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 [2], 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

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

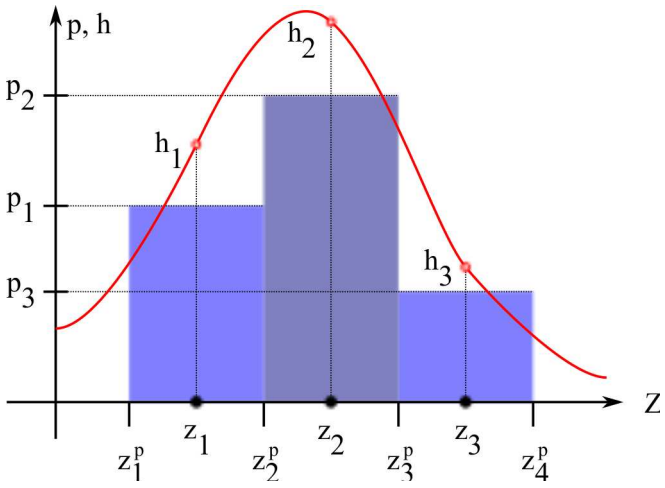

Figure 1.1: Three point Partition of Z

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.

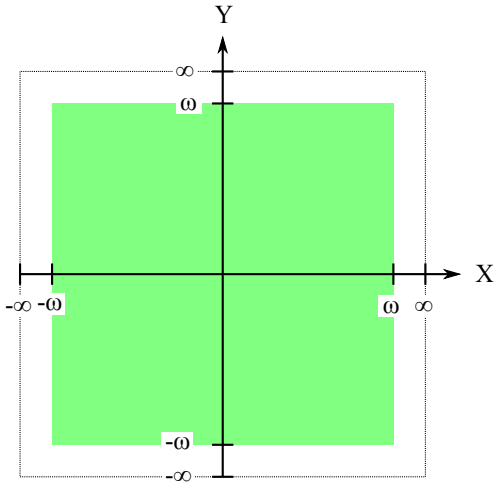

Figure 1.2: Fundamental Numeric Domain

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

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

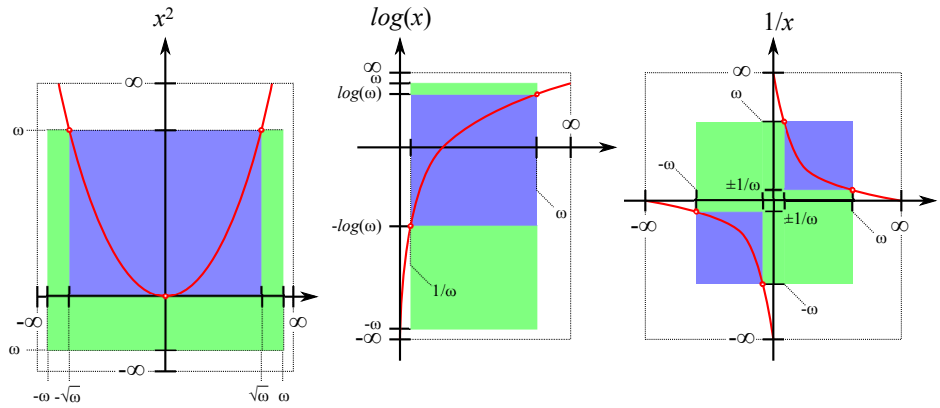

Figure 1.3: Refined Domains of Primitive Functions

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

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

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

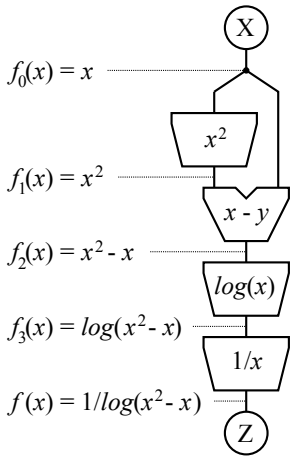

Figure 1.4: Parse Tree of $Z = 1/log(X^2 - X)$

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

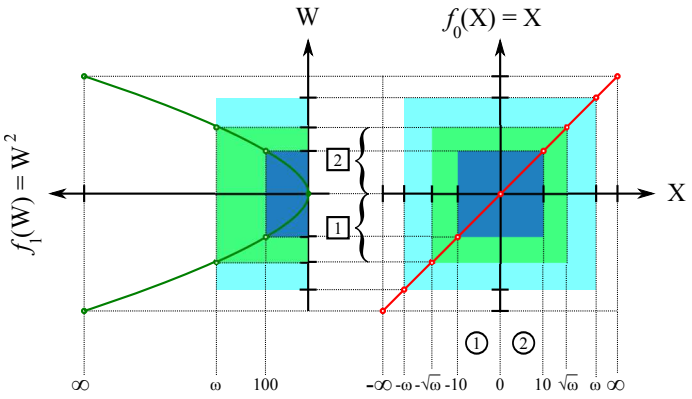

Figure 1.5: Refinement $f_0(X)$ by $f_1(X)$

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 $\boxed{1} = (-\sqrt{\omega}, 0)$, $\boxed{2} = (0, \sqrt{\omega})$, $\textcircled{1} = (-10, 0)$, $\textcircled{2} = (0, 10)$.

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 $\{\textcircled{1}, \textcircled{2}\}$ 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,

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

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

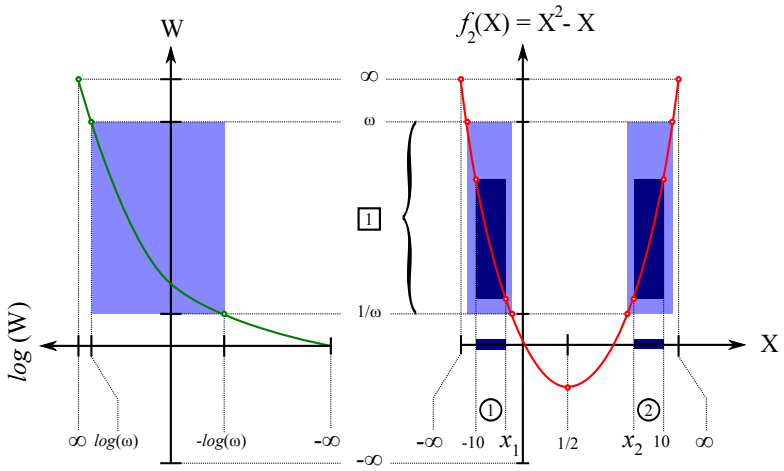

Figure 1.6: Domain of $f_2(X)$

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 }.

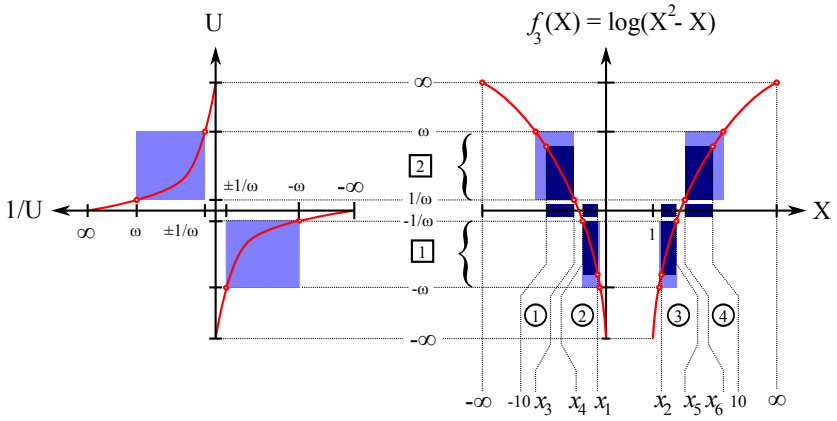

Figure 1.7: Domain of $f_3(X)$

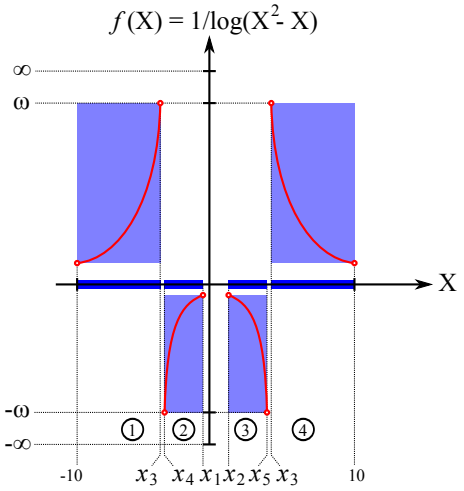

Figure 1.8: $Z = f(X)$ with Domain

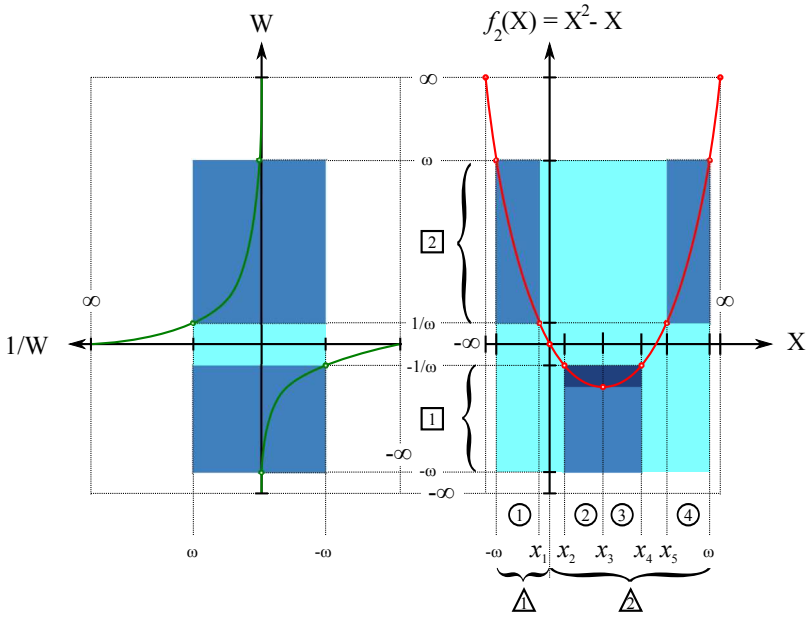

Figure 1.9: Refining $\frac{1}{x^2-x}$

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 in-

tersecting 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 $1/x$.

Appendix A

Exhaustive Bounded Root Finding Method

This section follows the *EveryRoot* method developed by Robert C. Tausworthe [3] 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.0.2 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.0.3 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 [1] or the much more recent Improved Brent's Method [4] 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.0.4 The *EveryRoot* Method

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

TO DO

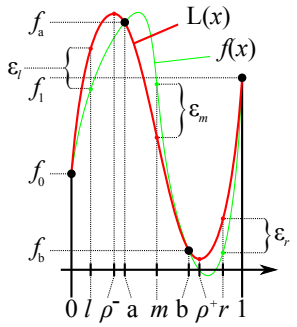

Figure A.1: Example Cubic Lagrange Interpolation

A.0.5 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 $\mathbf{f}$ 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 $\hat{\rho}$.

Bibliography

[1] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

[2] Sean Dineen. *Probability Theory in Finance: A Mathematical Guide to the Black-Scholes Formula*. Americal Mathematic Society, 2000.

[3] Robert C. Tausworthe. Finding every root of a broad class of real continuous functions in a given interval. *IPN Progress Report*, 42(176), 2009.

[4] Zhengqiu Zhang. An improvement to the brents method. *International Journal of Experimental Algorithms (IJEA)*, 2(1), 2011.