perspective

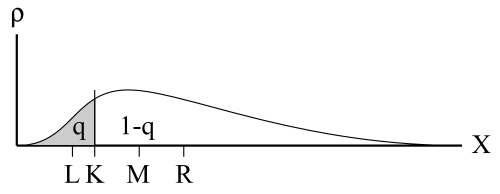

Figure 6.9: Probability Distribution Partitioned at K

6.1.1 Geometric Black-Scholes Pricing

Suppose now that a random variable X has probability density function $\rho(x)$ and is "well behaved" enough to have mean $M = \mathbb{E}[X] < \infty$. Suppose further there is a partition point K as shown in figure 6.9 and let $q = Pr\{X < K\}$ and so $1 - q = Pr\{X \geq K\}$.

Form two new *truncated* random variables from the left and right sections of the K-partitioned random variable X with probability densities,

$$X_L \sim \rho_L(x) = \frac{\rho(x)\mathbf{1}_{X<K}}{q}$$
$$X_R \sim \rho_R(x) = \frac{\rho(x)\mathbf{1}_{X\geq K}}{1-q}$$

as depicted in figure 6.10 where the L and R are expected values of X_L and X_R. Notice that the affine combination of L and R via q, illustrated in figure 6.11, yields the original mean M,

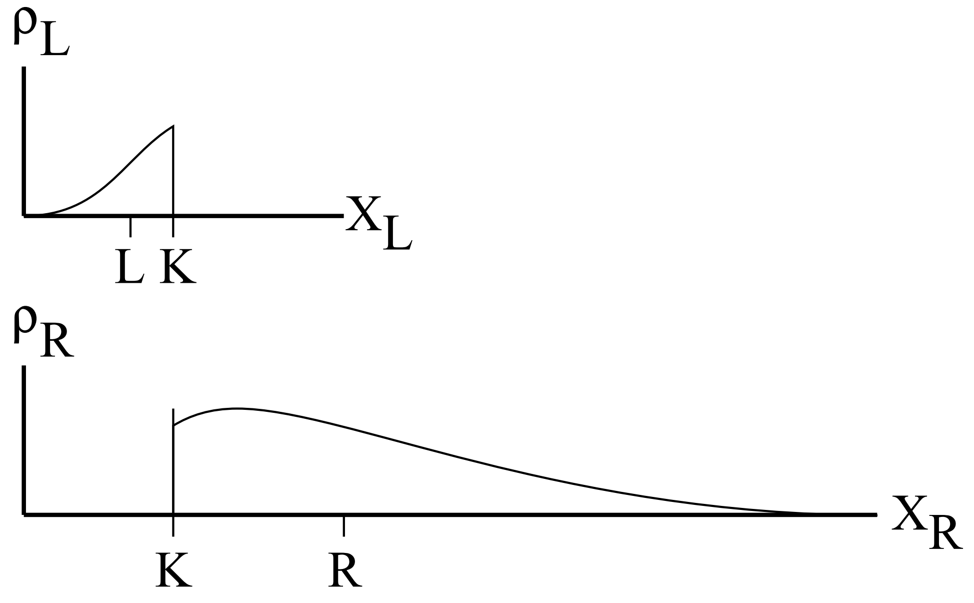

Figure 6.10: Left- and Right-Truncated Random Variables

$$M = q * L + (1 - q) * R$$

Toward Geometric Black-Scholes pricing define new random variables $Y = K + [X - K]^+$ and Z, a discrete random variable such that $Pr\{Z = K\} = 1$ with discrete density $\rho_Z = \delta_{z=K}$. Then the density ρ_Y of Y is

$$\rho_Y(y) = q * \rho_z(y) + (1 - q) * \rho_R(y)$$

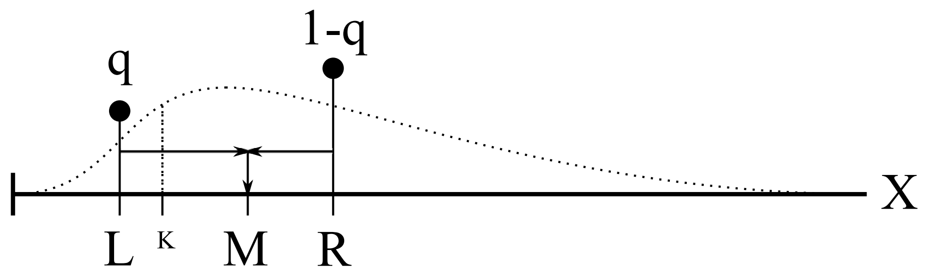

Figure 6.11: Mean Relationship Between Bifurcated Sections

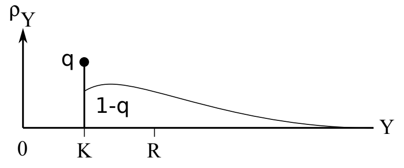

Figure 6.12: Positive Density

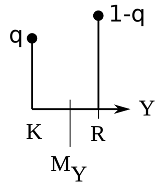

Figure 6.13: Geometric Black-Scholes Pricing

and depicted in figure 6.12. Note in particular that Y is a *mixed* discrete and continuous random variable.

To compute the mean of Y, the X_R portion of Y may be replaced with a discrete density $(1-q)$ at the mean R of X_R, as illustrated in figure 6.13. The mean M_Y of Y is then the affine combination

$$M_Y = \mathbb{E}[Y] = q * K + (1-q) * R$$

Finally Geometric Black-Scholes pricing requires computing the mean of $[X - K]^+$ for some random variable X such that $\mathbb{E}[X] < \infty$ and constant K. Using the notation developed in this section the result is expressed immediately as

$$\mathbb{E}[[X-K]^+] = \mathbb{E}[Y] - K$$

and more conveniently as,

$$\mathbb{E}[[X-K]^+] + K = q * K + (1-q) * R$$

As an illustrative example the traditional Black-Scholes formula is recovered by letting $X = S_T$ be a LogNormally distributed random variable representing the stock price of the underlying asset at some future time T. To use Geometric Black-Scholes to price a European-style call option with maturity time T and strike price K let $C_T = [X - K]^+$ and find the current price of the call option as $C_0 = \mathbb{E}[C_T]$. Following the steps above symbolically first identify the probability density of S_T,

$$\rho(x) = \frac{1}{x\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{ln(x)-\mu}{\sigma})^2}$$

Following Dineen [8], the current price S_0 is the mean of S_T,

$$S_0 = \mathbb{E}[S_T] = e^{\mu + \frac{\sigma^2}{2}}.$$

Since S_0 is a known quantity it may be used to solve for the unknown LogNormal parameter μ,

$$\mu = ln(S_0) - \frac{\sigma^2}{2}$$

and the parameter σ, according to standard interpretations, e.g. Dineen [8], represents the asset volatility and must be given. The option price C_0 is expressed by the formula derived above as

$$\begin{aligned} C_0 + K &= q * K + (1-q) * R \\ C_0 &= (1-q) * R - (1-q) * K \end{aligned}$$

Finding the mean R of the right-truncated LogNormal and its associated probability $(1-q)$,

$$\begin{aligned} R &= \frac{1}{1-q}\frac{1}{\sqrt{2\pi}\sigma}\int_K^\infty e^{-\frac{1}{2}(\frac{ln(x)-\mu}{\sigma})^2}\, dx \\ &= \frac{1}{1-q} e^{\mu+\sigma^2/2} \Phi\left(\frac{-ln(K)+\mu+\sigma^2}{\sigma}\right) \\ &= \frac{1}{1-q} S_0 \Phi\left(\frac{ln(S_0/K)+\sigma^2/2}{\sigma}\right) \end{aligned}$$

and

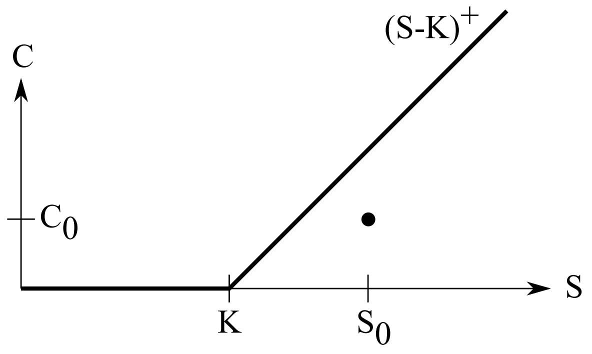

Figure 6.14: Stock-Call Space

$$\begin{aligned}1 - q &= \Phi\left(-\frac{ln(K) - \mu}{\sigma}\right)\\ &= \Phi\left(\frac{ln(S_0/K) - \sigma^2/2}{\sigma}\right)\end{aligned}$$

where

$$\Phi(z) = Pr(Z <= z) \text{ for } Z \sim Normal(\mu, \sigma^2)$$

the traditional Black-Scholes formula follows immediately,

$$C_0 = S_0\Phi\left(\frac{ln(S_0/K) + \sigma^2/2}{\sigma}\right) - K\Phi\left(\frac{ln(S_0/K) - \sigma^2/2}{\sigma}\right)$$

6.1.2 Portfolio Construction

Given a stock S represented at time $t = 0$ by value S_0 and at $t = T$ by random variable S_T and a European call option C based on S with strike price K at time T the domain of the joint probability distribution of S and C is shown in figure 6.14. The point (S_0, C_0) corresponds to the joint initial price of the stock and call. The broken line is the familiar curve for a call option graph. Perpendicular to the $SC - plane$ is the joint probability density of S and C at time T. The probability density is zero at points away from the broken line.

Since the SC-space represents the price per unit of each financial security there is a dual space where each point represents the number of units held in a hypothetical portfolio. This dual space is referred to here as the portfolio space. Given a portfolio vector π and a point w in SC the dollar value of pi at w is $\pi^T\ w$. Recall that the projection matrix of SC onto a vector v in SC is given by,

$$\Pi = \frac{v\ v^T}{v^T\ v}$$

Notice that given portfolio $\pi = (\pi_s, \pi_c)$ the random variable $Z_T = \pi_s * S_T + \pi_c * C_T$ appears graphically as a hyperplane in SC-space. Notice in particular that if the projection vector in SC-space is π itself then the projection matrix becomes,

$$\Pi = \frac{\pi\ \pi^T}{\pi^T\ \pi}$$

and more to the point the portfolio value V given $x \in SC$-space is,

$$\begin{aligned} V &= \pi^T\ x \\ &= \pi^T \frac{\pi\ \pi^T}{\pi^T\ \pi} x \\ &= \pi^T \Pi x \end{aligned}$$

which means the value of a portfolio, π, may be visualized by representing π as a vector in SC-space, projecting any other point in SC space orthogonally onto π and then computing the inner product of π and the projected point to find the portfolio value.

Suppose, for example, $\pi = (2, 1)$, that is the portfolio contains two shares of stock and one call option with strike price K. The initial price of the stock is S_0 and the initial price of the call is C_0 as usual. This situation is depicted in figure 6.15. The portfolio is represented by Z, the linear combination of S and C. Notice that if $S_T = 0$ then $Z = 0$. If $S_T = K$ then $Z = 2K$ since the call expires out-of-the-money and the value of the portfolio reflects the two shares of stock alone. Geometrically the $Z = 2K$ is found by orthogonally projecting the point $(K, 0)$ to the $\pi = (2, 1)$ vector (the Z-line) and measuring the result with the dual vector, also π to find $V = 2 * K + 1 * 0$. The initial value of the portfolio is shown graphically as Z_0 which is consistent with the computed value

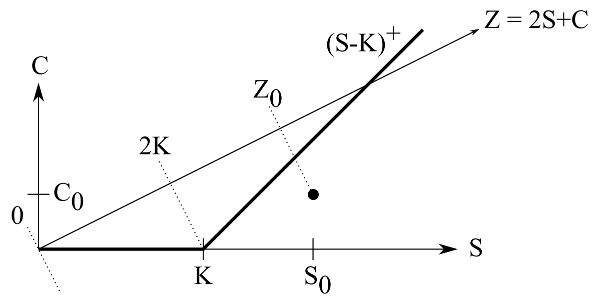

Figure 6.15: Stock-Call Space with (2,1) Portfolio

of $Z_0 = 2 * S_0 + 1 * C_0$. This construction is consistent with the understanding that random variables are represented in a joint space and observed individually through projection.

Orthogonal projection implies the existence of a null space hyperplane. A natural construction is that of a portfolio whose initial value lies in this null space. The initial cost of such a portfolio is zero. Suppose that a call option with strike price K costs C_0 with underlying stock of initial cost S_0 and that $S_0/C_0 = 5$, for example. A zero-cost portfolio is $\pi = (-1, 5)$, that is, sell one share of stock and buy 5 call options. This situation is depicted in figure 6.16. The zero-cost portfolio is represented by $Z = -S + 5C$. Notice that $min(Z) = -K$ and that if the final stock price S_T is $2K$ instead of K the value of the portfolio Z jumps from $-K$ to $3K$, four multiples of K since the portfolio contains 5 calls and one shorted share of stock whereas the terminal stock price rising form 0 to K results in Z falling from 0 to $-K$ since it contains the shorted stock and 5 out-of-the-money call options.

The probability distribution of the example Z is approximated in figure 6.17 where the negative-put-like behavior is shaded and marked with its total probability q assuming S_T is represented by a $LogNormal$ random variable. Notice in particular that the probability distribution of Z is not recognizable as either $LogNormal$ nor truncated $LogNormal$.

6.1.3 Black Scholes Construction: SPY Example

The work on Levy-Stable distributions by Nolan [16] takes the daily prices of a particular stock (ticker: SPY) shown in figure 6.18. The daily returns for SPY are plotted in figure 6.19.

According to Nolan [16] a good fit of the SPY returns is achieved by a mixture

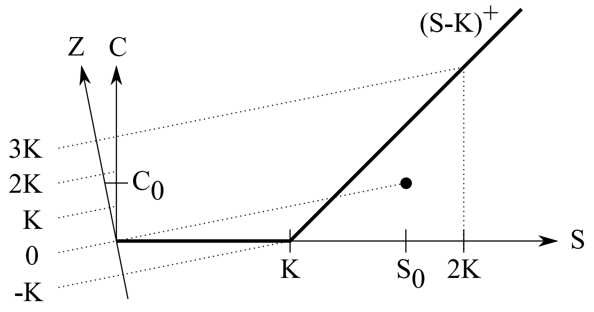

Figure 6.16: Stock-Call Space with Zero-Cost Portfolio

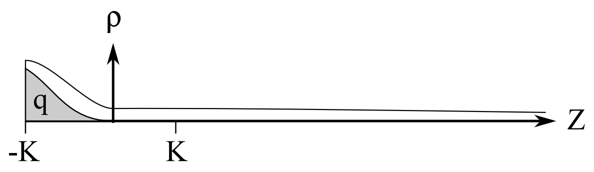

Figure 6.17: Probability Distribution of Zero-Cost Portfolio

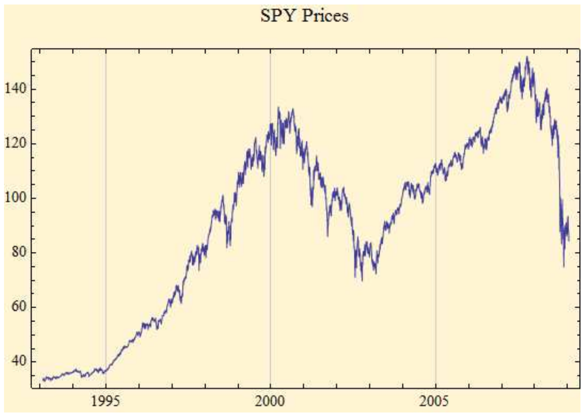

Figure 6.18: Stock Price (ticker symbol: SPY)

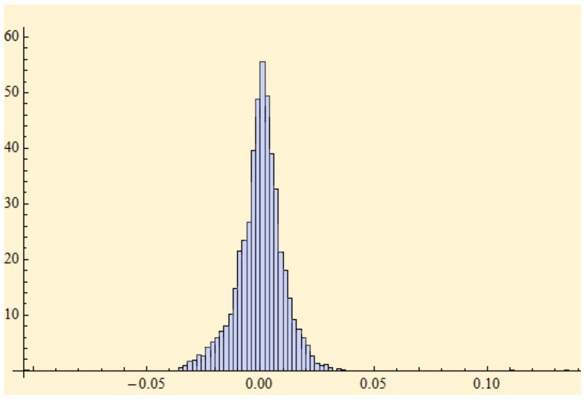

Figure 6.19: Stock Price (ticker symbol: SPY) daily return distribution

of a LogNormal distribution and a Levy-Stable distribution. Using the numeric random variable facilities of RICO to plot $LogNormal(0,\sigma)\times LevyStable(\alpha,\beta,\delta,\gamma)$ for the specific fit parameters cited by Nolan [16],

```
alpha = 1.86034
beta  = -0.0919429
gamma = 0.00600552
sigma = 0.532775
delta = 0.000232571
u     = log(gamma)
LN    = LogNormalNumeric(0, sigma, 100)
LS    = LevyStableNumeric(alpha, beta, delta, gamma, 100)
LNS   = LN * LS
Plot().xrange(-.1, .15).plot(LNS).show()
```

The fit found by Nolan [16] is overlayed on the SPY returns histogram in figure 6.20. Suppose the current price of SPY is \$80/share. Following the Black Scholes construction, the share price of SPY at $T =$ one day in the future is denoted S_T defined as,

$$S_T = 80 \times LNS$$

where LNS is the LogNormal-LevyStable distribution given the fit data found by Nolan [16]. Continuing from the previous code listing, the dsitribution of S_T is computed by RICO as in the following listing with the result is shown in figure 6.21.

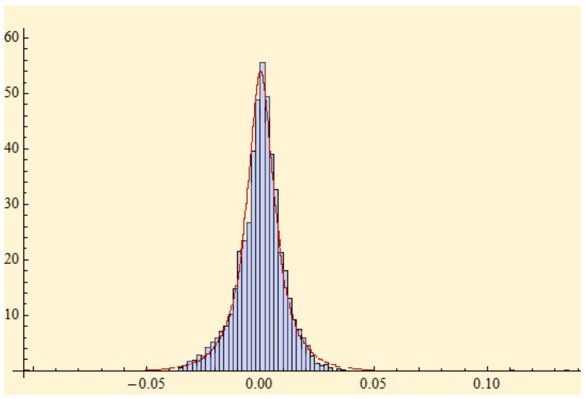

Figure 6.20: Stock Price (ticker symbol: SPY) daily return distribution

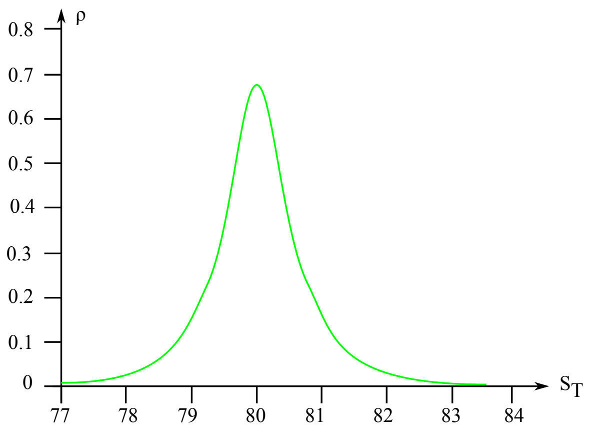

Figure 6.21: SPY one day in future given $80/share price today

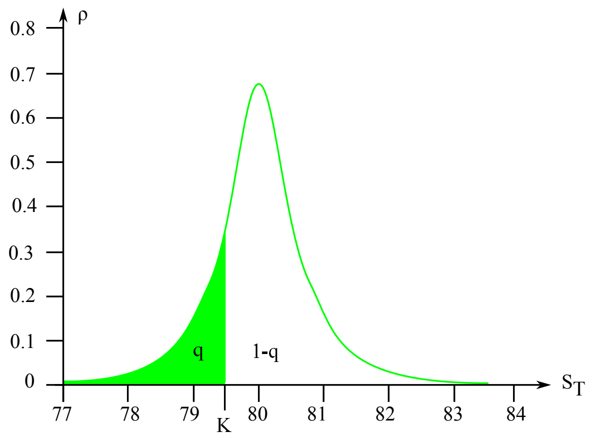

Figure 6.22: SPY one day in future given \$80/share price today

```
ST = 80*exp(LNS)
Plot().xrange(77,84).yrange(0,.8).plot(ST).show()
```

Supposing that the strike price for a 1 day option on SPY is $K = \$79.50$. The probability distribution of S_T is then split at K into two pieces as shown in figure 6.22. Let

$$q := P(S_T < K)$$

In this case $q \approx 0.22$ and is represented by the shaded region of figure 6.22. The payoff random variable of a 1-day European-style call option according to the Black-Scholes construction C_T is the following function of S_T

$$C_T = [S_T - K]^+$$

shown in figure 6.23. Notice that C_T is both discrete and continuously distributed. In particular,

$$P(C_T = 0) = q \qquad\qquad P(C_T > 0) = 1 - q$$

The salient point of this example is that the next step of the Black-Scholes construction cannot be completed for ths example. The reason is that the expected value of the continuous portion of C_T is infinite! For the fit parameters used in this

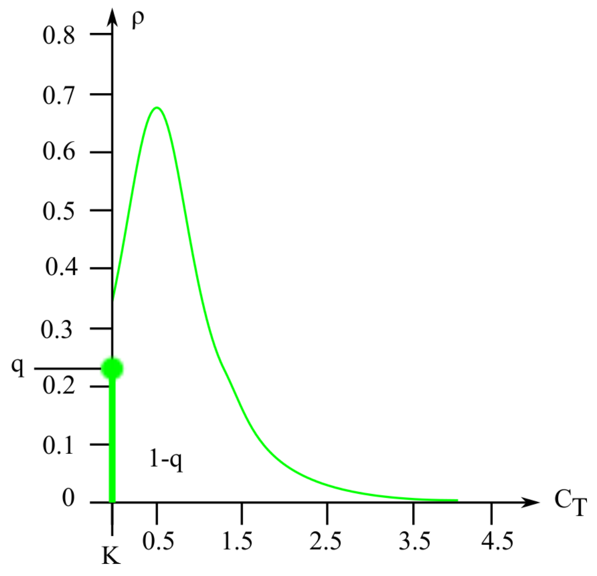

Figure 6.23: Call option payoff on one-day SPY

example the LNS distribution is *fat tailed* and the example ends without a price for the one-day call option, C_T.

In practice return rates do not necessarily follow a Gaussian distribution. The Black Scholes construction remains computable so long as expected values of truncated distributions are finite. Note in particular that Monte Carlo techniques may not expose this fact.

Appendix A

Exhaustive Bounded Root Finding Method

This section follows the *EveryRoot* method developed by Robert C. Tausworthe [18] under contract with Raytheon, Inc.

Given an objective function f defined on a normalized *investigation* interval, $[0, 1]$, the EveryRoot method finds nearly every root by exhaustive investigation. The EveryRoot method amalgamates several well-known root finding methods. By convention assume that $f_0 = f(0)$, etc.

A.0.4 Specific Assumptions

1. The objective function is presumed continuous on the investigation interval.

2. Given root r it is presumed no other roots exist within interval $[r - xGuard, r + xGuard]$ for a prespecified tolerance value, $xGuard$.

3. A value r is deemed to be a root if $|f(r)| \leq \epsilon f$ for prespecified tolerance value ϵf and $|f(r \pm xGuard)| > \epsilon f$.

4. If successive root estimates r_i and r_{i+1} differ by less than prespecified tolerance value ϵr then r_i is deemed to be a root of f.

5. If an interpolating polynomial L of f differs from the objective function at prespecified points within the interval by less than prespecified tolerance ϵL then L is deemed a *reliable* facsimile of f.

A.0.5 Method Outline

The general approach of the EveryRoot method is to recursively break the investigation interval $[0, 1]$ into subintervals ultimately creating a partition where every subinterval contains exactly one or no roots.

For any subinterval either none, one or both of the endpoints. If an endpoint is a root r then it is centered in a *guard* interval, $[r - xGuard, r + xGuard] \cap [0, 1]$ where $r \in \{0, 1\}$ and $[0, 1] - ([r - xGuard, r + xGuard] \cap [0, 1])$ becomes the new investigation subinterval.

Given the endpoints of $[0, 1]$ are not roots, if $f_0 f_1 < 0$ then at least one root exists in interval $(0, 1)$. To find one of the possible roots in $(0, 1)$ a *bracketing* method such as Brent's Method [5] or the much more recent Improved Brent's Method [20] is used. The internal root r is found and guarded. The two flanking subintervals $[0, r - xGuard]$ and $[r + xGuard, 1]$ are then investigated.

If $f_0 f_1 > 0$ then the objective function f is fit using an *optimized* cubic Lagrange interpolation polynomial L. The optimization comes in the form of two specific function evaluation points within the investigation interval plus the two endpoints. The interpolation method is detailed below. The polynomial L is deemed reliable upon evaluating it at three specific points within the interval where L is most likely to differ from f and finding that the relative error in all three cases is within ϵL tolerance. If L is un-reliable then the investigation interval is bisected into two subintervals.

Suppose L is a reliable facsimile of f and $f_0 f_1 > 0$. If L has an extrema e such that $f_0 L_e < 0$, i.e. on the other side of zero from f_0 and f_1, and subsequently $f_0 f_e < 0$ then the investigation interval is subdivided into $[0, e]$ and $[e, 1]$ so that a bracketing method may be employed. If L_e is *relatively close* to zero then a *root polishing* method such as Newton-Raphson or Halley's method maybe employed beginning at some other point in the interval. The method author suggests that if an extrema differs from zero with relative error less than a factor of three (3) of the fit tolerance ϵL then L_e is relatively close to zero. If there is no extrema of L within the interval or none of the extrema are relatively close to zero then the interval is deemed to contain any roots.

A.0.6 The *EveryRoot* Method

```
Write the algorithm, find test cases, then write
                 this subsection.
```

TO DO

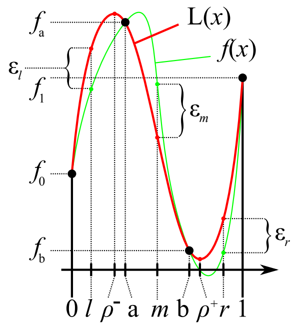

Figure A.1: Example Cubic Lagrange Interpolation

A.0.7 Optimized Cubic Lagrange Interpolation

Given an objective function f over a normalized interval $[0, 1]$ two internal points $0 < a < b < 1$ are chosen for fitting a cubic polynomial L,

$$L(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$$

The corresponding four function evaluations are formed into the vector **f** where

$$\mathbf{f} = (f_0, f_a, f_b, f_1)^T$$

Evaluating $L(x)$ at points $\{0, a, b, 1\}$ and requiring $L(x)$ to equal $f(x)$ at those points yields the following linear relationship,

$$\begin{pmatrix} f_0 \\ f_a \\ f_b \\ f_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & a & a^2 & a^3 \\ 1 & b & b^2 & b^3 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$$

The polynomial coefficients are found by inverting the matrix in the above equation,

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \frac{1}{D} \begin{pmatrix} D & 0 & 0 & 0 \\ -(D+d) & b & -a & D \\ 2d & -(b+1) & a+1 & -d \\ -d & 1 & -1 & d \end{pmatrix} \begin{pmatrix} f_0 \\ f_a \\ f_b \\ f_1 \end{pmatrix}$$

where $d = b - a,\ D = abd$ and it is assumed for symmetry that $b = 1 - a$.

Figure A.1 depicts an example interpolation with many of the elements discussed in this section present.

As may be found in any elementary calculus text the error of an n^{th}-order polynomial interpolation of $f(x)$ is given by

$$L(x) - f(x) = -\frac{f^{(n+1)}(c)}{(n+1)!}\prod_{i=0}^{n}(x - x_i) \tag{A.1}$$

$$= \frac{f^{(4)}(c)}{4!}\, x(1-x)(x^2 - x + a - a^2) \tag{A.2}$$

where $n = 3$, $x_i \in \{0, a, 1-a, 1\}$ and for some $c \in (0,1)$. The largest errors occur at the extrema of the fourth-order polynomial $E(x) = x(1-x)(x^2 - x + a - a^2)$ in the error expression. The extrema points are called *ridge-error* points. Solving $E(x)$ for the ridge-error points,

$$\frac{d}{dx}E(x) = 0 \implies x = \left\{\frac{1}{2}\left(1 \pm \sqrt{2a^2 - 2a + 1}\right), \frac{1}{2}\right\}$$

Evaluating $E(x)$ at each ridge-error point yields the unique extrema,

$$\left\{\frac{1}{4}a^2(1-a)^2, -\left(\frac{1-2a}{4}\right)^2\right\}$$

Equating the absolute value of the two ridge-point extrema yields the following candidates for a,

$$2a(1-a) \equiv 1 - 2a \implies a = 1 \pm \frac{1}{\sqrt{2}}$$

Since a is required to lie in the interval $(0, \frac{1}{2})$ so that $0 < a < b < 1$, the ridge-point values are equated in absolute value by the choice

$$a = 1 - \frac{1}{\sqrt{2}} \approx 0.29289322 \tag{A.3}$$

Given this choice of a the ridge-error points are,

$$l = \frac{1}{2}\left(1 - \sqrt{2a^2 - 2a + 1}\right) = \frac{1}{2}\left(1 - \sqrt{2 - \sqrt{2}}\right) \approx 0.117317$$

$$m = \frac{1}{2}$$

$$r = \frac{1}{2}\left(1 + \sqrt{2a^2 - 2a + 1}\right) = \frac{1}{2}\left(1 + \sqrt{2 - \sqrt{2}}\right) \approx 0.882683$$

and the $L(x)$ coefficients are found to be,

$$\begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -(3+2\sqrt{2}) & 4+3\sqrt{2} & -(2+\sqrt{2}) & 1 \\ 4+4\sqrt{2} & -(10+7\sqrt{2}) & 8+5\sqrt{2} & -(2+2\sqrt{2}) \\ -(2+2\sqrt{2}) & 6+4\sqrt{2} & -(6+4\sqrt{2}) & 2+2\sqrt{2} \end{pmatrix} \begin{pmatrix} f_0\\ f_a\\ f_b\\ f_1\end{pmatrix} \tag{A.4}$$

Let A_{opt} be the matrix in the above equation. Noticing that

$$L(x) = (1, x, x^2, x^3) \begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix} = (1, x, x^2, x^3)\; A_{opt}\; \mathbf{f}$$

it is possible to precompute *ridge-error vectors*, v_l, v_m, v_r corresponding to the ridge-error values l, m, r found above for the sake of computational efficiency. Let

$$\mathbf{v}_k = (1, x_k, x_k^2, x_k^3)\; A_{opt} \qquad \text{for} \qquad k \in \{l, m, r\}$$

The ridge-error vectors are then,

$$\mathbf{v}_l = \frac{1}{4}\left(1+\sqrt{2-\sqrt{2}}, 1+\sqrt{2+\sqrt{2}}, 1-\sqrt{2+\sqrt{2}}, 1-\sqrt{2-\sqrt{2}}\right)^T \tag{A.5}$$

$$\mathbf{v}_m = \frac{1}{4}\left(1-\sqrt{2}, 1+\sqrt{2}, 1+\sqrt{2}, 1-\sqrt{2}\right)^T \tag{A.6}$$

$$\mathbf{v}_r = \frac{1}{4}\left(1-\sqrt{2-\sqrt{2}}, 1-\sqrt{2+\sqrt{2}}, 1+\sqrt{2+\sqrt{2}}, 1+\sqrt{2-\sqrt{2}}\right)^T \tag{A.7}$$

The ridge-error points and vectors are then used to compute the following set of interpolation errors,

$$\epsilon_l = |f(l) - \mathbf{v}_l^T \mathbf{f}\,| \tag{A.8}$$

$$\epsilon_m = |f(m) - \mathbf{v}_m^T \mathbf{f}\,| \tag{A.9}$$

$$\epsilon_r = |f(r) - \mathbf{v}_r^T \mathbf{f}\,| \tag{A.10}$$

The *EveryRoot* method uses the above interpolation errors to determine if $L(x)$ is a reliable facsimile to $f(x)$ over the normalized interval $[0, 1]$. If $L(x)$ is deemed reliable the *EveryRoot* method requires the extrema points of $L(x)$. These are found by solving the quadratic $L'(x) = 0$ and accepting only real roots, that is,

$$L'(x) = 3\alpha_3\, x^2 + 2\alpha_2\, x + \alpha_1 = 0$$

$$\rho = -\frac{\alpha_2}{3\alpha_3} \pm \sqrt{\left(\frac{\alpha_2}{3\alpha_3}\right)^2 - \frac{\alpha_1}{3\alpha_3}}$$

The curvature of $L(x)$ is given by its second derivative,

$$L''(x) = 6\alpha_3\, x + 2\alpha_2$$

When $L''(x)$ is evaluated at the extrema, if any, then the *EveryRoot* method can decide if the extrema is near zero and warrents further investigation. Notice, for example, that in figure A.1 the objective function $f(x)$ crosses zero even though $L(x)$ does not. The example in the figure has grossly exaggerated error values for $\{\epsilon_l, \epsilon_m, \epsilon_r\}$ and $L(x)$ would certainly be rejected by the *EveryRoot* method as un-reliable. An interesting modification to the *EveryRoot* method presents itself in the case of a reliability rejection; subdivide the interval into three parts, not two, namely $\{(0, a), (a, b), (b, 1)\}$. The rationale for this modification is that each of the endpoints has already been evaluated so that $\{f_0, f_a, f_b, f_1\}$ are already in hand.

Notice further from the example depicted in figure A.1 that because $L(x)$ is concave at the upper root, ρ^+, it seems a good place to initialize a root polishing method. In the figure it appears that a root is already found. This is accidental, but makes the point that even a reliable $L(x)$ should not be taken as a duplicate of the objective $f(x)$.

A rule to consider as a modification to the *EveryRoot* method is if

$$|L(\rho)| < max\{\epsilon_l, \epsilon_m, \epsilon_r\}$$

then root polishing should be initiated. A root polisher such as Halley's method, which approximates $f(x)$ with a parabola may best be initiated at the relevant ρ and to bound it by the nearest known neighbors. In the figure the relevant extrema is ρ^+ and the bounding interval is (b, r).

Building the Interpolation Subroutine

The *EveryRoot* algorithm enters the Lagrange interpolation subroutine when $f_0 f_1 > 0$, that is, the end points are on the same side of zero. The evaluations of f_0 and

f_1 are assumed. The Lagrange interpolation subroutine must evaluate $f(x_a) = f_a$ and $f(x_b) = f_b$ to create the α polynomial coefficients. Suppose f_a is such that $f_0 f_a < 0$, that is, on the other side of zero, then the two subintervals described by the partition $[0, a, 1]$, are sent back to the *EveryRoot* method where each is marked as known to contain at least one root.

Continuing with the interpolation subroutine, three more function evaluations at the ridge-error points must be made, namely, $\{f_l, f_m, f_r\}$. The total number of new function evaluations so far is five. If any of the ridge-error evaluations result in a zero crossing the appropriate partition of the investigation interval is sent back to the *EveryRoot* method and each maked as known to contain at least one root. Suppose, for example, only f_l crosses zero such that $f_0 f_l < 0$ then the sub-partitition $[0, l, a]$ is sent back with each containing a root and the remaining $[a, 1]$ is resent to the interpolation subroutine as an investigation interval since $f_a f_1 > 0$.

At this point it is decided whether the interpolation is reliable. If not, there are five new interval endpoints that cannot be recycled so these become natural investigation subintervals, that is, the initial (normalized) interval is partitioned into six subintervals, $[0, l, a, m, b, r, 1]$ and each is re-fed in turn into the interpolation subroutine.

If the interpolation is deemed reliable then extrema are calculated. Either the extrema are real or not and if real inside the interval or not. If there is no extrema within the interval of positive curvature when $f_0 > 0$ and negative if $f_0 < 0$, in other words, close to zero, then the entire interval $[0, 1]$ is marked as containing no roots and the subroutine ends.

If there is an extrema ρ of appropriate curvature then a final function evaluation is made at $f_\rho = f(\rho)$. If $f_0 f_\rho < 0$ then the partition $[0, \rho, 1]$ is sent back each known to contain a root otherwise the two nearest neighbor points are chosen and the roots of the derivative of the objective f are found. For example in figure A.1, ρ^+ is the suggested minima. The nearest neighbors are $\{b, r\}$ and the true minima of f could be left or right of ρ^+.

Suppose the interval $[b, r]$ is suspected of containing the minima. If the endpoints are such that $f'(b) < 0$ and $f'(r) > 0$ then a root bounding method applied to f' over $[b, r]$ is used to find $\hat{\rho}$, the true minima of f. Otherwise the interpolation subroutine is re-entered over interval $[b, r]$ and the flanking intervals $\{[0, b], [r, 1]\}$ are marked as not containing roots. If $\hat{\rho}$ exists and $f_0 f(\hat{r}) > 0$ the interval is declared to no contain a root else the intervals are $\{[b, \hat{\rho}], [\hat{\rho}, r]\}$ each sent to the bounded root method since each contains a root. The possibility that $f(\hat{\rho}) \approx 0$ is considered and if true a single root is returned at $\hat{\rho}$.

Appendix B

Tables and Chairs, Sharply

B.1 An Exemplary 2D Sharp Example

The example presented here is small enough to describe in full detail and incorporate many of the key features of algebraically correlated random variables described in this work. Presented in two stages, first with sharp inputs replicating an example from Gass [13] and the second with algebraically correlated random variable inputs. It will then describe how the example may be generalized.

The basis for the tables and chairs example can be found in Gass [13] wherein a decision must be made by a small furniture manufacturer under resource constraints. The choice is whether to manufacture tables or chair or some combination of both. The goal is to maximize the revenue from the sale of the tables and chairs assuming that all will be sold. The specifics are,

1. There is 400 board-feet of wood available.
2. There is 450 man-hours of labor available.
3. It takes 5 board-feet of wood and 10 man-hours of labor to make a chair.
4. It takes 20 board-feet of wood and 15 man-hours of labor to make a table.
5. Chairs sell for \$45 each.
6. Tables sell for \$80 each.

Stating the problem in standard form according to Boyd [3] and Greenberg [14],

$$\begin{array}{lr} \text{maximize} & 45x_c + 80x_t \\ \text{s.t.} & 5x_c + 20x_t \leq 400 \\ & 10x_c + 15x_t \leq 450 \\ & x_c, x_t \geq 0 \end{array}$$

where x_c represents the number of chairs to manufacture and sell and x_t represents the number of tables to manufacture and sell.

To form a baseline, first solve this optimization problem using the simplex method as described in the simplex method section. The problem will then be solved again with the prices kept unknown. Random pricing may now be introduced into the problem.

B.1.1 Tables and Chairs with All Inputs Sharp and Known

The simplex method requires all constraints to be stated as equalities so a slack variable introduced into each inequality. The problem is restated as,

$$\begin{array}{lr} \text{maximize} & 45x_c + 80x_t \\ \text{s.t.} & 5x_c + 20x_t + s_W = 400 \\ & 10x_c + 15x_t + s_L \leq 450 \\ & x_c, x_t, s_W, s_L \geq 0 \end{array}$$

where s_W is the slack variable for the wood resource equation and s_L is the slack variable for the labor resource equation. All variables, x_c, x_t, s_W and s_L are constrained to be non-negative.

Since each slack variable appears exclusively once in the constraint equations and their coefficients are $+1$ they collectively form a basis for the simplex tableau in table B.1

The problem variables are collected into the list $X = (x_c, x_t, s_W, s_L)$. Using the order of this list, denote the variables in the current basis with a 1 and the others with a 0. The current simplex state is then described with the binary value,

$$State_0 = 0011$$

	x_c	x_t	s_W	s_L	b
s_W	5	20	1	0	400
s_L	10	15	0	1	450
Revenue	45	80	0	0	

Table B.1: Tables and Chairs Simplex Tableau for State 0011

To pivot the table, find the variable to enter the basis and the basis variable to exit the basis. To find the entering variable, compute the cost impact of each,

$$\begin{aligned} Z_c &= 45 - (5*0 + 10*0) && = 45 \\ Z_t &= 80 - (20*0 + 15*0) && = 80 \end{aligned}$$

where Z_c is the cost impact of introducing variable x_c into the basis and Z_t is the cost impact for x_t. Recall that increasing x_c the assumed zero value for a non-basis variable by one unit (one more chair sold, for example) will increase revenue by the price of one chair, $\$45$, and will decrease the slack variables s_W and s_L by 5 and 10 units respectively. Since there is no revenue impact to increasing or decreasing slack variables the revenue impact is zero for each.

The entering variable is selected as,

$$\text{argmax}\{Z_c, Z_t\} \implies x_t$$

The next simplex state has the form $10??$ because x_c is the entering variable. The question marks indicate that the exiting variable is not yet known.

Since x_t is the entering variable, divide $b = (400, 450)$ element-wise by the basis coefficients for x_t namely 20 and 15 and find the minimum non-negative value. In particular one finds,

$$\text{argmin}\{\frac{400}{20}, \frac{450}{15}\} \implies s_W$$

Since $\frac{400}{20} < \frac{450}{15}$ and the former value is associated with basis variable s_W it is chosen as the exit variable. Recall that the reason is that the entering variable x_t is allowed to increase from zero and this forces the basis variable in each equation toward zero. In the first equation a unit increase in x_t is a 20 unit decrease in $s_W = 400$, but only a 15 unit decrease in $s_L = 450$. Since no variable is allowed

	x_c	x_t	s_W	s_L	b
x_t	$\frac{1}{4}$	1	$\frac{1}{20}$	0	20
s_L	$6\frac{1}{4}$	0	-$\frac{3}{4}$	1	150
Revenue	45	80	0	0	

Table B.2: Tables and Chairs Simplex Tableau for State 0101

to be negative at least one basis variable is driven to zero by an increase in the entering variable. The new simplex state is,

$$State_1 = 0101$$

To transform the equations and update the tableau, form the transformation matrix to state 1, B_1 and its inverse as,

$$B_1 = \begin{pmatrix} 20 & 0 \\ 15 & 1 \end{pmatrix} \qquad B_1^{-1} = \frac{1}{20}\begin{pmatrix} 1 & 0 \\ -15 & 20 \end{pmatrix}$$

recognizing each tableau column as a vector and multiplying on the left by B_1^{-1} creates the new tableau in table B.2.

The two non-basis variables are now x_c and s_W. The cost impact for introducing each into the basis,

$$Z_c = 45 - (\frac{1}{4} * 80 + 6\frac{1}{4} * 0) \qquad = 25$$
$$Z_w = 0 - (\frac{1}{20} * 80 - \frac{3}{4} * 0) \qquad = -4$$

where Z_w is the cost impact of (re)-introducing s_W into the basis. Since Z_w is negative, s_W it is not eligible to be a basis vector leaving x_c as the only available choice for entering variable.

Dividing b by the vector of coefficients associated with x_c and finding the smallest non-negative value yields,

$$\text{argmin}\{20 \div \frac{1}{4}, 150 \div 6\frac{1}{4}\} = \text{argmin}\{80, 24\} \implies s_L$$

demonstrating the s_L is the exiting variable. The B_2 basis transformation matrix and its inverse become,

	x_c	x_t	s_W	s_L	b
x_c	1	0	-0.12	0.16	24
x_t	0	1	0.08	-0.04	14
Revenue	45	80	0	0	

Table B.3: Tables and Chairs Simplex Tableau for State 1100

$$B_2 = \begin{pmatrix} \frac{1}{4} & 1 \\ 6\frac{1}{4} & 0 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} 0 & 0.16 \\ 1 & -0.04 \end{pmatrix}$$

The simplex state is then,

$$State_2 = 1100$$

and the tableau for state 1100 is shown in table B.3.

The two slack variables are no longer in the basis so they are both zero. This means that at the current state all available resources are exhausted to manufacture the tables and chairs. To determine if either of the two slack variables should be re-introduced into the basis, the revenoue impact is calculated for each,

$$\begin{aligned} Z_W &= 0 - (-0.12 * 45 + 0.08 * 80) &&= -1 \\ Z_L &= 0 - (0.16 * 45 - 0.04 * 80) &&= -4 \end{aligned}$$

Since each cost impact is negative there is no possible way to improve the revenue of the problem and the algorithm terminates with the results,

$$\begin{aligned} x_c &= 24 \\ x_t &= 14 \\ revenue &= 24 * 45 + 14 * 80 = \$2,200 \end{aligned}$$

since $(x_c, x_t) = b$. This means that the optimal choice for the manufacturer is to make 24 chairs and 14 tables which, when sold, will generate a revenue of $\$2,200$.

Figure B.1 shows the resource constraints (diagonal lines), the feasible region (shaded area) and the optimal point, $(24, 14)$. The simplex method starts at the

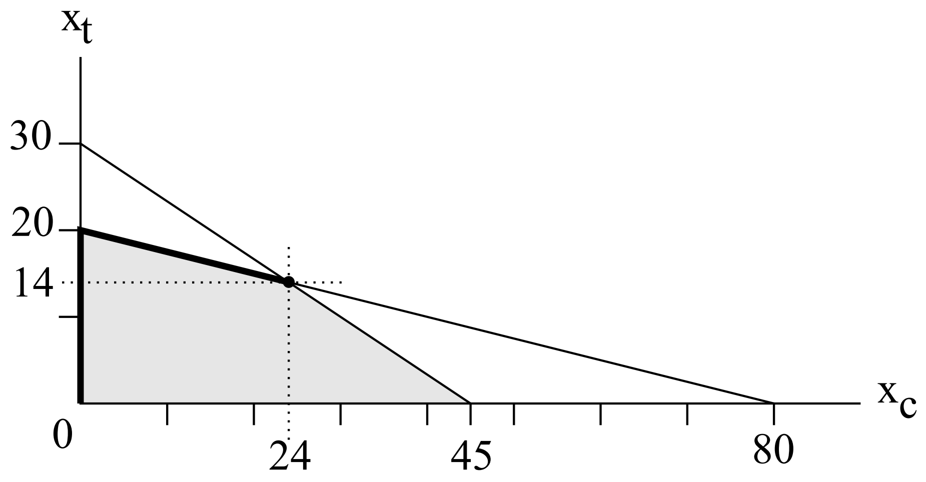

Figure B.1: Tables and Chairs Constraints and Optimal Point

origin in this case and follows the heavy line in figure B.1 from vertex to vertex of the polytope described by the half-space constraints to the optimal vertex. Notice that in this case there is an alternate vertex-path from the origin to the optimal vertex, namely passing through point $(45, 0)$. The choice of vertex-path is significant as will be seen in the next version of this example when the prices are left unknown.

B.1.2 Tables and Chairs with Unknown Prices

As an intermediate step, consider where uncertainty may be injected into the tables and chairs example. Leaving the prices sharp, but unknown leads to some revealing results.

A priori there are three places in the tables and chairs example where uncertainty may be injected; the constraint vector b, the price vector p and the constraint matrix A where,

$$A = \begin{pmatrix} 5 & 20 \\ 10 & 15 \end{pmatrix}$$

$$b = \begin{pmatrix} 400 \\ 450 \end{pmatrix}$$

$$p = \begin{pmatrix} 45 & 80 \end{pmatrix}$$

Recognize the values in the A matrix as the amount of resources of each type consumed to manufacture each kind of product, b is the number of resources of each kind available and p is the prices charged for each product.

Suppose that instead of the 5 in matrix A, a random variable is introduced. In the context of the tables and chairs example this means that the manufacturer is uncertain about the amount of wood necessary to construct a chair. If there are different kinds of products each would be given separate variables and similarly for each value in the A matrix.

The simplex method uses a *pivot-table* approach whereby a column and related row within the simplex tableau are chosen and a transition is made to a new state within the algorithm.

The choice of simplex tableau column to replace is made by comparing revenue impacts given a choice of one of the non-basis variables. The values involved in computing the revenue impact of each non-basis variable are prices and products of prices and A matrix coefficients from columns corresponding to the non-basis variables. Assuming that the A matrix values are fixed, linear combinations of p vector prices may be compared to find the non-basis variable corresponding to the non-negative maximum of revenue impact values.

The choice of simplex tableau row given a column choice is made by finding the quotient of b and the values in A corresponding to that column. Linear combinations of b vector values are compared to find the basis variable corresponding to the the non-negative minimum of linear combinations of constraint values.

Notice that p and b values do not interact directly within the simplex method. Choose the p vector for introduction of random variables suggesting, in the tables and chairs example, price uncertainty over the b vector values which would suggest resource uncertainty. No new insight is gained though choosing b over p or through choosing both for random variable introduction.

In this intermediate tables and chairs example prices p_c and p_t for chairs and tables are unknown, but sharp. Assume the proces are positive. The problem may then be stated in standard form as,

$$\begin{aligned} \text{maximize} \quad & p_c * x_c + p_t * x_t \\ \text{s.t.} \quad & 5x_c + 20x_t \leq 450 \\ & 10x_c + 15x_t \leq 450 \\ & x_c, x_t \geq 0 \\ & p_c, p_t > 0 \end{aligned}$$

0011	x_c	x_t	s_W	s_L	b
s_W	5	20	1	0	400
s_L	10	15	0	1	450
Revenue	p_c	p_t	0	0	

Table B.4: Tables and Chairs Simplex Tableau for State 0011 with Unknown Prices

The initial simplex tableau is shown in table B.4. The only differences from the initial tableau of the first example is the introduction of the state value 0011 into the upper-left corner and the unknown prices p_c and p_t.

Following the steps from the first example, find the entering non-basis variable by finding,

$$\begin{aligned}&\text{argmax}\{p_c - (5 * 0 + 10 * 0), p_t - (20 * 0 + 15 * 0)\}\\ =&\text{argmax}\{p_c, p_t\}\end{aligned}$$

Since $p_c, p_t > 0$ neither case may be disqualified there are several possibilities. Either $p_c < p_t$ or $p_c > p_t$ or $p_c = p_t$. Anticipating the introduction, in the next example, of continuous random variables in place of p_c and p_t, equality occurs with probability zero so that case is ignored.

If $p_c > p_t$ then choose x_c as the entering variable. To find the exiting variable compute,

$$\begin{aligned}&\text{argmin}\{\frac{400}{5}, \frac{450}{10}\}\\ =&\text{argmin}\{80, 45\} \implies s_L\end{aligned}$$

Since x_c is chosen as the entering variable and s_L as the exiting variable the new state is 1010 and the transition matrix B_{1010} and its inverse B^{-1}_{1010} is,

$$B_{1010} = \begin{pmatrix} 5 & 1 \\ 10 & 0 \end{pmatrix} \qquad B^{-1}_{1010} = \frac{1}{10}\begin{pmatrix} 0 & 1 \\ 10 & -5 \end{pmatrix}$$

The new 1010 tableau is shown in table B.5.

The non-basis variables are x_t and s_L. Compute the revenue of (re)-introducing each of them, respectively, and find the entering variable,

1010	x_c	x_t	s_W	s_L	b
x_c	1	1.5	0	0.1	45
s_W	0	12.5	1	-0.5	175
Revenue	p_c	p_t	0	0	

Table B.5: Tables and Chairs Simplex Tableau for State 1010 with Unknown Prices

$$\text{argmax}\{p_t - (1.5 * p_c + 12.5 * 0), 0 - (0.1 * p_c - 0.5 * 0)\} = \text{argmax}\{p_t - 1.5 * p_c, -0.1 * p_c\}$$

Since $p_c > 0$ then $-0.1 * p_c < 0$ so it must be disqualified as an option. The question arises; under what condition is the first options positive? That is,

$$\begin{aligned} p_t - 1.5 * p_c &> 0 \\ p_t &> 1.5 * p_c \\ \frac{2}{3}p_t &> p_c \end{aligned}$$

Since it is assumed that $p_c > p_t$ it is not possible for $\frac{2}{3}p_t > p_c$. The simplex algorithm terminates at this point. Since non-basis variables must be zero,

$$\begin{aligned} x_c &= 45 \\ x_t &= 0 \\ revenue &= 45p_c && \text{for } p_t < p_c \end{aligned}$$

Returning to the first decision point, assume $p_c < p_t$ as was the case in the first example. The tableau under the assumption of transition from state 0011 to state 0101 is shown in table B.6. Notice this is similar to table B.2 except for the unknown prices and the state being recorded in the upper left corner of the tableau.

From state 0101 the non-basis variables are x_c and s_W. Compute the revenue maximizing variables by the usual methods,

$$\begin{aligned} &\text{argmax}\{p_c - (\frac{1}{4}p_t + 6\frac{1}{4} * 0), 0 - (\frac{1}{20}p_t - \frac{3}{4} * 0)\} \\ =&\text{argmax}\{p_c - \frac{1}{4}p_t, -\frac{1}{20}p_t\} \end{aligned}$$

0101	x_c	x_t	s_W	s_L	b
x_t	$\frac{1}{4}$	1	$\frac{1}{20}$	0	20
s_L	$6\frac{1}{4}$	0	$-\frac{3}{4}$	1	150
Revenue	p_c	p_t	0	0	

Table B.6: Tables and Chairs Simplex Tableau for State 0101 with Unknown Prices

1100	x_c	x_t	s_W	s_L	b
x_c	1	0	-0.12	0.16	24
x_t	0	1	0.08	-0.04	14
Revenue	p_c	p_t	0	0	

Table B.7: Tables and Chairs Simplex Tableau for State 1100 with Unknown Prices

Since $p_t > 0$ the second option of $-\frac{1}{20} * p_t < 0$ and is disqualified. If $p_c < \frac{1}{4} * p_t$ then the simplex algorithm terminates. The results of this termination are,

$$\begin{aligned} x_c &= 0 \\ x_t &= 20 \\ revenue &= 20p_t && \text{for } p_c < \frac{1}{4}p_t \end{aligned}$$

For the simplex algorithm to not terminate at this part requires that $p_c > \frac{1}{4}p_t$. Recall that $p_c < p_t$ is assumed at this point. Since these two assumptions are compatible (i.e. not impossible), continue the simplex algorithm. From the calculation for entering variable, notice that x_c is the entering variable and the exiting variable must be found. Recall this situation in the first example and that the result that s_L is the exiting variable and that the resulting tableau is shown in table B.7. This new figure is nearly identical to the previous table B.3.

Recall that the first example terminated the simplex algorithm upon reaching state (1100). Find the entering variable with the calculation,

$$\begin{aligned} &\text{argmax}\{0 - (-0.12p_c + 0.08p_t), 0 - (0.16p_c - 0.04p_t)\} \\ =&\text{argmax}\{0.12p_c - 0.08p_t, 0.04p_t - 0.16p_c\} \\ =&\text{argmax}\{3p_c - 2p_t, p_t - 4p_c\} \end{aligned}$$

For the second option to be positive and therefore available for consideration requires that $p_t > 4p_c$ which is to say $p_c < \frac{1}{4}p_t$. However, to reach this state is was assumed that $p_c > \frac{1}{4}p_t$ so the second option is not available.

If $p_c < \frac{2}{3}p_t$ then the simplex algorithm terminates just as it did in the first example since $45 < \frac{2}{3} * 80 = 53.33...$. The result in this case is,

$$\begin{aligned} x_c &= 14 \\ x_t &= 24 \\ revenue &= 14p_c + 24p_t \qquad\qquad \text{for } \frac{1}{4}p_t < p_c < \frac{2}{3}p_t \end{aligned}$$

If $\frac{2}{3}p_t < p_c$ then the first option for the entering variable is available and the entering variable is found to be s_W.

The exiting variable in this case is then found as,

$$\text{argmin}\{24 \div -0.12, 14 \div 0.08\}$$

Since the first option is negative it is disqualified leaving the second option and therefore the second of the two basis variables x_t as the exiting variable.

Pivoting on the s_W column and the x_t row results in the transition matrix B_{1010} and its inverse B_{1010}^{-1} as,

$$B_{1010} = \begin{pmatrix} 1 & -0.12 \\ 0 & 0.08 \end{pmatrix} \qquad\qquad B_{1010}^{-1} = \begin{pmatrix} 1 & 1.5 \\ 0 & 125.5 \end{pmatrix}$$

Applying the transition matrix B_{1010}^{-1} to the 1100 state tableau return us to the 1010 state tableau shown in figure B.5. State 1010 terminates so that,

$$\begin{aligned} x_c &= 45 \\ x_t &= 0 \\ revenue &= 45p_c \qquad\qquad \text{for } \frac{2}{3}p_t < p_c < p_t \end{aligned}$$

Since different paths where followed to arrive at state 1010, the conditions for reaching this state are different. Notice that the conditions for reaching this state

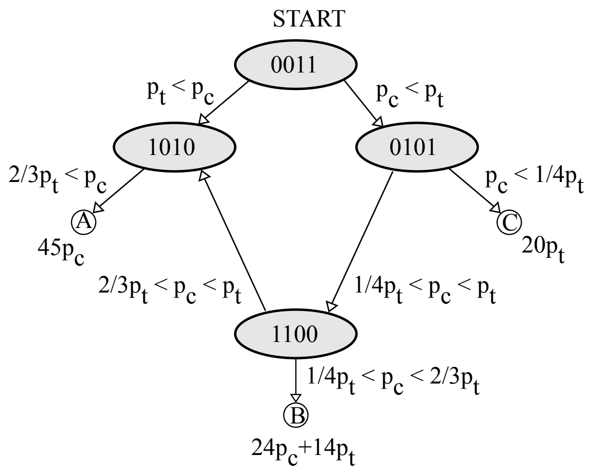

Figure B.2: Tables and Chairs Directed Graph

directly from the initial 0011 state are $p_t < p_c$ do not intersect the conditions for reaching this state from state 1010. Combine the two conditions to see that the revenue outcome of $45p_c$ is reached if $\frac{2}{3}p_t < p_c$.

The result of the investigation is that there are three possible cases for any pair of positive prices given that all other values in the tables and chairs example remain the same, that is,

$$\text{Revenue } = \begin{cases} 45p_c & \text{if } \frac{2}{3}p_t < p_c \\ 24p_c + 14p_t & \text{if } \frac{1}{4}p_t < p_c < \frac{2}{3}p_t \\ 20p_t & \text{if } p_c < \frac{1}{4}p_t \end{cases}$$

The results of this example are summarized by the directed graph in figure B.2. The nodes of the graph are the states of the simplex algorithm allied to this example, the edges are marked with the conditions under which the simplex algorithm will follow the edge and the three output cases are labeled A, B and C accompanied by the resulting revenue.

Suppose random variables P_t and P_c representing the price of tables and chairs respectively are given. Even if P_t and P_c are correlated in some manner they have a joint distribution which is represetned by the shaded region in figure B.3. The figure (B.3) has three levels of shading with each region labeled with its outcome (A, B or C) corresponding to the directed graph in figure B.2. Note that if the two price random variables are independent then the shaded region representing

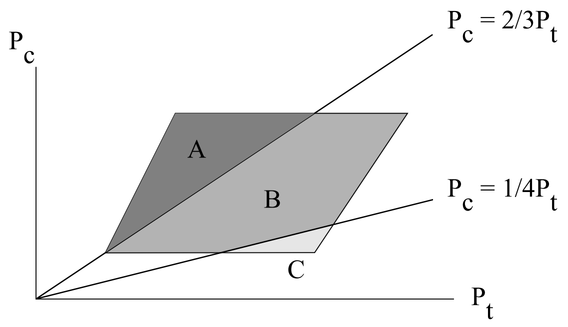

Figure B.3: Tables and Chairs Partitioned Price Probability Space

the joint probability distribution of P_t and P_c would be rectangular.

Appendix C

Symbolic Expression Transformation

C.1 Simplify

A Rico object may represent a function whose operands are themselves Rico objects. A Rico Object then forms a *parse tree*. The reference to *parsing* stems from a tree of Rico objects being created in response to user interaction or user-provided algorithm. The possibility of algorithmic creation of parse trees drives the focus to expression simplification since algorithms may generate parse trees that involves tens or even tens of millions of terms.

As detailed elsewhere not all forms of algebraically equivalent expressions are equally easy to evaluate numerically. As an example consider $X + XY$ where X and Y are mutually independent random variables. The addition operation between X and XY is not simple since the operands are algebraically correlated. However, the equivalent expression, $X(1 + Y)$ involves the addition and multiplication of mutually independent random variables. Independent combinations of random variables tend to be easier to implement numerically than correlated combinations since there is no correlation component.

Another reason for expending the design cost of creating a function that simplifies Rico parse trees is to reduce the evaluation work load. For example, an expression such as $(X + X^2)/X$ is better represented as the algebraically equivalent $1 + X$. Similarly $log(exp(X))$ is better represented as X. While a user may not type such an expression as $log(exp(X))$, an algorithm may generate it or it may arise as an intermediate step in the simplification process itself.

Rather than attempting to define a *simplification* metric for all admissible expressions, the goal of this section is to demonstrate that the process called *simplification* within the Rico project reliably produces algebraically equivalent output expressions for any admissible input expression and tends to produce simpler equivalent expressions.

The simplification algorithm is detailed below. It employs expression simplification heuristics that will also be detailed. A testing regimine will be detailed and the claim that the testing regimine will guarantee the claim of expression equivanence.

C.1.1 Simplification Overview

When given an arithmetic expression in the form of a Rico object representing an expression parse tree the simplification algorithm will recursively call itself on any operand objects (also called *child* objects) forming a *depth-first* algorighm. This knowledge allows each step in the algorithm to assume that any operand objects are in *simplest form*. The definition of *simplest* in this context is itself recursive. The meaning of *simplest* emerges within the simplification steps.

The simplification algorithm is comprised of several stages, the first is properly called the simplifying step. The purpose of this step is to remove all integer exponents (save 1 and -1) from sums of objects, recover identity operations such as $exp(log(X)) \rightarrow X$ and $X/X \rightarrow 1$, generally convert any expression to the quotient of sums of products to the extend possible.

Under certain circumstances division of polynomials may be attempted in secondary stages of simplification. A conditional last stage of simplification is factoring of common terms.

Another motivating example is,

$$\frac{(x+2)^2+1}{x^2+4x+5} \rightarrow 1$$

The components to the simplification algorithm are described below with the lowest order first.

C.1.2 Rico Objects

Before expression simplification is attempted the admissible expressions must be detailed. For the most part Rico creates it's parse tree as numerical operations are encountered. If, for example, the user directs Rico to form the square root of a Rico object, the result is a square root object with the starting object as the sole operand. The process is similar for log, exp, subtraction, division, etc. The exceptions are addition, multiplication and exponentiation (or power).

There are three basic kinds of Rico objects to consider during simplification; NaN, numeric, and non-numeric where *NaN* stands for not-a-number. A NaN Rico object arises to indicate a result is undefined such as 0/0, etc. A numeric Rico object is one of the allowed numeric types such as integers, floating point values and fractions (of integers). A non-numeric Rico object represents either a built-in random variable or a function of Rico objects whose result is not numeric and well defined (not NaN).

It is convenient to use compact symbols to represent the three kinds of Rico objects,

ϕ is a NaN Rico object
ν is a numeric Rico object
ρ is a non-numeric Rico object

and capital letters such as X, near end of alphabet represent generic Rico objects.

Simplify Negate

The negation simplifier accepts one operand and obeys the following rules,

$$\begin{aligned} -\{\phi\} &\to \phi \\ -\{\nu\} &\to -\nu \\ -\{\nu \times X\} &\to \{-\nu\} \times X \\ -\{X\} &\to \{-1\} \times X \end{aligned}$$

The $-\{\nu \times X\}$ rule allows number objects to absorb the negation operation. The last rule replaces the negation operation with the product of X and -1, in the conventional number-first order.

Rico Addition

The Rico addition object accepts two or more operands. There are four cases for each operand,

$$\{\phi, 0, \nu, \rho\}$$

where it is understood that a rule that applies to zero is selected before a generic number ν and X applies to any Rico object. Since the rules are applied in order the specific ρ symbol tends not to appear. Rather, the generic X (and subsequent generics) appear most often.

Several features are accommodated in the following rules,

1. Propagate NaN values
2. Discard zeros
3. Perform numeric addition
4. Ensure number objects appear in first position in list of operands
5. Perform coefficient addition
6. Collapse cascaded addition operations

These rules may be encoded as if there are only two operands as follows,

$$\begin{aligned}
\phi + X &\rightarrow \phi \\
X + \phi &\rightarrow \phi \\
X + 0 &\rightarrow X \\
0 + X &\rightarrow X \\
\nu_1 + \nu_2 &\rightarrow \{\nu_1 + \nu_2\} \\
X + \nu &\rightarrow n + X \\
\nu_1 \times X + \nu_2 \times X &\rightarrow \{\nu_1 + \nu_2\} \times X \\
X + (Y + Z) &\rightarrow X + Y + Z \\
(X + Y) + Z &\rightarrow X + Y + Z \\
X + Y &\rightarrow X + Y
\end{aligned}$$

When there are more than two operands the behaviour is as if an accumulation of two-operand addition operations, i.e. $(X + Y + Z) = ((X + Y) + Z)$.

Rico Multiplication

Within Rico multiplication is handled in a similar manner to addition. Cascades products are flattened, zeros and NaN's dominate expressions and ones are filtered out. The rules are then,

$$
\begin{aligned}
NaN * Y &\rightarrow NaN \\
X * NaN &\rightarrow NaN \\
X * 0 &\rightarrow 0 \\
0 * Y &\rightarrow 0 \\
X * 1 &\rightarrow X \\
1 * Y &\rightarrow Y \\
(X * Y) * Z &\rightarrow X * Y * Z \\
X * (Y * Z) &\rightarrow X * Y * Z \\
(X * Y) * (Z * K) &\rightarrow X * Y * Z * K
\end{aligned}
$$

As in the addition case the NaN dominates the output of any function. Like the NaN value, zero dominates the multiplication output. Note in particular that NaN dominates zero. The next two rules involving one are the equivalent nullification as the zero is for addition. The last three rules detail the expression flattening of cascaded products.

Rico Power

The Power function, X^Y, requires more care so it doesn't provide any automatic flattening of cascaded powers as addition and multiplication do provide. There are still the automatically applied rules for NaN, 0 and 1,

$$
\begin{aligned}
X^{NaN} &\to NaN\\
NaN^{Y} &\to NaN\\
NaN^{NaN} &\to NaN\\
(X1+\cdots+X3)^{n} &\to X1^{n}+\cdots+X3^{n}\\
(X1\times\cdots\times X3)^{Y} &\to X1^{Y}\times\cdots\times X3^{Y}\\
(X^{Y})^{Z} &\to X^{Y\times Z}\\
exp(X)^{Y} &\to exp(X\times Y)\\
n1^{n2} &\to n3\\
X^{0} &\to 1\\
X^{1} &\to X
\end{aligned}
$$

The $n1^{n2} \to n3$ rule is triggered when both operands are numeric values and the rule says to evaluate these two operands into a single numeric value, possibly Nan. The numeric power rule is detailed below.

The rule $(X1 + X2 + ... + X3)^n \to X1^n + ... + A3^n$ assumes an integer exponent. The distributive law is them applied so that the result is a sum of powers.

The rule $(X1\times\cdots\times X3)^Y \to X1^Y\times\cdots\times X3^Y$ distributes powers over each product operand. This rule is in keeping with the overall strategy of exploring legal interactions bewteen operands. A later factoring step for exponents will recover the original form if possible.

The rule $X^0 \to 1$ presumes that X is non-numeric. This is no guarantee that X does not represent a random variable with non-zero probability concentrated at zero or either infinity. This rule will need to be revisited when discrete random variables are introduced. Doubtless this rule will be removed at that point.

Rico Numeric Power

The special case results of the power evaluation of two numeric values are detailed in table XXX.

x^y	0	1	$+\infty$	$-\infty$	NaN	$d2$
0	NaN	0	0	NaN	NaN	$\{NaN, 0\} \sim \{-, +\}$
1	1	1	NaN	NaN	NaN	1
$+\infty$	NaN	$+\infty$	NaN	NaN	NaN	$\{0, +\infty\} \sim \{-, +\}$
$-\infty$	NaN	NaN	NaN	NaN	NaN	$\{0, -\infty\} \sim \{-, +\}$
NaN	NaN	NaN	NaN	NaN	NaN	NaN
$d1$	1	x	$sgn(d1)\infty$	0	NaN	$d1^{d2}$

The expression $\{0, -\infty\} \sim \{-, +\}$ in the last column means that 0 is chosen if $d2 < 0$ else $-\infty$ is chosen. There is no possibility for $d2$ to be zero since that case is already addressed.

Equality Test

Several parts of the simplification algorithm rely on the ability to compare two objects for equality. Two Rico objects are *equivalent* if they are of the same Rico type and if they have child objects (operands for functional objects) then, modulo order in the case of addition and multiplication, those child objects are also equivalent.

The following Rico objects (expressions) are then equivalent,

$$X + Y + 3 \equiv 3 + X + Y$$
$$X * 5 \equiv 5X$$

but many algebraically equivalent expressions are not equivalent in the Rico sense. For example,

$$(X + 2)^2 + 1 \not\equiv X^2 + 4X + 5$$

Note especially that Basic random variables are a special type of Rico object. Basic random variables are issued a unique id code upon creation and two Basic random variables are distinguished by id code, not random variable type. For example if $X = Normal(2, 3)$ and $Y = Normal(2, 3)$, then $X \equiv Y$ iff $id(X) = id(Y)$.

Rico Numbers are equivalent if they represent identical values. Floating point Numbers are equivalent if they are within a specific numerical tolerance to compensate for minor least significant bit differences. Fractions in Rico are easily compared by comparing their integer numerator and denominator since they are always represented in lowest form with the numerator carrying the signed value.

C.1.3 Simplify Stage One

Stage One of the simplify algorithm is depth-first. At each node Stage One of the simplify algorithm is called and upon return a transformation appropriate to the current object is called. If the current object is a function and therefore has child objects (operands) then they are assumed to have already been Stage One simplified.

The Stage One object simplifications are detailed by object type. Many simplifications build on each other.

Simplify Subtract

The subtration function in Rico accepts exactly two operands. The Stage One simplification is to transform subtration to addition as follows,

$$A - B \rightarrow A + \{-1\} * B$$

The end result is that a Stage One simplified expression contains no subtraction functions.

Simplify Division

Analagously to the subtraction function, the division function is transformed as,

$$A \div B \rightarrow A * B^{-1}$$

Simplify Square Root

Building on the division function transformation there is replacement of the square root function,

$$\sqrt{A} \rightarrow A^{\frac{1}{2}}$$

Simplify Log

The Stage One simplification huristic is to turn products of sums to sums of products. This extends to the following transformation rules,

$$\begin{aligned} log(exp(X)) &\to A \\ log(X^Y) &\to Y * log(X) \\ log(X * Y * Z) &\to log(X) + log(Y) + log(Z) \end{aligned}$$

In this way, multiplication within a function is exposed outside as addition.

Simplify Exp

By similar reasoning to Stage One log simplification the exp rules are,

$$exp(log(X)) \to Aexp(X + Y + Z) \qquad \to exp(X) * exp(Y) * exp(Z)$$

In this way, addition within a function is exposed outside as multiplication.

Simplify Addition

A Rico addition object may contain two or more operand objects. The hallmark transformation at this stage is,

$$3X + X \to 4X$$

Notice that $3X$ is a multiplication object that contains a number object as an operand. It is assumed that the Stage One Multiply simplification has already occurred and that this multiplication object contains zero or one number object operands and furthermore if a number object is present it is the first operand in the list of operands. A further consideration is seen in the following transformation,

$$3XY + X + YX \to 4XY + X$$

The number object (3) in the above expression is commonly referred to as the coefficient of the expression. This coefficient must be split from the three operand multiply object into a number object and a two operand multiply object. The latter is referred to as the *base* object. Unique base objects is must be identified and the coefficients of any duplicated base objects are summed. Two related special cases are addressed as follows,

$$\begin{aligned}2XY + X + -2XY &\to X\\ 2XY + -2XY &\to 0\end{aligned}$$

Simplify Multiplication

Multiplication simplification returns NaN if any openand is of type NaN. The distributive law is then applied, as needed, and the result is a sum of Stage One simplified products.

$$\begin{aligned}X \times NaN &\to NaN\\ NaN \times Y &\to NaN\\ NaN \times NaN &\to NaN\\ (X+Y) \times Z &\to XZ + YZ\\ X \times (Y+Z) &\to XY + XZ\\ X^Y \times X^Z &\to X^{Y+Z}\\ n1 \times n2 &\to n3\\ (X \times Y) \times (Y \times Z) &\to XY^2Z\end{aligned}$$

The last term applies to an operand that is itself a product of an arbitrary number of operands. For example,

$$(X \times 2 \times Y) \times (Y \times Z) \to 2XY^2Z$$

C.1.4 Simplify Stage Two

```
This needs to be fleshed out.  First the StageOne
        Simplify needs to be completed.
```

TO DO

Factoring

The Factor tree operation is depth-first. It acts at addition nodes to factor out common nodes from sums of products of nodes. Two example transformations are,

$$AB + AC + D \rightarrow A(B + C) + D$$
$$5A + 5^2AB + 5^3ABC \rightarrow 5A(1 + 5B(1 + 5C))$$

where the juxtaposition implies multiplication of the separate nodes A, B, C. The first transformation demonstrates that the A node need not be common to all products. The second transformation comes from the concept of Net Present Value.

`Define what happens to fractional powers` TO DO

The expression $AB + AC + BC$ may be factored as $A(B + C) + BC$ or $B(A + C) + AC$ demonstrating that factorization is not uniquely defined. The Factor tree operation is implemented using a *sparse product matrix*.

An example without non-unit exponential is first considered. Given the expression,

$$AB + AC + D$$

A list of uniqe nodes is constructed; $\{A, B, C, D\}$. Using this list to define the columns referred to as a *header*, a sparse matrix of powers is constructed where each row represents a product of header elements and the rows collectively represent the sum of products. The sparse product matrix for this example is then,

$$\begin{pmatrix} A & B & C & D \\ 1 & 1 & & \\ 1 & & 1 & \\ & & & 1 \end{pmatrix}$$

The A is most common element since it appears in more rows than any other header element it is chosen as the factored element. If another header element, say B had been as common as A then the powers of A and B would be summed and the greater chosen as the factored element. If a tie persist then the first of the tied element in the header list is chosen.

To factor out the A, the sparse matrix rows are partitioned into those including the A element and not. Two new sparse product matrices are created. It is convenient that the headers of each are simple copies of the original header $\{A, B, C, D\}$. Since the matrices are sparse there is no computational cost beyond a possible header copy operation. The resulting expression may be written as,

$$\begin{pmatrix} A & B & C & D \\ 1 & 1 & & \\ 1 & & 1 & \end{pmatrix} + \begin{pmatrix} A & B & C & D \\ & & & 1 \end{pmatrix}$$

The A expression is now formally factored out and for a reason that will be made clear below zeros are left in the A column of the first matrix,

$$A \times \begin{pmatrix} A & B & C & D \\ 0 & 1 & & \\ 0 & & 1 & \end{pmatrix} + \begin{pmatrix} A & B & C & D \\ & & & 1 \end{pmatrix}$$

The Factor operation recurses until no further factoring is possible as is the case in the example above for both sparse product matrices. The resulting expression is then,

$$A \times (B + C) + D$$

To introduce further considerations a second example is warrented. Consider the expression,

$$5A + 5^2AB + 5^3ABC$$

The corresponding sparse matrix is,

$$\begin{pmatrix} 5 & A & B & C \\ 1 & 1 & & \\ 2 & 1 & 1 & \\ 3 & 1 & 1 & 1 \end{pmatrix}$$

The 5 and the A are equally common and the tie is broken by the sum of powers which is 6 for the 5 node and only 3 for the A node. The 5 is then factored, but instead of creating two sparse product matrices only one needs be created since the 5 appears in all rows and the expression becomes,

$$5 \times \begin{pmatrix} 5 & A & B & C \\ 0 & 1 & & \\ 1 & 1 & 1 & \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

Since the A is now most common it is factored and only one sparse product matrix results again,

$$5 \times A \times \begin{pmatrix} 5 & A & B & C \\ 0 & 0 & & \\ 1 & 0 & 1 & \\ 2 & 0 & 1 & 1 \end{pmatrix}$$

Now 5 and B are most common, but 5 has more power. The reason for the zero's becomes clear because a one must be left in the place of the first row,

$$5 \times A \times \Big(\begin{pmatrix} 5 & A & B & C \\ 0 & 0 & & \end{pmatrix} + \begin{pmatrix} 5 & A & B & C \\ 1 & 0 & 1 & \\ 2 & 0 & 1 & 1 \end{pmatrix} \Big)$$

The zero row of the first matrix resolves to a one and the 5 of the second matrix is factored,

$$5 \times A \times \Big(1 + 5 \times \begin{pmatrix} 5 & A & B & C \\ 0 & 0 & 1 & \\ 1 & 0 & 1 & 1 \end{pmatrix} \Big)$$

Next the B is factored and the matrices can be replaced by their equivalent expressions with juxtaposition in place of multiplication,

$$5A\Big(1 + 5B(1 + 5C)\Big)$$

Appendix D

Senstivity

D.1 Introduction

According to Saltelli (Saltelli 2004[10]) an important phase in model analysis is sensitivity and uncertainty analysis. We may add *risk analysis* to that list. Our focus model is the carbon abatement market which impacts business finances and uncertainty implies risk of loss of business profits.

The standard approach, according to Saltelli ([10]), is to use *Monte Carlo* analysis. A description of a family of Monte Carlo methods of analysis may be found in Tanner (Tanner 1996 [17]). As may be found in Saltelli or Tanner, the basic idea behind the Monte Carlo method is to assign random variables to model inputs to represent their uncertainty. A value from each input is chosen and the model is run. The result of the model run is stored and the process is repeated.

The techniques involved with model uncertainty and sensitivity analysis using Monte Carlo include, according to Tanner ([17]) and more recently Weare (Weare 2007 [19]), incorporation of *Markov Chains*. It may be that the technique we develop in this work for analyzing sensitivity, uncertainty and risk provides no computational advantage over modern Monte Carlo methods. It is the opinion of the author of this work that by taking a *glass-box* approach where we take the model implementation into account as opposed to the *black-box* approach where only model outputs are captured, contrasting results of model analysis will be obtained.

We begin our uncertainty, sensitivity and risk analysis in the standard, according to Saltelli ([10]), approach of assigning a probability distribution to each model input to represent its uncertainty. We then regard model inputs as random variables and notice that within our reference model implementation that these inputs are combined arithmetically. We understand from elementary texts such as Bickel [2] that functions of random variables, including their arithmetic combinations, result in still more random variables. We then ask ourselves what if we re-constructed our reference model to allow for random variables in place of numerical values and ran the entire model with this substitution. We wonder, would we then have undertaken an uncertainty, sensitivity and risk analysis by other means? This is the central question for the remainder of this work.

In the opinion of the author, an attractive feature that Monte Carlo methods seem to provide is that the model under analysis need not be re-implemented. We wish to preserve this feature of non-re-implementation in our work. As we have seen previously, programmatic objects in the PHoX modeling system are all derived from a common object called Variant. Our goal will be to describe an upgrade of Variant that includes random variables as a possible representation along side real values, lists, maps, matrices, etc. that we have already identified.

To accomplish our goal of substituting random variables for real-valued model inputs we find it necessary to extend the established theory of functions of random variables. Our theory extension will go far enough to include all the functions we find not only in the reference model, but in general *business class* models as far as we can determine. A business class model, as defined herein, uses functions and concepts available in standard business textbooks such as Brealey [4].

In following sections we describe an implementation that numerically computes all the functions of random variables we require. We will find that merely being able to programmatically represent objects such as $f(A)$, $A + B$, $A \div B$, etc. for independent random variables A and B is not sufficient for our purposes. We return to Saltelli ([10]) to find we must extend the theory of random variables and our implementation thereof to account for unavoidable correlations between random variables.

In keeping with our *glass box* approach we notice that it is no longer feasible to prepare our reference model for optimization and then pass this programmatic preparation to an external optimization engine expecting results. We must implement the optimization step within PHoX so that it will allow random variable inputs. The reference model implemented within PHoX uses a linear programming engine which we will implement using the *Simplex Method* as described in Gass (Gass 1975 [13]). We will describe the Simplex Method in detail to determine if any techniques for handling random variables are needed beyond those already described.

A final chapter is devoted to special considerations for random variable pro-

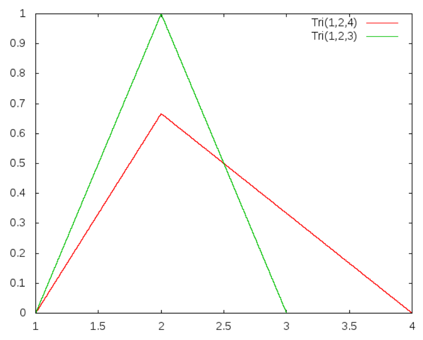

Figure D.1: Probability Distribution of Random Variables X = Tri(1,2,4) and Y = Tri(1,2,3)

cessing particular to the Simplex Method. The structure of the model results following the Simplex Method will be presented.

Our literature search suggests that the approach described herein is novel.

In a monograph by Drew et al. [9] a Maple module called A Probability Programming Language (APPL) is presented. The goal of Drew et al. is to develop data structures and algorithms with which to derive existing and new results in probability and statistics.

The data structures in APPL allow the representation of continuous random variables through piecewise symbolic functions. Using the hosting Maple's computer algebra system algorithms, new random variables are created through high-level statements. Citing an example from the Drew text [9], let

$$\begin{aligned} X &:= TriangularRV(1,2,4) \\ Y &:= TriangularRV(1,2,3) \\ V &:= Product(X,Y) \\ &\; PlotDist(V) \end{aligned}$$

For convenience the triangular distributions are shown in figure D.1.

The resulting probability distribution for the example $V = X \times Y$ above is given in the Drew text [9] by the probability density function $f_V(v)$,

$$f_V(v) = \begin{cases} -\frac{4}{3}v + \frac{2}{3}\log v + \frac{2v}{3}\log v + \frac{4}{3} & 1 < v \le 2 \\ -8 + \frac{14}{3}\log 2 + \frac{7v}{3}\log 2 + \frac{10}{3}v - 4\log v - \frac{5v}{3}\log v & 2 < v \le 3 \\ -4 + \frac{14}{3}\log 2 + \frac{7v}{3}\log 2 + 2v & \\ \quad -2\log v - v\log v - 2\log 3 - \frac{2v}{3}\log 3 & 3 < v \le 4 \\ \frac{44}{3} - 14\log 2 - \frac{7v}{3}\log 2 - \frac{8v}{3} - 2\log 3 & \\ +\frac{22}{3}\log v - \frac{2v}{3}\log 3 + \frac{4v}{3}\log v & 4 < v \le 6 \\ \frac{8}{3} - 8\log 2 - \frac{4v}{3}\log 2 - \frac{2}{3}v & \\ \quad +\frac{4}{3}\log v + \frac{v}{3}\log v + 4\log 3 + \frac{v}{3}\log 3 & 6 < v \le 8 \\ -8 + 8\log 2 + \frac{2v}{3}\log 2 + \frac{2}{3}v & \\ \quad +4\log 3 - 4\log v + \frac{v}{3}\log 3 - \frac{v}{3}\log v & 8 < v < 12 \end{cases}$$

As we'll discuss below, the PHoX modeling system is capable of numerically multiplying piecewise uniform representations of random variables. An overlay of the plot of the function $f_V(v)$ above with the PHoX result of multiplying the triangular X and Y is shown in figure D.2. We note that the two curves are similar enough as to be indistinguishable at the full scale shown.

A software package called RandVar available for the statistics computing environment R that involves computing with random variables is created and maintained by Kohl [15]. The package RandVar is a modest effort to allow functions and arithmetic combinations of random variables of various types. The 24 page manual and 10 page user guide suggest that correlated random variables are beyond the scope of the RandVar work. The focus of the RandVar work, judging by the test results available on the web, are on finding integer powers, encoding random variables into matrices and allowing for their multiplication.

D.2 A Standard Approach

In Saltelli et al. (Saltelli [10]) a pair of simple examples are subjected to an exhaustive treatment of sensitivity, uncertainty and risk analysis. The first example of the pair may be represented by,

$$Y = c_1 * X_1 + c_2 * X_2$$

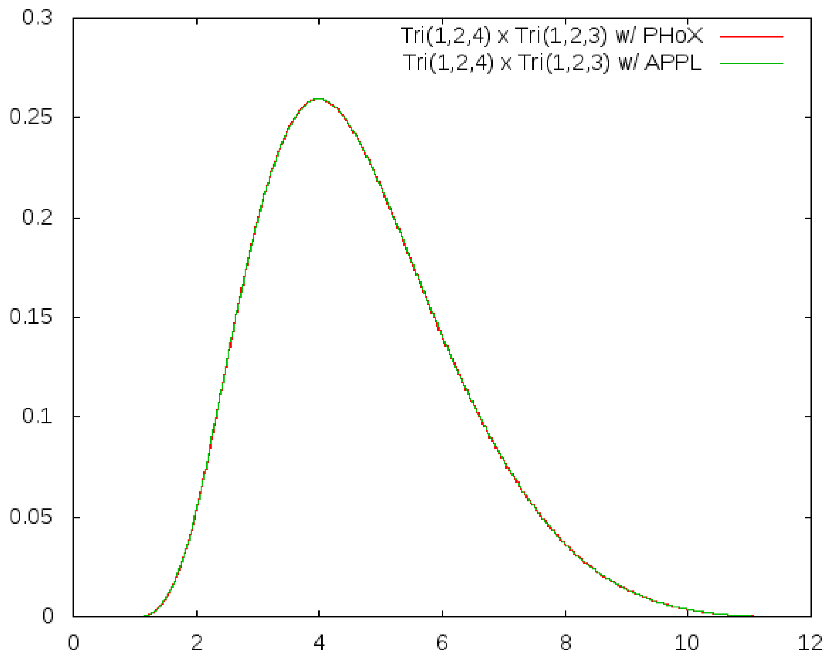

Figure D.2: Probability Distribution of Product of X = Tri(1,2,4) and Y = Tri(1,2,3)

where c_1 and c_2 are constants and X_1 and X_2 are *input factors* that have uncertain value. The risk represented in the Saltelli example is of the return of the linear combination of two hedged portfolios whose returns are represented by X_1 and X_2.

In the second Saltelli example the c_1 and c_2 are admitted as input factors and therefore uncertain. We represent this second example as,

$$Y = C_1 * X_1 + C_2 * X_2$$

where capital case implies uncertain values.

According to Saltelli ([10]) uncertainty analysis assigns probability distributions to each input factor and seeks the probability distribution of each *output factor*. In the the case of the Saltelli example pair there is only one output factor, namely Y.

In the PHoX system the main difference from Saltelli regarding the considered pair of examples is that to each input factor we associate a random variable and proceed to arithmetically combine random variables to produce output factors, in the Saltelli sense. Saltelli suggests that the probability distributions of each input factor be sampled some N number of times and the output factor Y be calculated N times. This is the essence of the Monte Carlo method.

To perform a sensitivity analysis we ask, according to Saltelli, which input factor is most *influential* upon the output factor(s). Saltelli makes the point that merely calculating partial derivatives of Y with respect to each input factor X_i is misleading. In the first example where the coefficients c_1 and c_2 are constants, they are merely returned as in,

$$\partial Y/\partial X_1 = c_1 \qquad\qquad \partial Y/\partial X_2 = c_2$$

If we suppose $c_1 > c_2$, but that the volatility of X_1 is less than that of X_2 we draw the inappropriate conclusion that the volatility of Y (and therefore the riskiness of holding Y as a portfolio) is most influenced by X_1, the low volatility/risk portfolio as opposed to X_2, the high volatility/risk portfolio. Certainly this would be true if $c_2 = 0$, or even if $c_2 << c_1$ depending on the relative volatility of X_2.

The question of which input factor is most influential on a given output factor bears some closer inquiry. Saltelli (Saltelli et al. [11]) offers a possible improvement,

$$S_i = \frac{\sigma_{X_i} \partial Y}{\sigma_Y \partial X_i}$$

where σ_{X_i} is the standard deviation of X_i and similarly σ_Y is the standard deviation of output factor Y. This normalized derivative is, according to Saltelli, recommended for sensitivity analysis by a guideline of the Intergovernmental Panel for Climate Change (IPCC)(1999,2000).

Appendix E

Correlation

E.1 Algebraic Correlation

A model within the PHoX modeling system is composed of several kinds of software objects possibly including arithmetic expressions such as,

$$X = A + \frac{A^2}{B}$$

where A and B are random variables. Established theory dictates that if A and B are independent random variable then we form a two dimensional joint probability distribution for them and perform line integrals along level curves so that,

$$P(X = x) = \int_{x=a+\frac{a^2}{b}} dP(A = a)\; dP(B = b)$$

The problem is that the PHoX modeling system provides a modeling language in which a user can implement a model, it does not know what algebraic expressions will arise within that user-defined model. The PHoX system must handle arbitrary arithmetic expressions involving random variables or at least a well-defined subset.

Since the PHoX system is designed for business-class models rather than scientific modeling some simplifying assumptions will be made. The first such assumption is that arithmetic expressions that arise within a PHoX model are relatively rudimentary, of the kind encountered in a business textbook (Brigham [6]) or introductory finance textbook (Brealey/Myers [4]). To make these assumptions more concrete we refer to the existing Carbon Emission Reduction Market (AB32) model implemented within the PHoX system.

The most complex arithmetic expression in the Carbon Emissions Reduction Market model described elsewhere in this paper is the levelized cost of a carbon emission reduction project. A levelized cost is typically the quotient of two net present value calculation of anticipated cash flows, in the case of the numerator, and the net present value of units of reduction achieved over time, in the case of the denominator. Each net present value calculation is made with some model-defined discount factor.

To introduce uncertainty through random variables into the calculation of a levelized cost we recognize that there is a random component that is common to costs and reductions achieved and a random component that is separate for each. If A, B and C are independent random variables and f, g, h and k are functions we may represent a levelized cost as,

$$Cost = \frac{f(A) + h(B)}{g(A) + k(C)}$$

Our goal is to represent arithmetic combinations of random variables that are, in some sense, at least as complicated as for the case of levelized costs in a computationally tractable manner. To reach this goal we will enumerate a range of cases we need the PHoX system to handle and describe issues involved in implementing appropriate computational algorithms.

E.1.1 Notation and Definitions

Let A, B, etc. are independent random variables and that '$*$' represents any of the binary arithmetic operations $(+, -, \times, \div)$. When two random variables are related by a subscript as in A_1 and A_2 we assume that each is related to the same *parent* random variable A by a rational polynomial function. For example we may have $A_1 = f(A)$ and $A_2 = g(A)$ where f and g are rational polynomial functions of A. We will explain this choice of function class below. We refer to variables related by subscript as *directly related*. We refer to random variables directly related to exactly one first class random variable as *basic*.

E.1.2 Directly Related Random Variables

Claim: The set of rational polynomial functions $\mathbb{Q}$ of a single variables,A, is closed with respect to the binary arithmetic operations $(+, -, \times, \div)$.

Proof of Claim: Let $f, g \in \mathbb{Q}$ where,

$$\begin{aligned} f(A) &= p(A)/q(A) \\ g(A) &= r(A)/s(A) \end{aligned}$$

and $p(A), q(A), r(A), s(A)$ polynomials in A. Allowing that c is an arbitrary constant and that juxtaposition implies multiplication, we need to check each operation,

$$\begin{aligned} f(A) + g(A) &= \frac{p(A)}{q(A)} + \frac{r(A)}{s(A)} && = \frac{p(A)s(A) + r(A)q(A)}{q(A)s(A)} \\ f(A) - g(A) &= \frac{p(A)}{q(A)} - \frac{r(A)}{s(A)} && = \frac{p(A)s(A) - r(A)q(A)}{q(A)s(A)} \\ f(A) \times g(A) &= \frac{p(A)}{q(A)} \times \frac{r(A)}{s(A)} && = \frac{p(A)r(A)}{q(A)s(A)} \\ f(A) \div g(A) &= \frac{p(A)}{q(A)} \div \frac{r(A)}{s(A)} && = \frac{p(A)s(A)}{q(A)r(A)} \end{aligned}$$

Since $p(A) \pm q(A)$ is a polynomial in A and $p(A) \times q(A)$ is also a polynomial in A we reach our desired conclusion.

The reason this restriction of only allowing basic arithmetic operations on an input variable, A, in our model is valid is because the kind of models we that interest us tend to have physical dimensional inputs. That is, the input data to the models we consider represent dimensions such as *meters* or *number of items per year*, etc. The software implementation of models we have considered results in a dichotomy of *foreground* and *background* or behind-the-schemes coding. In background coding any operation or function is allowed in order to perform some specific task such as an optimization or other specialized function. In foreground coding we find that more formal steps must be taken. This tends to disallows the application of functions such as $exp()$ or $log()$ or $sin()$, etc. that require non-dimensional inputs.

We will see that the PHoX modeling system is capable of handling expressions that do not meet our definition of business-class, such as,

$$A + (B \div (A + C))$$

There is no presumption that random variables in the PHoX modeling system have associated physical unit. We seek to balance computational tractability, *representational fidelity* and range of acceptable models. We will next discuss the numerical representation of random variables in the PHoX modeling system. As with any numerical representation or numerical method there remains the question of whether the result accurately represents the theoretical object. This is what we mean by representational fidelity.

E.1.3 Categorizing Algebraically Correlated Cases

To investigate the algebraic expressions we may encounter in a PHoX-based model we adopt the following notation,

$$\text{Let } \mathbb{A} = \{(n_1, n_2, ..., n_m)\}, \forall m, n_1 \leq n_2 \leq .. \leq n_m \in \mathbb{N}$$

where $\mathbb{N} = \{1, 2, ...\}$ and $\mathbb{A}$ represents the set of expressions containing n_1 versions of a random variable A, n_2 versions of another random variable B, etc. where A, B, etc. are mutually independent. The $n_1 + n_2 + ... + n_m = N$ random variables may be combined using the basic arithmetic operator $(+, -, \times, \div)$ in any order. By convention we assume (without loss of generality) that $n_1 \leq n_2 \leq ... \leq n_m$.

We will systematically enumerate and address some elements of $\mathbb{A}$. For example, notice that $\mathbb{A}_{2,1} \in \mathbb{A}$ and that the expression $A_1 + A_2 \times B_1 \subset \mathbb{A}_{2,1}$. Since we may have that $A_1 = 2A$, $A_2 = A^3$ and $B_1 = 1/B$ that the expression $(3A + A^3/B) \in (A_1 + A_2 \times B_1)$ so that $3A + A^3/B \in \mathbb{A}_{2,1}$. Since the cases we will address will not surpass single digit indices we will, in an abuse of notation, equate,

$$\mathbb{A}_{2,1} = \mathbb{A}_{21}$$

The address the issue of computability of elements of $\mathbb{A}$. We shall see below that an essential feature of computability is describing the joint distribution of the random variables in any given expression.

We begin with the case of $\mathbb{A}_1$. Since this represents a single random variable there is nothing to compute. Employing the random variable indices as a way to differentiate two *versions* of a random variable as in $A_1 = f_1(A)$ and $A_2 = f_2(A)$ where f_1 and f_2 are any two rational polynomials of A we see that $\mathbb{A}_1$ contains only the generic case of A_1. The case of $\mathbb{A}_n$ represents the case $A_1 * A_2 * ... * A_n$ which may be denoted by A_{n+1} because it is directly related to A since $f_1(A) * f_2(A) * ... * f_n(A)$ is a rational polynomial in A for any combination of operators $(+, -, \times, \div)$.

The $\mathbb{A}_{11}$ set is enumerated as $\{A_1 + B_1, A_1 - B_1, A_1 \times B_1, A_1 \div B_1\}$. Since A and B are assumed to be mutually independent, A_1 and B_1 are also mutually independent. Before moving on we will describe how each case is handled for our piecewise uniform random variables.

Example: $X = A^2 \div (A - 2)$

It is illuminating to consider how to compute the random variable formed by the quotient of two directly correlated random variables. Let,

$$X = \frac{A^2}{A - 2}$$

which could be rewritten as,

$$A_3 = \frac{A_1}{A_2}$$

where $A_1 = A^2$, $A_2 = A - 2$ and $A_3 = X$. The expression $A_1 \div A_2$ seems to be an element of $\mathbb{A}_2$, but collapses to $\mathbb{A}_1$. Nonetheless, the manner in which we compute X will highlight an issue that is central to computing correlated random variable expressions, namely, joint probability distributions embedded into larger spaces. In this case the joint probability distribution of A_1 and A_2 is a one dimensional embedding into the (A_1, A_2)-space. The figure E.1 depicts the situation,

where,

$$A = \{(-2, -1, 0, 1, 2, 4), (p_1, p_2, p_3, p_4, p_5)\}$$

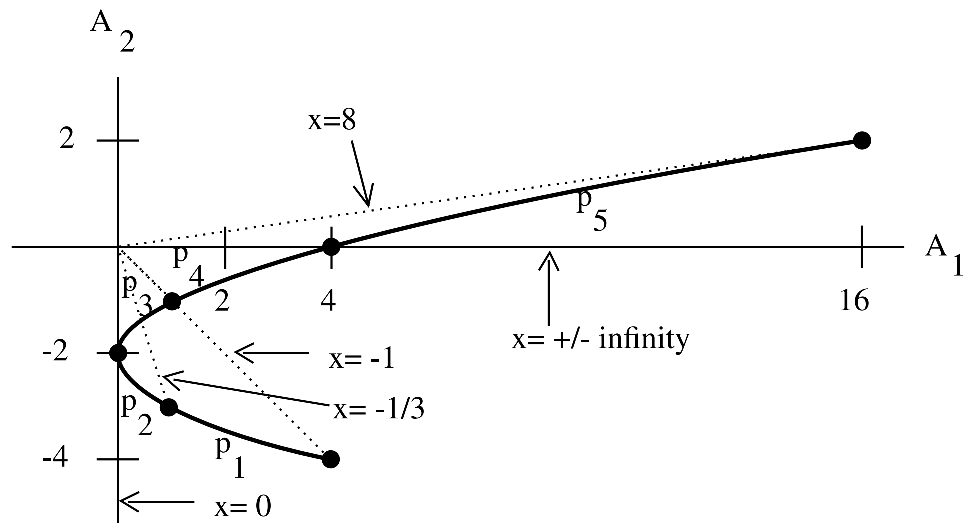

Figure E.1: Quotient of Two Directly Related Random Variables

If we assume that A is partitioned as $(-2, -1, 0, 1, 2, 4)$ the corresponding points in (A_1, A_2) are $((4, -4, (1, -3), (0, -2), (1, -1), (4, 0), (16, 2))$ with corresponding x-values $(-1, -1/3, 0, -1, \pm\infty, 8)$. The interval endpoints are marked with points in figure E.1. We refer to these representations of A_1 and A_2 as being *synchronized* with A. Now suppose we choose the X-partition as,

$$X = \{(-\infty, -1, -\frac{1}{3}, 0, 8, \infty), (q_1, q_2, q_3, q_4, q_5)\}$$

Proceeding by inspection of figure E.1 we estimate the probability values of X to be,

$$\begin{aligned} P(-\infty < X < -1) &= p_4 \\ P(-1 < X < -\frac{1}{3}) &= p_1 + \frac{1}{2}p_3 \\ P(-\frac{1}{3} < X < 0) &= p_2 + \frac{1}{2}p_3 \\ P(0 < X < 8) &= 0 \\ p(8 < X < \infty) &= p_5 \end{aligned}$$

	Expression	a_1	a_2	a_3	a_4	a_5	a_6
A		-2	-1	0	1	2	4
A_1	A^2	4	1	0	1	4	16
A_2	$A-2$	-4	-3	-2	-1	0	2
X	$A_1 \div A_2$	-1	-$\frac{1}{3}$	0	-1	$\pm\infty$	8

Table E.1: Syncrhonized Partitions Table

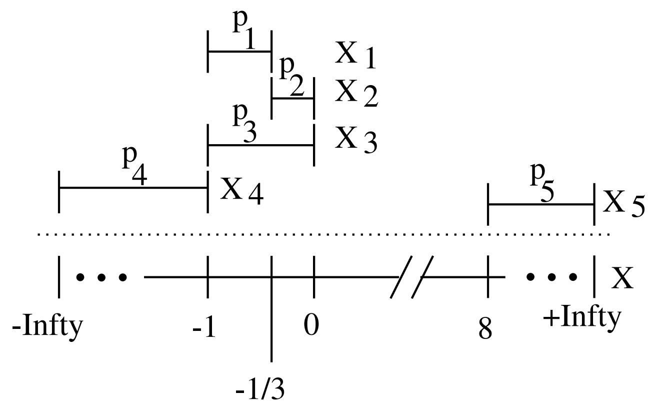

Figure E.2: Folding of Quotient of Two Directly Related Random Variables

The significance of this example is that it informs us how to proceed in what follows. We notice that the partition of A is synchronized with the partitions of A_1 and A_2 which in turn is synchronized to their quotient as shown in table E.1.

To find the probability values (q_1, q_2, q_3, q_4) for X we take each interval of X for which we have probability values $(p_1, ..., p_5)$ and allocated them to the chosen partition for X. In figure E.2 the X_3 interval, $(-1, 0)$ with associated probability p_3 is most interesting since it must be split over two two X partition intervals, $(-1, -1/3)$ and $(-1/3, 0)$.

The X_1 and X_2 intervals happen to line up precisely with respective X-partition intervals. The X_4 and X_5 intervals have infinite endpoints. There is an ambiguity of which polarity of infinity should be used for these two intervals. We have chosen $(-\infty, -1)$ for X_4 and $(8, +\infty)$ for X_5 because it is consistent with the graph, but a specific algorithm will have to wait until we have established the specific programmatic structures to represent and perform these calculations numerically.

Assuming uniform probability distributions within intervals we conclude for interval $X_3 = \{(-1, 0), (p_3)\}$ in our example that X partitions $(-1, -1/3)$ and $(-1/3, 0)$ should receive $\frac{2}{3}p_3$ and $\frac{1}{3}p_3$ respectively. Using this numerical, rather than inspection, method we find that our resultant X is,

$$X = \{(-\infty, -1, -\frac{1}{3}, 0, 8, +\infty), (p_4, p_1 + \frac{2}{3}p_3, p_2 + \frac{1}{3}p_3, 0, p_5)\}$$

When a random variable is presented with a first list of distinct values in ascending order and a second list of probability values presumed interstitial to the first list we say that random variable is in *proper form*. This as opposed to the synchronized form we used above.

We have solved a one dimensional example in a manner consistent with our approach for two dimensions and higher. Rather than partitioning a one dimensional interval such as X_3 is our example using a given X-partition we will, in future examples, partition two dimensional and higher objects.

E.1.4 Enumerating Algebraically Correlated Cases

Introducing more random variables and versions of random variables into an expression requires some rules and key observations in order to avoid a combinatorial explosion of cases.

Consider the expression $(A_1 + B_1)/A_2$. Since we have demonstrated that $1/A_2$ may be replaced by its reciprocal, renamed A_2 the expression is computationally equivalent to $(A_1 + B_1) \times A_2$. Since multiplication is commutative, the original expression is equivalent to $A_2 \times (A_1 + B_1)$. We note, in contrast, that the expression $A_2 \div (A_1 + B_1)$ is not subject to the same analysis. We do not claim that $1/(A_1 + B_1)$ is computationally equivalent to $(A_1 + B_1)$.

As a further contrasting example note that $A_2 \div (A_1 \times B_1)$ is the same as $A_2 \times ((1/A_1) \times (1/B_1))$ and since we may replace the reciprocal of a basic random variable with a basic random variable re-using the same name the expression is computationally equivalent to $A_2 \times (A_1 \times B_1)$. By associativity, this expression is equal to $(A_2 \times A_1) \times B_1)$ and by our combination rule we may replace $(A_2 \times A_1)$ with a single directly related random variable and are allowed to recycle one of the names to $A_1 \times B_1$. This shows that an apparent $\mathbb{A}_{21}$ case may collapse to a $\mathbb{A}_{11}$ case.

Computational Equivalence Rules

Expanding the above discussion into a set of rules for computational equivalence,

1. $-A_1 \sim A_1$
2. $1/A_1 \sim A_1$
3. $(A_1 * A_2) \sim A_1$ for any $* = (+, -, \times, \div)$
4. $(A_1 \div B_1) \sim (A_1 \times B_1)$
5. $(B_1 * A_1) \sim (A_1 * B_1)$ for any $* = (+, -, \times, \div)$
6. $(A_1 * B_1) * A_2 \sim A_2 * (A_1 * B_1)$ unless $A_2 \div (A_1 + B_1)$
7. $(A_1 + B_1) + A_2 \sim A_1 + B_1$
8. $(A_1 \times B_1) \times A_2 \sim A_1 \times B_1$
9. $(A_1 + B_1) \times A_2 \sim A_1 \times B_1 + A_2$

The last three rules are redundant since they recognize associativity and distributivity, but are enumerated for convenience.

Notice that since expressions with instances of the generic operator "$*$" represent sets of expressions we have,

$$(A_1 * B_1) * A_2 \subset A_2 * (A_1 * B_1)$$

because $(A_1 + B_1) \div A_2 \sim (A_1 + B_1) \times A_2$ while $A_2 \div (A_1 + B_1)$ is a separate expression.

Case Enumeration Procedure

To address cases beyond $\mathbb{A}_{11}$ we will use the following procedural steps,

1. Present the variables names involved. E.g. $\mathbb{A}_{21} = A_1 * A_2 * B_1$
2. Show all distinct expressions by permutation of variable names.
3. Show all distinct expressions through association of pairs of variables.
4. Show all distinct expressions where the generic operation "$*$" is replaced with supported arithmetic operations.

Notice that in the cases of A_{22} there are two version of A and two version of B present namely $A_1 * A_2 * B_1 * B_2$ so the permutation $B_2 * B_1 * A_1 * A_2$ is the same as $A_1 * A_2 * B_1 * B_2$ because the names A and B may be swapped since they appear in equal number and the subscripts can be swapped, that is, $A_2 * A_1 \sim A_1 * A_2$.

E.1.5 Enumerating the $\mathbb{A}_{21}$ Case

Following our case enumeration steps,

1. $\mathbb{A}_{21} = A_1 * A_2 * B_1$

2. There are $3!/2! = 3$ permutations of $\mathbb{A}_{21}$,

$$A_1 * A_2 * B_1 \qquad A_1 * B_1 * A_2 \qquad B_1 * A_1 * A_2$$

3. Applying associations our expressions become,

$$A_1 * (A_2 * B_1) \quad (A_1 * B_1) * A_2 \quad A_1 * (B_1 * A_2) \quad (B_1 * A_1) * A_2$$

   Applying the equivalence and renaming rules we see that,

$$\begin{aligned} A_1 * (A_2 * B_1) &\sim A_2 * (A_1 * B_1) \\ (A_1 * B_1) * A_2 &\subset A_2 * (A_1 * B_1) \\ A_1 * (B_1 * A_2) &\sim A_2 * (A_1 * B_1) \\ (B_1 * A_1) * A_2 &\subset A_2 * (A_1 * B_1) \end{aligned}$$

   Leaving only the generic case: $A_2 * (A_1 * B_1)$.

4. Introducing specific arithmetic operations we realize by enumeration rule 5 that there are $3*2 = 6$ cases which we represent symbolically as $(+, \times, \div) \otimes (+, \times)$. In the following enumeration we indicate which cases we will retain as distinct and particular to $\mathbb{A}_{21}$ and which are relegate to case $\mathbb{A}_{11}$ and so have already been addressed,

$$\begin{array}{ll} A_2 + (A_1 + B_1) & \in \mathbb{A}_{11} \text{ by rule 7} \\ A_2 \times (A_1 + B_1) & \sim A_1 + (A_2 \times B_1) \in \mathbb{A}_{21} \text{ by rule 9} \\ A_2 \div (A_1 + B_1) & \sim A_1 \div (A_2 + B_1) \in \mathbb{A}_{21} \\ A_2 + (A_1 \times B_1) & \sim A_1 + (A_1 \times B_1) \\ A_2 \times (A_1 \times B_1) & \in \mathbb{A}_{11} \text{ by rule 8} \\ A_2 \div (A_1 \times B_1) & \in \mathbb{A}_{11} \text{ by rules 5 and 8} \end{array}$$

Thus we find two irreducible cases particular to $\mathbb{A}_{21}$ and write,

$$\mathbb{A}_{21} = \{A_1 + (A_2 \times B_1), A_2 \div (A_1 + B_1)\}$$

These two cases represent something new. They are each result in a random variable that is arithmetically correlated to two independent random variables. We will describe below how to numerically compute these resultant random variables.

An Example: $X = A_1 + (A_2 \times B_1)$

In the computation of correlated random variables section we will present a software implementation for making such calculations in a manner that balances computational performance and result fidelity. Our goal with this example is to expose some of the issues that our eventual software implementation must address.

We first recognize that the expression $A_1 + (A_2 \times B_1)$ represents a class of algebraic expressions depending on the particular choice for A_1, A_2 and B_1. Some such choices can be handled by techniques we have presented so far. If, for example, we choose $Y = A + A \times B$ we see that we can factor out the A and form $Y = A \times (1 + B)$ which may be rewritten as $Y = A \times B_1$ and we see the expression reduces so a previous case..

We now present an example in the class of $A_1 + (A_2 \times B_1)$ which have not yet described how to handle,

$$\begin{aligned} X &= (A^2 + A) + ((A+1) \times (1/B + 1)) \\ \text{Let } A_1 &= A^2 + A \\ \text{Let } A_2 &= A + 1 \\ \text{Let } B_1 &= 1/B + 1 \\ X &= A_1 + A_2 \times B_1 \end{aligned}$$

We recognize this expression for X as a proper element of $\mathbb{A}_{21}$.

To approach this example problem we recall that A and B are each partitioned into intervals with associated probability values since each is a piecewise uniform distribution. We presume that we have established a partition for X and that our task in computing X is to assign probability values to each of its partition intervals. This task is broken down into finding the probability contribution each partition interval of A and B makes to the partition intervals of X. In figure E.3 we show the rectangular joint probability distribution jointly partitioned by A and B. The A_1, A_2 and B_1 axes are referred to as *ancillary* since the share partition endpoints with the A and B axes, but are not assumed monotonic. For example, a value such as zero may appear more than once on an ancillary axis.

We refer to the rectangle $(a_i, a_{i+1}) \times (b_j, b_{j+1})$ as the i,j-rectangle and if $P((a_i, a_{i+1})) = p_i$ and $P((b_j, b_{j+1})) = q_j$ then $P(i, j - rectangle) = p_i \times q_j$, the joint probability.

To find the probabilities allocated to each partition element of the resultant X we use the partition of X to partition each i,j-rectangle and allocate the resulting

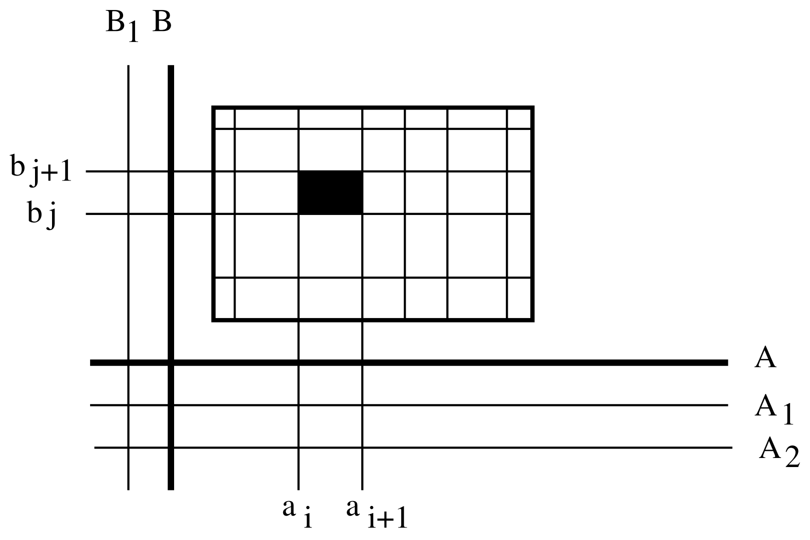

Figure E.3: Partition of Joint Rectangle

fractions of the joint probability associated with the i, j-rectangle to the respective partition elements of X.

For clarity of this example we suppose that A and B are uniformly distributed and that,

$$A \sim U(1,2)$$
$$B \sim U(\frac{1}{2},1)$$

Since A_1 and A_2 are functions of A and B_1 is a function of B their synchronous forms are,

$$A_1 = (2,6)$$
$$A_2 = (2,3)$$
$$B_1 = (3,2)$$

Just as in the previous example we see that A_1 and A_2 form a parametric curve with respect to A shown in figure E.4.

To find the joint probability density we find the direct product of the (A_1, A_2) graph and B_1 with result shown in figure E.5.

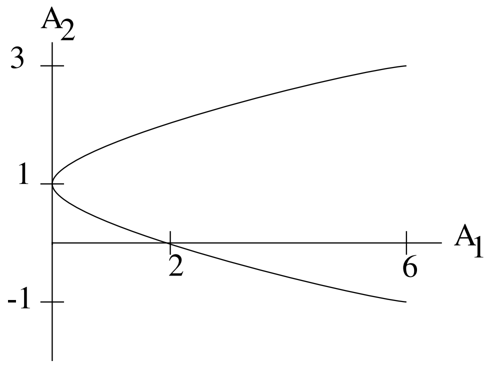

Figure E.4: Joint Distribution of Two Directly Related Random Variables

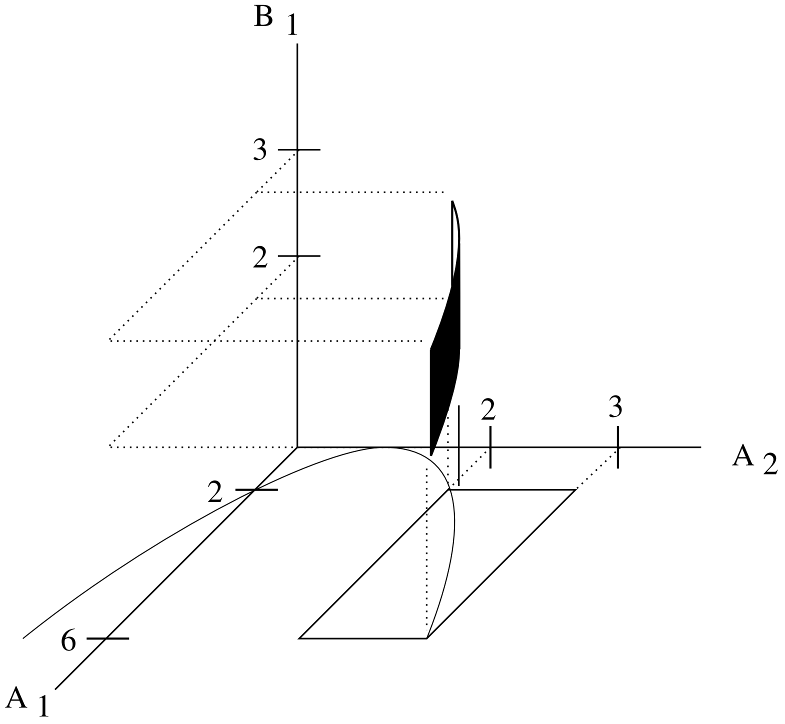

Figure E.5: Joint Distribution of an $\mathbb{A}_{21}$ Case

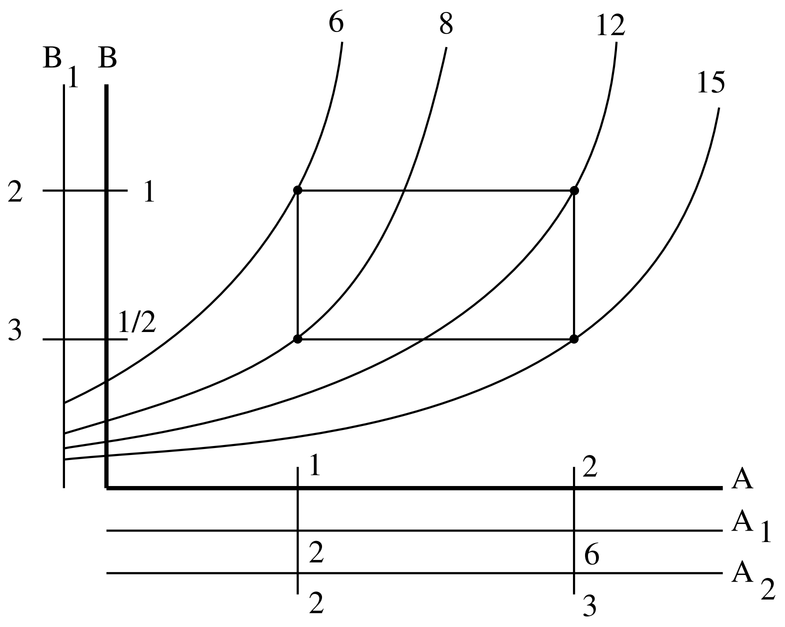

Figure E.6: Partition of Joint Distribution of an $\mathbb{A}_{21}$ Case

A given partition of X implies a partition of the (A_1, A_2, B_1)-space. The joint probability in this example is distributed over the section of cylindrical parabola shown in figure E.5. The partitions of X will appear as non-intersecting level surfaces.

We have shown above that our assumption of uniform probability distribution over each partition interval of each first class random variable, A and B in this case, does not necessarily result in a uniform distribution of probability over the joint surface shown above. This frees us to make the assumption that transforming the X-partition surfaces and joint probability surface to (A, B)-space will result in similar partitioning of the joint probability into the X-partition elements.

In figure E.6 we show the joint probability area of A_1, A_2 and B_1 with respect to the first class A and B. This area is always rectangular because A and B are disjoint. We include the ancillary axes A_1, A_2 and B_1 as before. We further indicate the x-values for the vertices and the X level curves that pass through the vertices. If we choose $(6, 8, 12, 15)$ as the partition for X is shown in figure E.6.

To find the x level curves we solve $X = A_1 + A_2 \times B_1$ for $B(A|x)$, that is, B as a function of A for a fixed value of x. In this example we find,

$$\begin{aligned}
x &= A_1 + A_2 \times B_1 \\
&= A^2 + A + (A+1) \times (\frac{1}{B} + 1) \\
B(A|x) &= \frac{A+1}{x - (A^2 + A) - (A+1)} \\
&= \frac{A+1}{x - (A+1)^2}
\end{aligned}$$

The fraction of the total area of the rectangle between any two adjacent x-curves is proportional to the amount of probability allocated to the corresponding X-partition element.

To recap, we are assuming that the joint probability of A and B in this example is distributed uniformly over this rectangle. Since A and B happen to be uniform random variables this is the only joint uniformly distributed rectangle in the example and carries a probability of one. Visually we estimate X to be,

$$X = \{(6, 8, 12, 15), (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})\}$$

To compute the probability values of X we either must integrate the area between x curves (within X partition intervals) or make further assumptions to reduce the problem to computations of triangles and trapezoids. We defer that discussion to the computation section. It turns out we will find it advantageous to make a different choice.

We realize from this example that some expressions of the form $A_1 + A_2 \times B_1$ are reducible to the previously discussed independent case and some, like the example addressed here, are not reducible. By focusing on the (A, B)-space partition for $\mathbb{A}_{nm}$ cases we will find a unified computation approach.

An Example: $X = A_1 + (A_2 \times B_1)$, Again

To better understand how level curves affect the computation of correlated random variables we revisit the previous example. Let,

$$X = A_1 + (A_2 \times B_1)$$

Rather than choosing specific functions f and g so that $A_1 = f(A)$ and $A_2 = g(A)$ for the common random variable A, we instead assume $A_2 = m_i \times A_1 + b_i$, where m_i and b_i are constants that depend on the partition interval of A_1. This allows us to better understand the form $A_1 + (A_2 + B_1)$ in general terms.

We choose the i^{th} partition of A_1 and j^{th} partition of B_1 and suppose as before,

$$\begin{aligned} A_1 &= (2, 6) \\ A_2 &= (2, 3) \\ B_1 &= (3, 2) \end{aligned}$$

Assuming the affine relation between A_1 and A_2 for this partition interval choice we find,

$$\begin{aligned} A_2 &= \frac{3-2}{6-2} \times (A_1 - 2) + 2 \\ &= \frac{1}{4} A_1 + \frac{3}{2} \end{aligned}$$

To find the x-level curves we find X to be x and solve for $B_1(A_1|x)$ in,

$$\begin{aligned} x &= A_1 + (A_2 \times B_1) \\ B_1 &= \frac{x - A_1}{A_2} \\ B_1(A_1|x) &= \frac{x - A_1}{\frac{1}{4} A_1 + \frac{3}{2}} \end{aligned}$$

We again suppose the partition for X to be $(6, 8, 12, 15)$ and plot the level curves with the i, j-rectangle as shown in figure E.7.

where we have A_1 and B_1 as axes and A_2 as the sole ancillary axis. We have kept the orientation of B_1 the same as the previous example for ease of comparison. We find that the x-level curves in this example compare favorably to the previous example even though this example made no assumptions about the specific relationship between A_1 and A or between A_2 and A. All that was assumed here is that A_1 and A_2 are directly related and supposed the relationship to be affine.

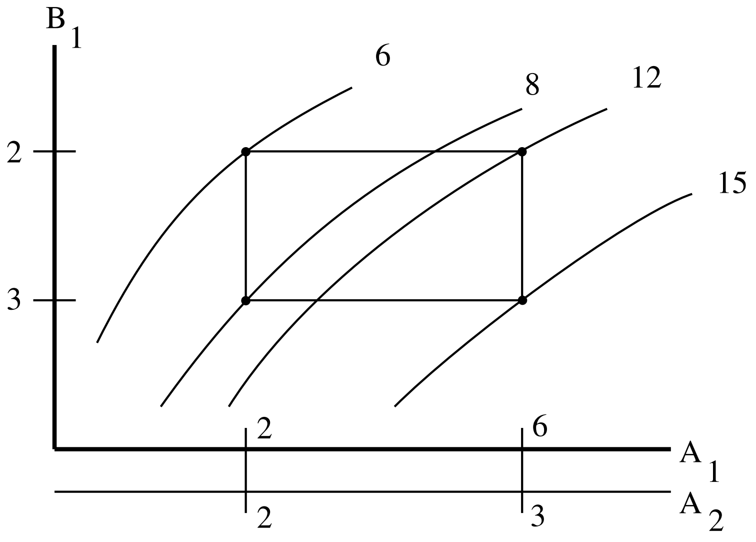

Figure E.7: Alternative Treatment of an $\mathbb{A}_{21}$ Case

This technique of assuming a piecewise affine relationship between versions of a random variable in an expression will be adopted for the remainder of this section.

Computation Technique for $A_1 \div (A_2 + B_1)$

With so much groundwork laid in the previous section we may proceed more directly. We again form the expression,

$$x = A_1 \div (A_2 + B_1)$$

where x is a constant representing the value of a level curve. We assume that working copies of A and B are appropriately partitioned and that we are describing how to compute a specific pair, (i, j) of partition intervals, one from A and the other from B. That is, $A^i = [a_1, a_2]$ and $B^j = [b_1, b_2]$. For this choice of partition interval pair we further assume an affine relationship between A_1 and A_2 as,

$$A_2(A_1) = m * (A_1 - a_{11}) + a_{21}$$

where we have dropped the superscripts from A_1^i and A_2^i for clarity and assumed as before the relevant intervals,

$$A_1^i = U(a_{11}, a_{12}) \qquad A_2^i = U(a_{21}, a_{22}) \qquad B_1^j = U(b_{11}, b_{12})$$

and that $m = \frac{a_{22}-a_{21}}{a_{12}-a_{11}}$. Solving for B_1 in terms of A_1 and x,

$$\begin{aligned} B_1(A_1|x) &= \frac{A_1}{x} - A_2 \\ &= \frac{A_1}{x} - m(A_1 - a_{11}) - a_{21} \\ &= (\frac{1}{x} - m)A_1 + (ma_{11} - a_{21}) \end{aligned}$$

Notice that the B_1-axis intercept occurs at the same distance from the origin as the $A_1|A_2 = 0$ point on the A_1-axis. Thus the effect of introducing an extra version of A, namely A_2 into the calculation of $A_1 \div B_1$ is an offsetting from the origin of the focal point of the level curves.

Using the values from the previous example our new result is shown in figure E.8.

where we have again chosen a partition of $X = A_1 \div (A_2 + B_1)$ that happens to pass through the vertices of the (i, j) partition cell for illustration purposes. Unlike the previous example where the level curves are actually curved it is very computationally straight-forward to calculate the areas enclosed by the x-partition intervals for each (i, j) partition cell. This will be explored further after the $\mathbb{A}_{211}$ cases is described.

Further Considerations of the $\mathbb{A}_{21}$ Case

The practical issues involved in computing $(2, 1)$ cases of correlated random variables include maintaining a construction history for each computed variable. This means that if an expression for some variable, call it X, is formed during model creation the supporting system must remember which input random variables are involved in forming X. If X depends solely on input random variable A then we may internally refer to it as A_1 regardless of the name given it by the model developer. This amounts to maintaining a *parse tree* which is a standard object in computer science.

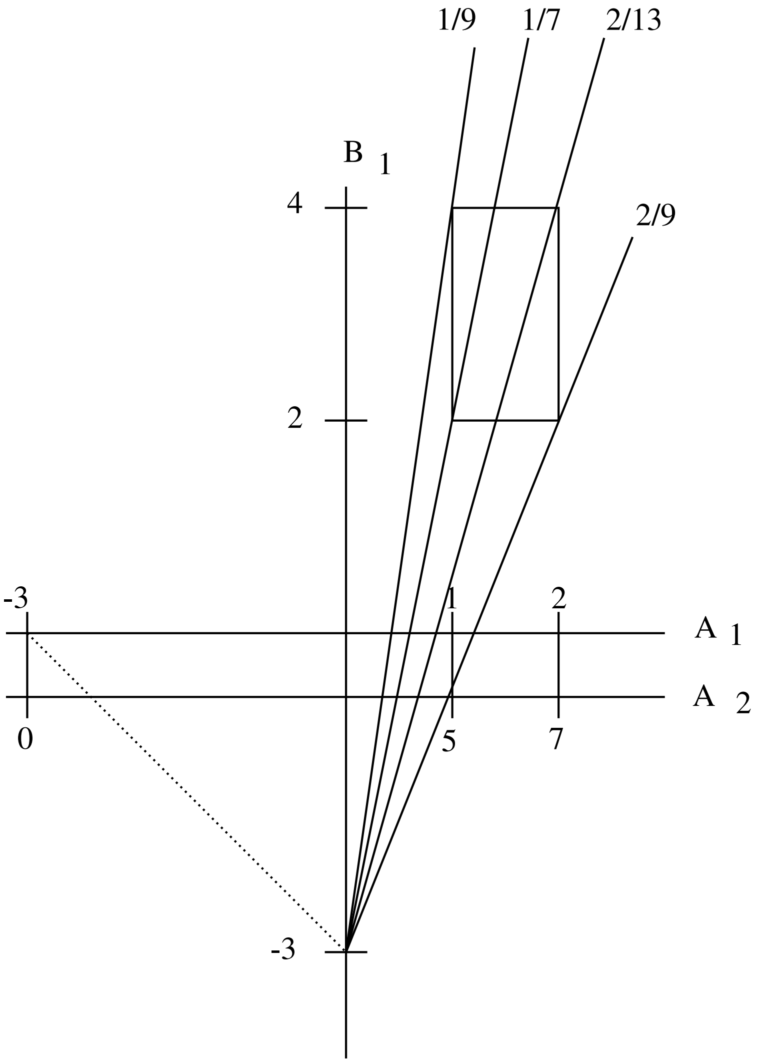

Figure E.8: Example of an $\mathbb{A}_{21}$ Case

To form a new *model variable* (any variable created within the model) the PHoX system goes through an exercise similar to the one we used to reduce our $(2,1)$ cases to one of four special cases after first determining that we are indeed facing a $(2,1)$ case.

Running a model within the PHoX system involves several *passes*. The first pass in a *compiler* pass where all the model variables are formed and analyzed as to their particular case and which compute strategy will be employed. This includes identifying poles and peaks for each rational polynomial. Identifying the poles and peaks can be performed internally to the PHoX system or off-loaded to a symbolic solver. This compiler pass only needs to be performed when the model code changes, not for changes to input data. Compiling is an exercise in structural analysis of model code. This has favorable implications for model performance.

E.1.6 Enumerating the $\mathbb{A}_{31}$ Case

Following our case enumeration steps,

1. $\mathbb{A}_{31} = A_1 * A_2 * A_3 * B_1$
2. There are $4!/3! = 4$ permutations of $\mathbb{A}_{31}$,

   $$A_1 * A_2 * A_3 * B_1, \quad A_1 * A_2 * B_1 * A_3, \quad A_1 * B_1 * A_2 * A_3, \quad B_1 * A_1 * A_2 * A_3$$

3. Applying associations our expressions become,

   $$\begin{array}{lll} A_1 * (A_2 * (A_3 * B_1)), & & \\ (A_1 * (A_2 * B_1)) * A_3, & A_1 * (A_2 * (B_1 * A_3)), & \\ (A_1 * (A_2 * B_1) * A_3, & A_1 * ((A_2 * B_1) * A_3), & \\ ((A_1 * B_1) * A_2) * A_3, & (A_1 * (B_1 * A_2)) * A_3, & A_1 * ((B_1 * A_2) * A_3), \\ ((B_1 * A_1) * A_2) * A_3 & & \end{array}$$

   Applying the equivalence and renaming rules leaves us with a single case,

   $$A_1 * (A_2 * (A_3 * B_1))$$

   We may write this as,

   $$A_1 \otimes \mathbb{A}_{21}$$

where we have already reduced the $(A_2 * (A_3 * B_1))$ expression to,

$$A_3 \times B_1 + A_2 \qquad\qquad A_2 \div (A_3 + B_1)$$

so that we must expand,

$$A_1 * (A_3 \times B_1 + A_2) \qquad\qquad A_1 * (A_2 \div (A_3 + B_1)$$

4. Expanding the cases we find the following $3 + 3 = 6$ cases,

$$\begin{aligned}
A_1 + (A_3 \times B_1 + A_2) &\qquad \sim A_1 + A_2 \times B_2 \text{ by rule 7}\\
A_1 \times (A_3 \times B_1 + A_2) &\qquad \sim A_1 + A_2 \times B_2 \text{ by rule 3}\\
A_1 \div (A_3 \times B_1 + A_2) &\qquad \sim A_1 + A_2 \times B_2 \text{ by rule 7}\\
A_1 + (A_2 \div (A_3 + B_1) &\qquad \in \mathbb{A}_{31}\\
A_1 \times (A_2 \div (A_3 + B_1) &\qquad \sim A_2 \div (A_3 + B_1) \text{ by rule 3}\\
A_1 \div (A_2 \div (A_3 + B_1) &\qquad \sim A_1 + A_2 \times B_2 \text{ by rule 7}
\end{aligned}$$

We see that we have one irreducible case peculiar to $\mathbb{A}_{31}$ and write,

$$\mathbb{A}_{31} = A_1 + (A_2 \div (A_3 + B_1)$$

Computation Technique for $A_1 + (A_2 \div (A_3 + B_1))$

We follow the same patter as before but assume, for the $(i, j)^{th}$ partition interval of A and B that A_2 and A_3 are both affine transforms of A_1 and write,

$$\begin{aligned}
A_2(A_1) &= m(A_1 - a_{11}) + a_{21}\\
A_3(A_1) &= n(A_1 - a_{11}) + a_{31}
\end{aligned}$$

where,

$$\begin{aligned}
A_1^i &= U(a_{11}, a_{12}) \qquad A_2^i = U(a_{21}, a_{22}) \qquad A_3^i = U(a_{31}, a_{32})\\
B_1^j &= U(b_{11}, b_{12})\\
m &= \frac{a_{22} - a_{21}}{a_{12} - a_{11}} \qquad n = \frac{a_{32} - a_{31}}{a_{12} - a_{11}}
\end{aligned}$$

Assuming $X = A_1 + (A_2 \div (A_3 + B_1))$ and fixing $X = x$ to find the level curves of $B_1(A_1|x)$ we have,

$$\begin{aligned} x &= A1 + A_2 \div (A_3 + B_1) \\ (x - A_1) \times (A_3 + B_1) &= A_2 \\ B_1(A_1|x) &= A_2 \div (x - A_1) - A_3 \\ &= \frac{m(A_1 - a_{11}) + a_{21}}{x - A_1} - n(A_1 - a_{11}) - a_{31} \end{aligned}$$

To illustrate with our familiar example where,

$$A_1 = U(1,2) \qquad A_2 = U(5,7) \qquad A_3 = U(8,9)$$

we see level curves asymptotic to their own x value on the A_1-axis shown in figure E.9.

We notice for future reference that the level curves are almost straight lines in this example chosen. While this situation may not persist for all examples it suggest that a piecewise linear approximation to the level curves may be a computational compromise over time-consuming calculation for each (i, j) interval and each X-partition interval.

E.1.7 Enumerating the $\mathbb{A}_{41}$ Case and Beyond

Now that we have detailed the previous cases and come to $\mathbb{A}_{41}$ we realize that a full enumeration and association of $A_1 * A_2 * A_3 * A_4 * B_1$ is not necessary. There is only one version of B present. It will be associated with a version of A which we assume without loss of generality to be A_4 operating on the left giving $(A_4 * B_1)$. To this expression we associate one of the remaining versions of A which we assume to be A_3 again operating on the left since this is the most general case, $(A_3 * (A_4 * B_1))$. Proceeding similarly for the remaining versions of A we find the one generic expression to be,

$$A_1 * (A_2 * (A_3 * (A_4 * B_1)))$$

Furthermore we recognize this expression as,

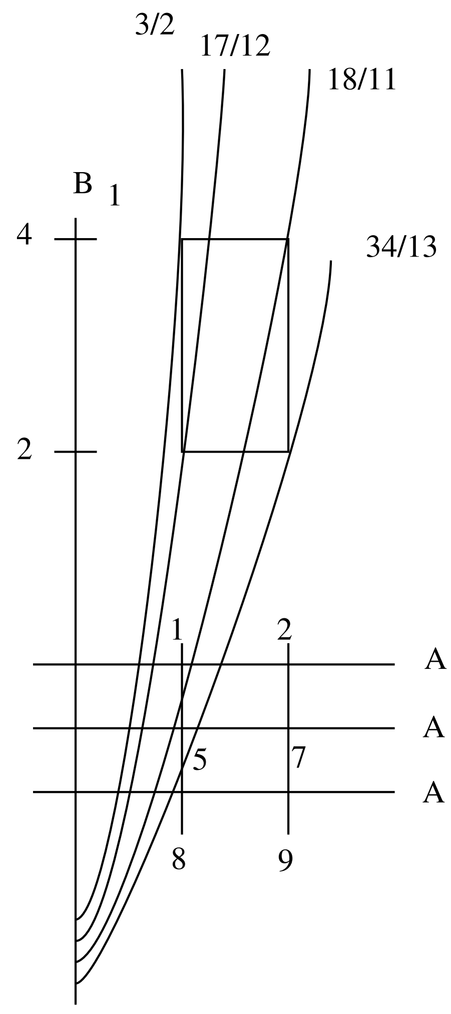

Figure E.9: Example of an $\mathbb{A}_{31}$ Case

$$A_1 \otimes \mathbb{A}_{31}$$

and since $\mathbb{A}_{31}$ has only one case peculiar to it we have,

$$A_1 * (A_2 + A_3 \div (A_4 + B_1))$$

which we enumerate as,

$$\begin{aligned}
&A_1 + (A_2 + A_3 \div (A_4 + B_1)) && \sim A_1 + A_2 \div (A_3 + B_1) \in \mathbb{A}_{31} \\
&A_1 \times (A_2 + A_3 \div (A_4 + B_1)) && \sim A_1 + A_2 \div (A_3 + B_1) \in \mathbb{A}_{31} \\
&A_1 \div (A_2 + A_3 \div (A_4 + B_1)) && \in \mathbb{A}_{41}
\end{aligned}$$

Thus the only irreducible case particular to $\mathbb{A}_{41}$ is,

$$\mathbb{A}_{41} = (\frac{A_1}{A_2 + \frac{A_3}{A_4+B_1}})$$

If we relabeled the versions of A and added a new version of A on the left we would have,

$$\mathbb{A}_{51} = (A_1 + \frac{A_2}{A_3 + \frac{A_4}{A_5+B_1}})$$

Notice that if we multiply on the left by A_1 instead of add, the A_1 would combine with A_2 in the numerator and reduce the case to $\mathbb{A}_{41}$. If we divide on the left by A_1 instead of add we would have,

$$\begin{aligned}
\frac{A_1}{\frac{A_2}{A_3+\frac{A_4}{A_5+B_1}}} &= \frac{A_1}{A_2}(A_3 + \frac{A_4}{A_5 + B_1}) \\
&\sim A_1 + \frac{A_2}{A_3 + B_1}
\end{aligned}$$

which is an element of $\mathbb{A}_{31}$ and not particular to $\mathbb{A}_{51}$. Thus our statement of $\mathbb{A}_{51}$ having a single irreducible element particular to it stands.

The emerging pattern suggests that $\mathbb{A}_{n1}$ for $n > 2 \in \mathbb{N}$ each consist of a single irreducible expression particular to it in the form of a continued fraction.

E.1.8 Enumerating the $\mathbb{A}_{111}$ Case

Before we enumerate the $\mathbb{A}_{211}$ cases we must consider the $\mathbb{A}_{111}$ case. It is represented by,

$$\mathbb{A}_{111} = A_1 * B_1 * C_1$$

Since we have equal numbers of each version of random variable, that is one each, there is only one permutation. Introducing associations,

$$(A_1 * B_1) * C_1 \qquad\qquad A_1 * (B_1 * C_1)$$

reveals a property that will be important when describing the $\mathbb{A}_{211}$ case. We notice that $(A_1 * B_1) \in \mathbb{A}_{11}$ and that $(A_1 * B_1)$ is independent of C_1. Letting $AB_{11} := (A_1 * B_1)$ for some operation "$*$" we see that $AB_{11} * C_1 \in \mathbb{A}_{11}$. Similarly $A_1 * BC_{11} \in \mathbb{A}_{11}$ meaning that there are no elements particular to $\mathbb{A}_{111}$. In that sense $\mathbb{A}_{111}$ is empty.

The point to understand from the $\mathbb{A}_{111}$ case is that while it is possible to form a three dimensional joint distribution for A_1, B_1 and C_1, it is unnecessary. We say that expressions of the form $(A_1 * B_1) * C_1$ or $A_1 * (B_1 * C_1)$ are *sequential*, that is, they may be computed sequentially.

E.1.9 Enumerating the $\mathbb{A}_{211}$ Case

Following the enumeration steps we developed for two dimensional arithmetically correlated random variables we have,

1. The generic case for $\mathbb{A}_{211}$ is $A_1 * A_2 * B_1 * C_1$.

2. There are $4!/2!/2! = 6$ permutations since B_1 is interchangeable with C_1 because there are the same number of versions (one) for each.

$$\begin{array}{lll} A_1 * A_2 * B_1 * C_1, & A_1 * B_1 * A_2 * C_1, & A_1 * B_1 * C_1 * A_2, \\ B_1 * A_1 * A_2 * C_1, & B_1 * A_1 * C_1 * A_2, & B_1 * C_1 * A_1 * A_2 \end{array}$$

3. Introducing associations we recognize that associating $(A_1 * A_2)$ would reduce the expression to $\mathbb{A}_{111}$ and similarly associating $(B_1 * C_1)$ would reduce

the expression to $\mathbb{A}_{21}$.

$$\begin{array}{lll}
(A_1 * (A_2 * B_1)) * C_1 & A_1 * ((A_2 * B_1) * C_1) & \\
((A_1 * B_1) * A_2) * C_1 & (A_1 * B_1) * (A_2 * C_1) & A_1 * (B_1 * (A_2 * C_1)) \\
(A_1 * (B_1 * A_2)) * C_1 & A_1 * ((B_1 * A_2) * C_1) & \\
((A_1 * B_1) * C_1) * A_2 & A_1 * (B_1 * (C_1 * A_2)) & (A_1 * B_1) * (C_1 * A_2) \\
((B_1 * A_1) * A_2) * C_1 & (B_1 * A_1) * (A_2 * C_1) & B_1 * (A_1 * (A_2 * C_1)) \\
((B_1 * A_1) * C_1) * A_2 & (B_1 * A_1) * (C_1 * A_2) & B_1 * (A_1 * (C_1 * A_2)) \\
(B_1 * (C_1 * A_1)) * A_2 & B_1 * ((C_1 * A_1) * A_2) &
\end{array}$$

We rule the expressions where B_1 or C_1 appears outside the parentheses as sequential and remove them leaving,

$$\begin{array}{lll}
A_1 * ((A_2 * B_1) * C_1) & & \\
(A_1 * B_1) * (A_2 * C_1) & A_1 * (B_1 * (A_2 * C_1)) & \\
A_1 * ((B_1 * A_2) * C_1) & & \\
((A_1 * B_1) * C_1) * A_2 & A_1 * (B_1 * (C_1 * A_2)) & (A_1 * B_1) * (C_1 * A_2) \\
(B_1 * A_1) * (A_2 * C_1) & & \\
((B_1 * A_1) * C_1) * A_2 & (B_1 * A_1) * (C_1 * A_2) & \\
(B_1 * (C_1 * A_1)) * A_2 & &
\end{array}$$

Enumeration Rule 4 allows us to commute basic random variables within an associative pair. If we rewrite the above with the preference of associating $(A_1$ with B_1 and A_2 with C_1 and with A_1 or A_2 appearing first in any pair we create a number of duplicate expressions,

$$\begin{array}{lll}
A_1 * ((A_2 * C_1) * B_1) & & \\
(A_1 * B_1) * (A_2 * C_1) & A_1 * (B_1 * (A_2 * C_1)) & \\
A_1 * ((A_2 * C_1) * B_1) & & \\
((A_1 * B_1) * C_1) * A_2 & A_1 * (B_1 * (A_2 * C_1)) & (A_1 * B_1) * (A_2 * C_1) \\
(A_1 * B_1) * (A_2 * C_1) & & \\
((A_1 * B_1) * C_1) * A_2 & (A_1 * B_1) * (A_2 * C_1) & \\
(C_1 * (A_1 * B_1)) * A_2 & &
\end{array}$$

Removing the duplicates leaves the following with annotations,

$$A_1 * ((A_2 * C_1) * B_1) \subset A_1 * (B_1 * (A_2 * C_1))$$
$$(A_1 * B_1) * (A_2 * C_1)$$
$$A_1 * (B_1 * (A_2 * C_1))$$
$$((A_1 * B_1) * C_1) * A_2 \subset A_2 * (C_1 * (A_1 * B_1)) \sim A_1 * (B_1 * (A_2 * C_1))$$
$$(C_1 * (A_1 * B_1)) * A_2 \subset A_2 * (C_1 * (A_1 * B_1)) \sim A_1 * (B_1 * (A_2 * C_1))$$

Thus we have two remaining generic cases,

$$A_1 * (B_1 * (A_2 * C_1)) \qquad (A_1 * B_1) * (A_2 * C_1)$$

4. Now we introduce specific operations in place of the generic "$*$" operation. Recall that division is computationally similar to multiplication within a basic random variable pair because basic random variables may be replaced by their reciprocal. Recall that division is a separate operation from multiplication is followed by addition of the right-hand operand. That is $B_1 \div (A_2 + C_1)$ is computationally dissimilar from $B_1 \times (A_2 + C_1)$. We first expand the inner parentheses of the first case namely $(B_1 * (A_2 * C_1))$,

$$B_1 + (A_2 + C_1) = A_2 + (B_1 + C_1) \qquad \in \mathbb{A}_{11}$$
$$B_1 + (A_2 \times C_1)$$
$$B_1 \times (A_2 + C_1)$$
$$B_1 \times (A_2 \times C_1) = A_2 \times (B_1 \times C_1) \qquad \in \mathbb{A}_{11}$$
$$B_1 \div (A_2 + C_1)$$

Discarding the reduced $(+, +)$ and $(\times, \times)$ cases we expand the full generic case, $A_1 * (B_1 * (A_2 * C_1))$,

$$A_1 + (B_1 + (A_2 \times C_1)) = B_1 + (A_1 + (A_2 \times C_1)) \qquad \text{sequential}$$
$$A_1 \times (B_1 + (A_2 \times C_1)) \sim A_1 \times B_1 + A_2 \times C_1$$
$$A_1 \div (B_1 + (A_2 \times C_1))$$
$$A_1 + (B_1 \times (A_2 + C_1)) = (A_1 + A_2 \times B_1) + B_1 \times C_1 \qquad \text{sequential}$$
$$A_1 \times (B_1 \times (A_2 + C_1)) \sim (A_1 + (A_2 \times C_1)) \times B_1 \qquad \text{sequential}$$
$$A_1 + (B_1 \div (A_2 + C_1))$$
$$A_1 \times (B_1 \div (A_2 + C_1)) = B_1 \times (A_1 \div (A_2 + C_1)) \qquad \text{sequential}$$

Thus we expand $A_1 * (B_1 * (A_2 * C_1))$ to the following cases,

$$(A_1 \times B_1) + (A_2 \times C_2) \qquad \frac{A_1}{B_1 + (A_2 \times C_1)} \qquad A_1 + \frac{B_1}{A_2 + C_1}$$

Now we turn to the expansion of the $(A_1 * B_1) * (A_2 * C_1)$ expression. Following all the familiar rules,

$$\begin{array}{ll}
(A_1 + B_1) + (A_2 + C_1) \sim (A_1 + B_1) + C_1 & \text{sequential} \\
(A_1 + B_1) + (A_2 \times C_1) \sim (A_1 + A_2 \times C_1) + B_1 & \text{sequential} \\
(A_1 + B_1) \times (A_2 + C_1) & \\
(A_1 + B_1) \times (A_2 \times C_1) \sim (A_1 + (A_2 \times B_1)) + C_1 & \text{sequential} \\
(A_1 \times B_1) + (A_2 + C_1) \sim (A_1 + (A_2 \times B_1)) + C_1 & \text{sequential} \\
(A_1 \times B_1) + (A_2 \times C_1) & \\
(A_1 \times B_1) \times (A_2 + C_1) = (A_1 \times (A_2 + C_1)) \times B_1 & \text{sequential} \\
(A_1 \times B_1) \times (A_2 \times C_1) = (A_1 \times (A_2 \times C_1)) \times B_1 & \text{sequential} \\
(A_1 + B_1) \div (A_2 + C_1) & \\
(A_1 \times B_1) \div (A_2 + C_1) = (A_1 \div (A_2 + C_1)) \times B_1 & \text{sequential}
\end{array}$$

There are three new cases, but one is a duplicate so the final set of computationally dissimilar cases peculiar to $\mathbb{A}_{211}$ are,

$$\mathbb{A}_{211} = \{(A_1 \times B_1) + (A_2 \times C_2),\ (A_1 + B_1) \times (A_2 + C_1), \\ \frac{A_1}{B_1 + (A_2 \times C_1)},\ A_1 + \frac{B_1}{A_2 + C_1},\ \frac{A_1 + B_1}{A_2 + C_1}\}$$

E.1.10 Insights Gained

Taking a *glass box* approach to model analysis as opposed to the traditional black box approach adopted by the Monte Carlo method, as defined in Saltelli [11], leads us directly to the consideration of algebraic correlation. We found that the PHoX modeling system uses a symbolic computation approach to determine the simplest case for processing each mathematical expression involving random variable operands. The PHoX modeling system then uses robust algorithms that balance numerical accuracy and computational performance. The robustness of the algorithms allows the symbolic processing algorithms to be more coarse-gained in their determination of specific mathematical expression case.

An advantage the PHoX approach of direct computation of random variables has over randomization of inputs such as the Monte Carlo approach is that of *coverage*. This advantage will become more clear when we describe the consequences of programmatic conditional statements such as *if*, *then*, *else* have when employing random variable operands.

If, instead of the PHoX approach of arithmetically combining random variables, we had repeatedly run our model with sets of random values chosen from the random variables representing our model inputs, then likely outcomes would necessarily occur most often. More to the point, we would spend most of the model runs in an effort to observe the likely outcomes. A more significant issue arises when we consider unlikely outcomes.

Consider a model the predicts patient outcome when given a certain drug. For simplicity suppose that the output of this model is success (patient survives) or failure (patient dies). If the true likelihood of failure is, say, 1 in 1 billion, then we expect to run a randomized-input type model at one billion times to observe a single failure. To be confident the single observation of failure is one in one billion we need to run the model many times one billion runs so that several failures are observed. Using *computational effort* (time and hardware resources) as a proxy for importance of outcome we see that the randomized-input approach attaches nearly no importance to the most significant outcome, patient death in this case.

In the PHoX-style model there is no importance associated with likelihood and similar computational effort is applied to each quanta of outcome likelihood. In the drug example, PHoX would apply similar computational effort to each of the two outcomes, success and failure, without prejudice for either.

Bibliography

[1] Richard Bellman. *Dynamic Programming*. Dover Publications, Inc., 2003.

[2] Peter Bickel and Kjell Doksum. *Mathematical Statistics Vol. 1, 2nd ed.* Prentice Hall, 2001.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] Richard A. Brealey, Stewart C. Meyers, and Franklin Allen. *Principles of Corporate Finance*. McGraw-Hill Irwin, 2006.

[5] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

[6] Eugene Brigham, Louis Gapenski, and Michael Ehrhardt. *Financial Management, Theory and Practice, 9th ed.* Dryden, 1999.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[8] Sean Dineen. *Probability Theory in Finance: A Mathematical Guide to the Black-Scholes Formula*. Americal Mathematic Society, 2000.

[9] John H. Drew, Diane L. Evans, Andrew G. Glen, and Lawrence M. Leemis. *Computational Probability: Algorithms and Applications in the Mathematical Sciences*. Springer, 2008.

[10] A. Saltelli et al. *Sensitivity Analysis in Practice*. John Wiley and Sons Ltd., 2004.

[11] A. Saltelli et al. *Global Sensitivity Analysis, The Primer*. John Wiley and Sons Ltd., 2008.

[12] Thomas Robert Fielden. *Modeling Market and Regulatory Mechanisms for Pollution Abatement with Sharp and Random Variables*. PhD thesis, Portland State University, 2012.

[13] Saul I. Gass. *Linear Programming Methods and Applications 4th ed.* McGraw-Hill Book Company, 1975.

[14] Harvey J. Greenberg. Mathematical programming models for environmental quality control. *Operations Research*, 43(4):578–622, 1995.

[15] Matthias Kohl and Peter Ruckdeschel. Implementation of random variables. Technical report, Stamats.de, 2011.

[16] J. P. Nolan. *Stable Distributions - Models for Heavy Tailed Data*. Birkhauser, Boston, 2013. In progress, Chapter 1 online at academic2.american.edu/∼jpnolan.

[17] Martin A. Tanner. *Tools for Statistical Inference, 3rd ed.* Springer, 1996.

[18] Robert C. Tausworthe. Finding every root of a broad class of real continuous functions in a given interval. *IPN Progress Report*, 42(176), 2009.

[19] Jonathan Weare. Efficient monte carlo sampling by parallel marginalization. *Proc. of the Nat. Academy of Sciences of the United States of America*, 104(31):12657–12662, 2007.

[20] Zhengqiu Zhang. An improvement to the brents method. *International Journal of Experimental Algorithms (IJEA)*, 2(1), 2011.