The Truncated Normal Distribution

John Burkardt
Department of Scientific Computing
Florida State University
https://people.sc.fsu.edu/~jburkardt/presentations/truncated_normal.pdf

12 September 2023

Abstract

The normal distribution is a common model of randomness. Unlike the uniform distribution, it proposes a most probable value which is also the mean, while other values occur with a probability that decreases in a regular way with distance from the mean. This behavior is mathematically very satisfying, and has an easily observed correspondence with many physical processes. One drawback of the normal distribution, however, is that it supplies a positive probability density to every value in the range $(-\infty, +\infty)$, although the actual probability of an extreme event will be very low. In many cases, it is desired to use the normal distribution to describe the random variation of a quantity that, for physical reasons, must be strictly positive. A mathematically defensible way to preserve the main features of the normal distribution while avoiding extreme values involves the *truncated normal distribution*, in which the range of definition is made finite at one or both ends of the interval. It is the purpose of this report to describe the truncation process, to consider how certain basic statistical properties of the new distribution can be determined, to show how to efficiently sample the distribution, and how to construct an associated quadrature rule, or even a sparse grid quadrature rule for a problem with multidimensional randomness.

Contents

1 The Standard Normal Distribution

1.1 Mathematical Definition

The standard normal distribution is a probability density function (PDF) defined over the interval $(-\infty, +\infty)$. The function is often symbolized as $\phi(0, 1; x)$. It may be represented by the following formula:

$$\phi(0, 1; x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Like any PDF associated with a continuous variable, $\phi(0, 1; x)$ may be interpreted to assert that the probability that an object x, randomly drawn from a group that obeys the standard normal distribution, will have a value that falls between the values a and b is:

$$\Pr(a \leq x \leq b) = \int_a^b \phi(0, 1; x)\, dx$$

1.2 The Mean and Variance

The *mean* of a distribution $\rho(x)$, symbolized by or $\text{mean}(\rho(*))$, may be thought of as the average over all values in the range. If we assume the range is (a, b), then it is defined as the following weighted integral:

$$\text{mean}(\rho()) = \int_a^b x\, \rho(x)\, dx$$

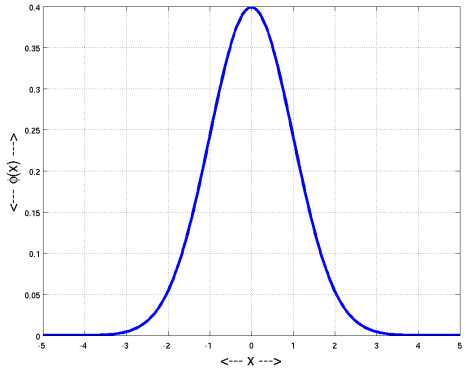

Figure 1: **The standard normal PDF**

Because the standard normal distribution is symmetric about the origin, it is immediately obvious that $\text{mean}(\phi(0, 1; *)) = 0$.

The *variance* of a distribution $\rho(x)$, symbolized by $\text{var}(\rho(*))$ is a measure of the average squared distance between a randomly selected item and the mean. Assuming the mean is known, the variance is defined as:

$$\text{var}(\rho(*)) = \int_a^b (x - \mu)^2 \, \rho(x) \, dx$$

For the standard normal distribution, we have that $\text{var}(\phi(0, 1; *)) = 1$.

Note that the *standard deviation* of any distribution, represented by $\text{std}(\rho(*))$, is simply the square root of the variance, so for the standard normal distribution, we also have that $\text{std}(\phi(0, 1; *)) = 1$.

1.3 The Cumulative Distribution Function

Recall that any probability density function $\rho(x)$ can be used to evaluate the probability that a random value falls between given limits a and b:

$$\Pr(a \leq x \leq b) = \int_a^b \rho(x) \, dx$$

Assuming that our values range over the interval $(-\infty, +\infty)$, we may define the function $F(\rho; b)$, the probability that a random value is less than or equal to b:

$$F(\rho; b) = \Pr(x \leq b) = \int_{-\infty}^b \rho(x) \, dx$$

If it is possible to evaluate $F(\rho; b)$ or to tabulate it at regular intervals, we can use this function to compute the probability of any interval, since

$$\Pr(a \leq x \leq b) = \int_a^b \rho(x) \, dx = F(\rho; b) - F(\rho; a)$$

A function like $F(\rho; x)$ is known as the *cumulative density function* or CDF for the corresponding PDF $\rho(x)$.

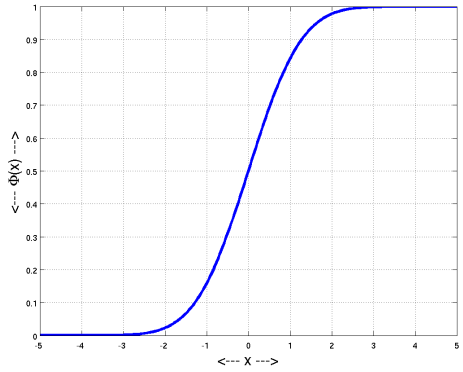

Figure 2: **The standard normal CDF**

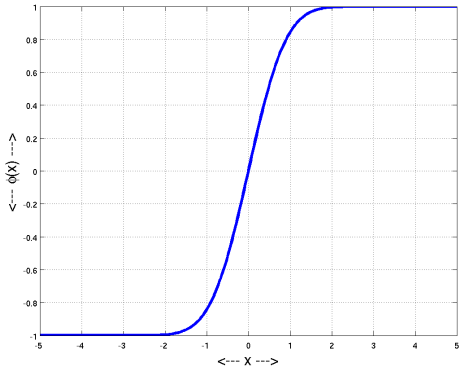

Figure 3: **The error function ERF**

In the case of the standard normal distribution, the CDF is denoted by $\Phi(0, 1; x)$, and is defined by

$$\Phi(0, 1; x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt$$

There is no simple formula to evaluate the normal CDF. Instead, there are extensive tables and computational algorithms. One common approach is based on a relationship with the *error function*. The error function, symbolized by **erf(x)**, is defined by

$$\text{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\, dt$$

Thus, the error function can be related to the CDF of the standard normal distribution:

$$\Phi(0, 1; x) = \frac{1}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))$$

so if an automatic procedure is available to evaluate $\text{erf}(x)$, it is easy to evaluate $\Phi(0, 1; x)$ as well. For instance, MATLAB has a built-in function **erf(x)** and Mathematica has **Erf[x]**.

Software for directly evaluating the standard normal CDF includes Algorithm AS 66 by David Hill[10].

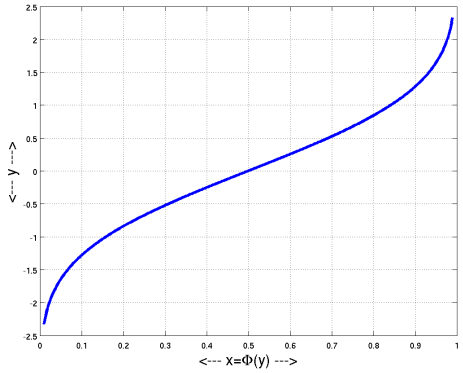

Figure 4: **The standard normal inverse CDF**

1.4 The Inverse Cumulative Distribution Function

Because the standard normal PDF is everywhere positive and integrable, it follows that the CDF $\Phi(0,1;x)$ is a strictly monotone function on $(-\infty,+\infty)$ which takes on, exactly once, every value in the open interval $(0,1)$. This implies the existence of an inverse cumulative density function (iCDF), denoted by $\Phi^{-1}(0,1;p)$, defined on $(0,-1)$ and returning values in $(-\infty,+\infty)$, such that

$$\begin{aligned}\Phi^{-1}(0,1;\Phi(0,1;x)) =& x\\ \Phi(0,1;\Phi^{-1}(0,1;p)) =& p\end{aligned}$$

The inverse CDF allows us to start with a probability $0 < p < 1$, and return a cutoff value $\Phi^{-1}(p) = x$, such that the probability of a value that is less than or equal to x is precisely p. We will see in a moment how access to such a function allows us to appropriately sample the density function.

In statistics, the inverse CDF of the normal distribution is sometimes referred to as the "percentage points" of the distribution.

Because of the relationship between the normal CDF and the error function, the inverse error function can be used to evaluate the iCDF. In particular, we have:

$$\begin{aligned}p = \Phi(0,1;x) =& \frac{1}{2}(1+\operatorname{erf}(\frac{b}{\sqrt{2}}))\\ 2p-1 =& \operatorname{erf}(\frac{x}{\sqrt{2}}))\\ \operatorname{erf}^{-1}(2p-1) =& \frac{b}{\sqrt{2}}\\ \sqrt{2}\operatorname{erf}^{-1}(2p-1) =& x\\ x = \Phi^{-1}(0,1;p) =& \sqrt{2}\operatorname{erf}^{-1}(2p-1)\end{aligned}$$

and thus, if we have access to an inverse error function, we can compute the inverse of the standard normal CDF as well.

Inverse error functions are available, for instance, in MATLAB as **erfinv()**, and in Mathematica as **InverseErf[]**.

Software to directly compute the inverse CDF of the standard normal distribution includes Applied Statistics Algorithm 111 by Beasley and Springer[1], Applied Statistics Algorithm 241 by Wichura[13], and the software package CDFLIB by Barry Brown, James Lovato, and Kathy Russell[2].

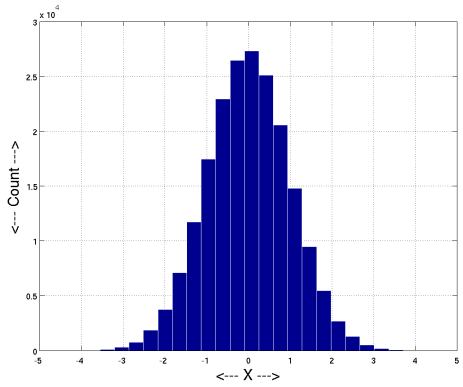

Figure 5: **200,000 sample values in 25 bins**

1.5 Sampling the Normal Distribution

Sampling a distribution means to select one item from the range of legal values, using the PDF as the probability of selection. A histogram of the selected data should roughly approximate the shape of a graph of the PDF.

Assuming we have some function **rand()** which is a source of uniform random numbers in the range $(0, 1)$, and that we have a means of evaluating $\Phi^{-1}(p)$, it is straightforward to sample the standard PDF as follows:

$$\begin{aligned} p =& \text{rand}() \\ x =& \Phi^{-1}(0, 1; p) \end{aligned}$$

1.6 Moments of the Standard Normal

The k-th moment of a PDF $\rho(x)$, which may be denoted $\mu'_k(\rho(*))$, is the weighted integral of x^k over the range of the PDF:

$$\mu'_k(\rho(*)) = \int_a^b x^k \, \rho(x) \, dx$$

In particular, $\mu'_0 = 1$ (because $\rho(*)$ is a PDF) and $\mu'_1 = \text{mean}(\rho(*))$, the mean value of the distribution.

Because the standard normal PDF is symmetric about the origin, all the moments of odd index are zero. The general formula is

$$\mu'_k(\phi(0, 1; *)) = \begin{cases} 0 & \text{if } k \text{ is odd;} \\ (k-1)!! = (k-1) \cdot (k-3) ... \cdot 3 \cdot 1 & \text{if } k \text{ is even.} \end{cases}$$

Here, the notation $(k-1)!!$ indicates the double factorial function.

1.7 Central Moments and the Variance

The k-th central moment of a PDF $\rho(x)$, which may be denoted $\mu_k(\rho(*))$, is the weighted integral of the difference $(x - \mu)^k$ over the range of the PDF:

$$\mu_k(\rho(*)) = \int_a^b (x - \mu)^k \, \rho(x) \, dx$$

In particular, $\mu_2(\rho(*)) = \mathrm{var}(\rho(*))$.

Because the standard normal distribution has zero mean, the central moments are the same as the moments, and so

$$\mu_k(\phi(0,1;*)) = \begin{cases} 0 & \text{if } k \text{ is odd;} \\ (k-1)!! = (k-1)\cdot(k-3)...\cdot 3\cdot 1 & \text{if } k \text{ is even.} \end{cases}$$

In particular, we note that $\mu_2(\phi(0,1;*)) = \mathrm{var}(\phi(0,1;*)) = 1$.

1.8 Quadrature Rule Computation

We expect to encounter integrals of the form

$$I(f) = \int_{-\infty}^{+\infty} f(x)\,\phi(0,1;x)\,dx$$

and we wish to be able to approximate such integrals by using a *quadrature rule*.

A quadrature rule for the normal PDF $\phi(0,1;x)$ is a set of n points x_i and weights w_i for which we can make the integral estimate:

$$\int_{-\infty}^{+\infty} f(x)\,\phi(0,1;x)\,dx = I(f) \approx Q(f) = \sum_{i=1}^{n} w_i \cdot f(x_i)$$

A quadrature rule is said to have *precision* k if $I(x^j) = Q(x^j)$ for integers $0 \le j \le k$. A quadrature rule with precision k will integrate exactly all polynomials of degree k or less. Quadrature rules may be roughly divided into simple interpolatory rules and Gaussian rules. An interpolatory rule of n points will typically achieve a precision of $n-1$ or perhaps n, while a Gaussian rule will achieve a precision of $2n-1$. Because of their greater precision, it is very useful to be able to construct a Gaussian quadrature rule for a given PDF. Algorithms for doing so were described by Golub and Welsch[7].

1.9 Orthogonal Polynomial Family

An algorithm for computing quadrature rules appropriate for the standard normal distribution requires the determination of an associated *orthogonal polynomial family*. This requires the determination of an indexed family of polynomials $p_i(x), i = 0, ...$ which are orthogonal with respect to the PDF.

To explain what is going on, let us suppose that we use our PDF to define a new inner product of any two functions $f()$ and $g()$ by:

$$< f, g > \equiv \int_{-\infty}^{+\infty} f(x)\,g(x)\,\phi(0,1;x)\,dx$$

We can define a corresponding function norm:

$$||f|| = \sqrt{< f, f >}$$

and we will restrict our attention to the space of all functions whose norm is finite.

If we can get a basis for this space, we know a lot about how it works. It is natural to analyze functions in terms of polynomials. A family of orthogonal polynomials $p_i(x)$ with respect to a given PDF is exactly an orthogonal basis for the given space, so that:

$$\int_{-\infty}^{+\infty} p_i(x)\,p_j(x)\,\phi(0,1;x)\,dx = \delta_{i,j}$$

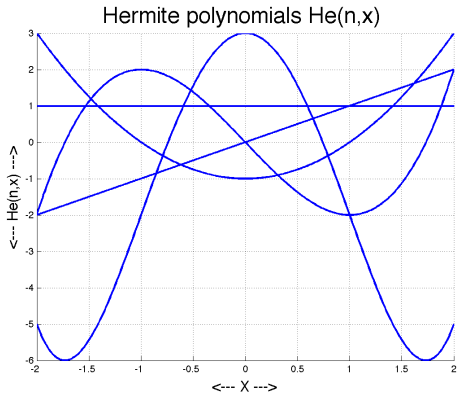

Figure 6: **The first 5 Hermite polynomials.**

(This formula assumes that I have taken one more step, and divided each basis function by its norm.)

The basis polynomials give a way of understanding all the elements of the space, and are similar to thinking of the column vectors of the identity matrix as a basis for all vectors in $\mathbb{R}^n$.

For the normal distribution, the orthogonal family is the *Hermite polynomials*, whose first elements are:

$$\begin{aligned} H_0(x) &= 1 \\ H_1(x) &= x \\ H_2(x) &= x^2 - 1 \\ H_3(x) &= x^3 - 3x \end{aligned}$$

Orthogonal polynomial families must satisfy a three term recurrence, based on coefficients $\{a_j, b_j, c_j\}$, which Golub and Welsch write as:

$$\begin{aligned} p_{-1}(x) &= 0 \\ p_0(x) &= 1 \\ p_j(x) &= (a_j\, x + b_j)\, p_{j-1}(x) - c_j\, p_{j-2}(x) \end{aligned}$$

For the Hermite polynomials, we have $a_j = 1$, $b_j = 0$ and $c_j = j - 1$, so that

$$\begin{aligned} H_{-1}(x) &= 0 \\ H_0(x) &= 1 \\ H_1(x) &= x\, H_0(x) - 0\, H_{-1}(x) = x \\ H_2(x) &= x\, H_1(x) - 1\, H_0(x) = x^2 - 1 \\ H_3(x) &= x\, H_2(x) - 2\, H_1(x) = x^3 - 3x \\ H_4(x) &= x\, H_3(x) - 3\, H_2(x) = x^4 - 6x^2 + 3 \\ &\text{...and so on} \end{aligned}$$

1.10 The Golub Welsch Procedure

We now define two indexed sets of quantities $\{\alpha_j, \beta_j\}$, determined from the recurrence coefficients of the orthogonal polynomial family:

$$\alpha_j = -\frac{b_j}{a_j}$$
$$\beta_j = \sqrt{\frac{c_{j+1}}{a_j a_{j+1}}}$$

From this information, the Golub-Welsch procedure now forms what is known as the *Jacobi matrix*.

$$J = \begin{pmatrix} \alpha_0 & \sqrt{\beta_1} & 0 & ... & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & ... & 0 \\ 0 & \sqrt{\beta_2} & \alpha_2 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \alpha_{n-1} \end{pmatrix}$$

It is then necessary to perform the eigen-decomposition of this symmetric tridiagonal matrix:

$$JX = X\Lambda$$

The diagonal values λ_i of Λ are the eigenvalues of J; these give us the points x_i of the quadrature rule; the weights w_i are computed from the first row of the normalized eigenvectors in X.

Software to carry out computations for this and several other PDF's is available in Elhay and Kautsky [4].

1.11 Quadrature Example

The Gaussian quadrature rule developed for the PDF associated with the Hermite orthogonal polynomial family is often termed a *Gauss-Hermite quadrature rule.* The software provided by Elhay and Kautsky frames the integration problem without the scaling factor needed to guarantee that the PDF integrates to 1. Thus, their typical integration problem is

$$I(f) = \int_{-\infty}^{+\infty} f(x)\, e^{-\frac{x^2}{2}}\, dx$$

and is thus "missing" the factor $\frac{1}{\sqrt{2\pi}}$. This means that, if we wish to adjust an Elhay-Kautsky quadrature rule so that it estimates integrals involving the standard normal PDF exactly as we have described it, we must multiply each quadrature weight by this missing factor.

For example, the Elhay-Kautsky procedure will return the following values for the 7 point Gauss-Hermite quadrature rule:

I	W	X
1	0.001374306216480	-3.750439717725742
2	0.077096676583483	-2.366759410734541
3	0.601899548885594	-1.154405394739968
4	1.145887211259887	0.000000000000000
5	0.601899548885594	1.154405394739968
6	0.077096676583483	2.366759410734541
7	0.001374306216480	3.750439717725742

If we suppose our integrand is the function $f(x) \equiv 1$, then $Q(f) = \sum_1^7 w_i f(x_i) = \sum_1^7 w_i \approx 2.5066 \approx \sqrt{2\pi}$. So it is obvious that, as a general rule, the weights of a quadrature rule associated with a PDF should sum to 1. Our corrected rule would be:

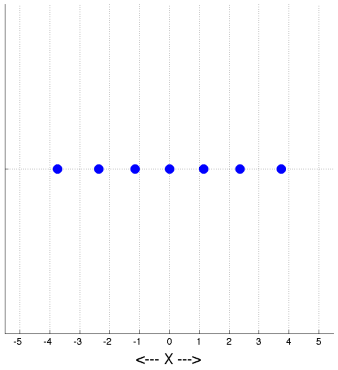

Figure 7: **The grid for the 7 point Gauss-Hermite rule.**

I	W	X
1	0.000548268855972	-3.750439717725742
2	0.030757123967586	-2.366759410734541
3	0.240123178605013	-1.154405394739968
4	0.457142857142858	0.000000000000000
5	0.240123178605013	1.154405394739968
6	0.030757123967586	2.366759410734541
7	0.000548268855972	3.750439717725742

Using this rule, if we seek an estimate of the integral

$$I(f) = \int_{-\infty}^{+\infty} \sqrt{1 + x^2/2}\, \phi(0, 1; x)\, dx$$

our result is 1.200108275; MATLAB includes a function called **integral()** that can estimate integrals over an infinite interval. It returns the value 1.200346935, but requires 210 function evaluations for this estimate.

1.12 Product Rules

Any quadrature rule for a one-dimensional interval $[a, b]$ can be used to construct a product rule for the corresponding product region $[a, b] \times [a, b]$. Such a product rule can be used to estimate corresponding integrals of a function $f(x, y)$ over the product region.

For integrals involving a PDF, it is necessary to create a product PDF. For the standard normal, this comes about in a very natural way:

$$\begin{aligned}
\phi^2(0, 1; x, y) =& \phi(0, 1; x) \cdot \phi(0, 1; y) \\
=& \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} \\
=& \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}
\end{aligned}$$

so that our integration problem is

$$I(f) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)\, e^{-\frac{x^2+y^2}{2}}\, dx\, dy$$

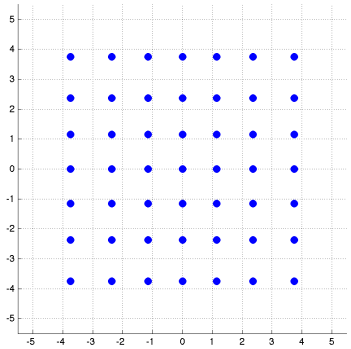

Figure 8: **The grid for the 7x7 point Gauss-Hermite product rule.**

and an n-point quadrature rule for this problem will have the form:

$$I(f) \approx Q(f) = \sum_{i=1}^{n} w_i f(x_i, y_i)$$

To construct a product rule for the 2D problem, we start with a 1D rule using n points with coordinates x_i and weights w_i. The new rule will use n^2 points, consisting of all possible pairings of two x values, weighted by the product of the corresponding w values. For example, if we used the 7 point Gauss-Hermite 1D rule to form a 49 point product rule for 2D, here is a partial list of the weights and points that would result:

I	W	X	Y
1	0.0000003005987384	-3.7504397177257420	-3.7504397177257420
2	0.0000168631731707	-3.7504397177257420	-2.3667594107345411
3	0.0001316520604261	-3.7504397177257420	-1.1544053947399679
4	0.0002506371913015	-3.7504397177257420	-0.0000000000000000
5	0.0001316520604261	-3.7504397177257420	1.1544053947399679
...	...	...	...
45	0.0001316520604261	3.7504397177257420	-1.1544053947399679
46	0.0002506371913015	3.7504397177257420	-0.0000000000000000
47	0.0001316520604261	3.7504397177257420	1.1544053947399679
48	0.0000168631731707	3.7504397177257420	2.3667594107345420
49	0.0000003005987384	3.7504397177257420	3.7504397177257420

The product rule procedure is quite flexible. Because we have used the same rule in both directions, we have created what might even be called a "power rule". However, in general, the product rule procedure allows us to have any number of spatial dimensions (2D, 3D, ...), to use an arbitrary interval in each dimension ($[a, b] \times [c, d] \times [e, f]$...), and to use different 1D quadrature rules in each dimension (7 point Gauss-Hermite $\times$ 11 point uniform $\times$ 5 point Gauss-Legendre...). as appropriate.

The main point is, however, that this means that once we have developed a procedure for producing one dimensional quadrature rules for the truncated normal distribution, we can automatically use such rules for problems in higher dimensions.

Suppose we take our 7 point 1D Gauss-Hermite rule and create a 2D product rule out of it. Then it can

be applied to estimate the following integral:

$$I(f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{1}{x^2 + y^2 + 1} \, \phi^2(0, 1; x, y) \, dx \, dy$$

our result is 0.4765506323; MATLAB includes a function called **integral2()** that can estimate integrals over a 2D infinite interval using an adaptive procedure. For our test integral, it returns the presumably superior estimate 0.4614554163, but requires 22,905 function evaluations to do so. Because our product rule is part of a family with increasing precision, and which has "knowledge" of the PDF built into the procedure, we may be confident that the 2D Gauss-Hermite estimates for the integral will rapidly produce accurate estimates for only a modest increase in the rule size from our initial 7x7 experiment.

1.13 Sparse Grid Rules

If we are interested in multivariate integration, then technically the product rule approach gives us everything we need. However, notice that the number of points used by a product rule involves raising the number of points in the 1D rule to the power of the dimension. While a 3 point rule in 1D looks cheap, in 10 dimension it needs almost 60,000 points and more than 3 billion points in 20 dimensions.

When modeling stochastic problems, each random variable corresponds to a dimension, and it's not uncommon for users to want 40 or 50 such variables. This would seem to rule out the use of standard quadrature rules, leaving nothing but Monte Carlo sampling to estimate the integrals.

The big advantage of quadrature rules is that they give very good integral estimates if the integrand is smooth. It turns out that the product rule approach is not the only way to get such high quality estimates; there is a *sparse grid* method, which is related to the product rule approach, but which gives good answers at a much cheaper cost.

To give an idea of of the efficiency of sparse grids, consider again our example of using a 3 point rule for 1D to construct rules for higher dimensions. The corresponding product rules in 10 and 20 dimensions required so many points as to be unusable. Starting from that same 3 point rule, a sparse grid will use 31 points in 10 dimensions, 61 points in 20 dimensions, and 121 points in 40 dimensions. Although even a sparse grid will run into limitations with sufficient increase in the size of the 1D rule or the spatial dimension, it is able to efficiently give integral estimates of known precision in dimensions that are unimaginable for the product rule.

There is an extensive theory for sparse grids. For this discussion, we will simply suggest how a sparse grid method can be constructed for the 2D case, the reader may assume that analogous procedures are available in general.

The simplest kind of sparse grid begins by designating a family of 1D quadrature rules, indexed in increasing order. The family should start with a 1 point rule. Typically, this is followed by a 3 point rule, but after that there are many variations as to how the number of points increases. For our example, the family will simply be the Gauss-Hermite rules of order 1, 3, 5, 7, 9, and so on. We will the symbol ℓ as the index for our 1D family; o to represent the number of points, and p to represent the precision. The beginning of a table of these values for our 1D family would be:

ℓ	o	p
0	1	1
1	3	5
2	5	9
3	7	13
4	9	17
5	11	21

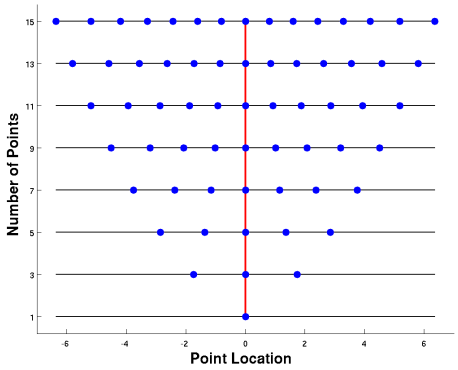

Figure 9: **An Indexed Family of Gauss-Hermite Rules.**

For later use in the example, let us note in particular the form of the Gauss Hermite rules of orders 1, 3, and 5:

I	W	X
1	1.0	0.0
1	$\frac{1}{6}$	$-\sqrt{3}$
2	$\frac{4}{6}$	0.0
3	$\frac{1}{6}$	$+\sqrt{3}$
1	0.0112574113277207	-2.856970013872805
2	0.2220759220056128	-1.355626179974265
3	0.5333333333333340	0.000000000000000
4	0.2220759220056128	1.355626179974265
5	0.0112574113277207	2.856970013872805

In our previous discussion of product rules, we mentioned that we are free to use different rules for different coordinates. For a 2D rule, we could consider combinations that we might symbolize by pairs of ℓ values, which we might write as $Q^{\ell_1} \times Q^{\ell_2}$. In this notation, $Q^5 \times Q^2$ is the product rule using an 11-point rule in x and a 5-point rule in y, for a total of 55 points. If we represent a rule this way, we can speak of its ℓ-*sum*, that is, the sum of the ℓ indices of its components. Thus, the 2D rule we just discussed has an ℓ-sum of 7.

The sparse grid procedure for 2D creates a new rule of level L by adding all the rules with ℓ-sum equal to L and subtracting the rules with the ℓ-sum equal to $L-1$. To see what this means, let's look at the firs "interesting" sparse grid, when $L = 1$, in 2D ($D = 2$) which can be represented as

$$\mathcal{A}(L = 1, D = 2) = Q^0 \times Q^1 + Q^1 \times Q^0 - Q^0 \times Q^0$$

This says add the two product rules with ℓ-sum 1, and subtract the product rule with ℓ-sum 0. We could interpret this statement as follows: *To get the $L = 1$ sparse grid approximation of an integral in $D = 2$ dimensions, compute the estimate from a 1x3 product grid, add the estimate from a 3x1 product grid, and subtract the estimate from a 1x1 product grid.* Let's do this for our example integrand function $\frac{1}{x^2+y^2+1}$.

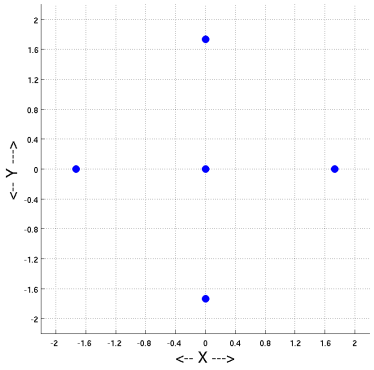

Figure 10: **The 5 point level 1 Gauss-Hermite sparse grid.**

i	w	x	y	f
1	1.0	0.0	0.0	1.0
1	$\frac{1}{6}$	$-\sqrt{3}$	0.0	0.25
2	$\frac{4}{6}$	0.0	0.0	1.00
3	$\frac{1}{6}$	$+\sqrt{3}$	0.0	0.25
1	$\frac{1}{6}$	0.0	$-\sqrt{3}$	0.25
2	$\frac{4}{6}$	0.0	0.0	1.00
3	$\frac{1}{6}$	0.0	$+\sqrt{3}$	0.25

Thus we get $\mathcal{A}(L = 1, D = 2) = 0.75 + 0.75 - 1.0 = 0.5$, which is not bad for a 7 point estimate, given that our 49 point estimate was 0.4765506323 and MATLAB's expensive estimate was 0.4614554163. If we are not satisfied with this initial estimate, we can proceed to the next sparse grid. We will see how to do that in a moment.

first let us notice something about our current calculation. We called this a 7 point rule, and it was, in the sense that it combined the results of rules of order 3, 3 and 1. However, notice that the point $(0.0, 0.0)$ occurred in every rule, and thus was counted three times. We could write this rule in a more compact form by listing all the points together. Since we subtract the results of some rules, we need to make their weights negative. And if a point occurs multiple times, we simply list it once, and combine its multiple weights into a single value.

Here is the equivalent description of our sparse grid rule of level $L = 1$, listing 5 points:

i	w	x	y	f
1	$\frac{2}{6}$	0.0	0.0	1.00
2	$\frac{1}{6}$	$-\sqrt{3}$	0.0	0.25
3	$\frac{1}{6}$	$+\sqrt{3}$	0.0	0.25
4	$\frac{1}{6}$	0.0	$-\sqrt{3}$	0.25
5	$\frac{1}{6}$	0.0	$+\sqrt{3}$	0.25

As the indexing suggests, the sparse grid rule of level $L = 1$ is simply one member of an indexed family of rules of increasing complexity and precision. The simplest member of the sequence is the rule for $L = 0$, which is the 1-point rule:

i	w	x	y
1	1.0	0.0	0.0

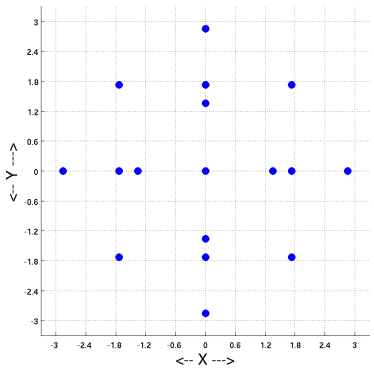

Figure 11: **The 17 point level 2 Gauss-Hermite sparse grid.**

The next rule is the $L = 2$ sparse grid, which can be described as the sum of all the grids with ℓ-sum 2 minus the grids with ℓ-sum 1:

$$\mathcal{A}(L = 2, D = 2) = Q^0 \times Q^2 + Q^1 \times Q^1 + Q^2 \times Q^0 - Q^0 \times Q^1 - Q^1 \times Q^0$$

If we wished to, we could compute all five of these product rules, use each one to approximate the integral, and then add the first three estimates and subtract the last two. As before, though, it may be simpler to combine the five rules into a single list of weights and points. The individual grids making up the level 2 sparse grid use 25 points; after we group the repeated points, we end up with 17 unique values.

The 17 point sparse grid level 2 rule looks like this:

i	w	x	y	f
1	0.0112574113277206	-2.856970013872805	0.000000000000000	0.109143166916601
2	0.0277777777777777	-1.732050807568878	-1.732050807568878	0.142857142857143
3	-0.0555555555555555	-1.732050807568878	0.000000000000000	0.250000000000000
4	0.0277777777777777	-1.732050807568878	1.732050807568878	0.142857142857143
5	0.2220759220056125	-1.355626179974265	0.000000000000000	0.352395294621861
6	0.0112574113277206	0.000000000000000	-2.856970013872805	0.109143166916601
7	-0.0555555555555555	0.000000000000000	-1.732050807568878	0.250000000000000
8	0.2220759220056125	0.000000000000000	-1.355626179974265	0.352395294621861
9	0.1777777777777780	0.000000000000000	0.000000000000000	1.000000000000000
10	0.2220759220056125	0.000000000000000	1.355626179974266	0.352395294621861
11	-0.0555555555555555	0.000000000000000	1.732050807568878	0.250000000000000
12	0.0112574113277206	0.000000000000000	2.856970013872805	0.109143166916601
13	0.2220759220056125	1.355626179974266	0.000000000000000	0.352395294621861
14	0.0277777777777777	1.732050807568878	-1.732050807568878	0.142857142857143
15	-0.0555555555555555	1.732050807568878	0.000000000000000	0.250000000000000
16	0.0277777777777777	1.732050807568878	1.732050807568878	0.142857142857143
17	0.0112574113277206	2.856970013872805	0.000000000000000	0.109143166916601

Using the 17-point level 2 rule, our integral estimate is 0.45604395604395598, which now compares favorably to the MATLAB estimate of 0.4614554163, and the Mathematica estimate of 0.461455316241865.

A simple chart may suggest how to think about this process, and how it will proceed from level to level, for the 2D case:

```
                                    +(0,3)
```

```
                          +(0,2)                -(0,2)+(1,2)
          +(0,1)          -(0,1)+(1,1)                -(1,1)+(2,1)
+(0,0)    -(0,0)+(1,0)          -(1,0)+(2,0)                -(2,0)+(3,0)

L=0         L=1               L=2                   L=3
```

The sparse grid with index L adds all the product rules with ℓ-sum equal to L, which we can think of as a diagonal line $\ell_1 + \ell_2 = L$, and subtracts the product rules on the previous diagonal line $\ell_1 + \ell_2 = L - 1$.

In 3D, the level L sparse grid combines all the product rules of ℓ-sum L, subtracts twice the product rules with ℓ-sum $L-1$ and adds with ℓ-sum $L-2$. The first level at which this pattern becomes clear is $L = 2$:

$$\begin{aligned}\mathcal{A}(L=2, D=3) =& Q^0 \times Q^0 \times Q^2 + Q^0 \times Q^1 \times Q^1 + Q^0 \times Q^2 \times Q^0 \\ &+ Q^1 \times Q^0 \times Q^1 + Q^1 \times Q^1 \times Q^0 + Q^2 \times Q^0 \times Q^0 \\ &- 2 \cdot Q^0 \times Q^0 \times Q^1 - 2 \cdot Q^0 \times Q^1 \times Q^0 - 2 \cdot Q^1 \times Q^0 \times Q^0 \\ &+ Q^0 \times Q^0 \times Q^0\end{aligned}$$

In higher dimensions, the procedure is more complicated, but still very regular, involving product rules combined with signed combinatorial coefficients. Assuming we have a family of 1D quadrature rules $\mathcal{Q}^\ell$ indexed by ℓ, We form a sparse grid $\mathcal{A}(L, D)$ for dimension $\mathbf{D}$ and level $\mathbf{L}$.

$\mathcal{A}(L, D)$ is a weighted sum of D-dimensional product rules

$$\mathcal{Q}^{\ell_1} \times \cdots \times \mathcal{Q}^{\ell_D}$$

The vector $\vec{\ell}$ lists the levels of the component rules used, and $|\vec{\ell}| = \ell_1 + \cdots + \ell_D$ is the sum.

$$\mathcal{A}(L, D) = \sum_{L-D+1 \le |\vec{\ell}| \le L} (-1)^{L-|\vec{\ell}|} \left(\begin{array}{c} D-1 \\ L-|\vec{\ell}| \end{array} \right) (\mathcal{Q}^{\ell_1} \times \cdots \times \mathcal{Q}^{\ell_D})$$

Thus, the rule $\mathcal{A}(L, D)$ is a weighted sum of D-dimensional product rules whose ℓ-sum or total level $|\vec{\ell}|$ never exceeds L.

Thus, if we are given an integral of a smooth function $f(x)$ of a multidimensional argument x, weighted by the multivariate normal PDF, we see that there is an automatic procedure for producing sparse grids of increasing level L, appropriate for estimating this integral to an increasing precision.

2 The General Normal Distribution

We will generalize the notion of the standard normal distribution, which has mean 0 and variance 1, to a general normal distribution with mean μ and standard deviation σ or variance σ^2. We would like to extend the use of the notations $\phi(*)$ and $\Phi(*)$ for the PDF and CDF respectively, but to do this, it is necessary to make the parameter dependence explicit. Thus, for the general distribution, we will write $\phi(\mu, \sigma; x)$ and $\Phi(\mu, \sigma; x)$. For the standard distribution, we may write $\phi(0, 1; x)$ and $\Phi(0, 1; x)$, or, if there is no danger of ambiguity, simply $\phi(x)$ and $\Phi(x)$.

2.1 Mathematical Definition

The general normal distribution is derived from the standard normal distribution. It is characterized by two parameters, μ and σ, which, we will shortly see, are in fact the mean and standard deviation. The function

may be symbolized as $\phi(\mu, \sigma; x)$ or $\mathcal{N}(\mu, \sigma)$. It may be represented by the following formula:

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

2.2 The Mean and Variance

Because the general normal distribution is symmetric about the parameter value $x = \mu$, it is immediately obvious that

$$\text{mean}(\phi(\mu, \sigma, *)) = \mu$$

To determine the variance, we note that, by a change of variable, the equation defining the variance of the general normal distribution can be rewritten as a multiple of the variance of the standard normal:

$$\begin{aligned}
\text{var}(\phi(\mu, \sigma; *)) =& \int_{-\infty}^{+\infty} (x-\mu)^2\, \phi(\mu, \sigma^2; x)\, dx \\
=& \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} (x-\mu)^2\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \\
=& \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \sigma^2 \cdot \xi^2\, e^{-\frac{\xi^2}{2}}\, \sigma d\xi \\
=& \frac{\sigma^2}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \xi^2\, e^{-\frac{\xi^2}{2}}\, d\xi \\
=& \sigma^2 \cdot \text{var}(\phi(0, 1; *)) \\
=& \sigma^2
\end{aligned}$$

Thus, for the general normal distribution, the square of the parameter σ is the variance.

2.3 Mapping to and from the Standard Normal

There is a simple transformation that relates variables governed by the standard and general normal distributions. Suppose that the variables ξ and x are characterized by the standard and general distributions, respectively. Then the following linear correspondences hold between the two variables:

$$\begin{aligned}
\xi =& \frac{x-\mu}{\sigma} \\
x =& \mu + \sigma\xi
\end{aligned}$$

This fact means that we can use a general normal distribution to describe any problem, while we can transform it at any time back to a standard normal distribution. This allows us to create a single set of tables and tests for the standard PDF that can be used for all the general cases as well.

2.4 The Cumulative Distribution Function

For the general normal distribution, we simply have

$$\Phi(\mu, \sigma; x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\, dt$$

In cases where we wish to evaluate the CDF $\Phi(\mu, \sigma; x)$, we need to use the transformation that maps the argument x to the corresponding argument ξ associated with the standard normal distribution. In other

words, if we write $\xi(\mu, \sigma; x)$ to represent the transformation from the general to the standard argument, we have:

$$\begin{aligned}\Phi(\mu, \sigma; x) =& \Phi(0, 1; \frac{x-\mu}{\sigma}) \\ =& \Phi(0, 1; \xi(\mu, \sigma; x)) \\ =& \frac{1}{2}(1 + \operatorname{erf}(\frac{\xi}{\sqrt{2}}))\end{aligned}$$

Software for directly evaluating the standard normal CDF includes Algorithm AS 66 by David Hill[10].

2.5 The Inverse Cumulative Distribution Function

Given a probability $0 < p < 1$, we have already seen how to compute the corresponding value ξ associated with the standard normal distribution. Therefore, to determine the corresponding value x associated with a general normal distribution, we need only to apply the linear transformation from ξ to x.

The resulting formula is:

$$\begin{aligned}x =& \mu + \sigma\xi \\ =& \mu + \sigma\Phi^{-1}(0, 1; p) \\ =& \mu + \sigma\sqrt{2}\operatorname{erf}^{-1}(2p - 1) \\ =& \Phi^{-1}(\mu, \sigma; p)\end{aligned}$$

In statistics, the inverse CDF of the normal distribution is sometimes referred to as the "percentage points" of the distribution.

2.6 Sampling the General Normal Distribution

If we wish to sample from a general normal distribution with parameters μ and σ, we use the appropriate transform:

$$\begin{aligned}p =& \text{rand}() \\ x =& \mu + \sigma \cdot \Phi^{-1}(0, 1; p)\end{aligned}$$

2.7 Moments of the General Normal Distribution

The k-th moment of the general normal PDF $\phi(\mu, \sigma; *)$,

$$\mu'_k(\phi(\mu, \sigma; *)) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} x^k \, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \, dx$$

We know that $\mu'_0 = 1$ and $\mu'_1 = \mu$

The following formula for the higher moments is based on Cook [3].

$$\begin{aligned}\mu'_k =& E\{x^k\} \\ =& E\{(\mu+\sigma\xi)^k\} \\ =& \sum_{i=0}^{k} \binom{k}{i} E(\xi^i)\,\sigma^i\,\mu^{k-i} \\ =& \sum_{j=0}^{\lfloor k/2\rfloor} \binom{k}{2j} (2j-1)!!\,\sigma^{2j}\,\mu^{k-2j}\end{aligned}$$

Here,

- $\lfloor k/2 \rfloor$ is the *floor()* function, which rounds down to an integer;
- $(2j-1)!!$ is using the *double factorial function* which here is simply the product of the odd numbers from 1 to $2j-1$;
- $\binom{k}{2j}$ is the combinatorial coefficient, here $\frac{k!}{(2j)!(k-2j)!}$.

2.8 Central Moments of the General Normal Distribution

For the central moments, we have:

$$\begin{aligned}\mu_k =& E\{(x-\mu)^k\} \\ =& E\{(\sigma\xi)^k\} \\ =& E(\xi^k)\,\sigma^k \\ =& \begin{cases} 0 & \text{if } k \text{ is odd;} \\ (k-1)!!\,\sigma^k & \text{if } k \text{ is even.}\end{cases}\end{aligned}$$

2.9 Quadrature Rules, Product Rules, Sparse Grid Rules

Suppose that we have an n-point quadrature rule, represented by (ξ, w), using points ξ_i and weights w_i, suitable for approximating integrals weighted by the standard normal pdf:

$$\int_{-\infty}^{+\infty} f(\xi)\,\phi(0,1;\xi)\,d\xi = I(f) \approx Q(f) = \sum_{i=1}^{n} w_i \cdot f(\xi_i)$$

Then we may apply the change of variable $x = \mu + \sigma\xi$ and write

$$\begin{aligned}I(f) =& \int_{-\infty}^{+\infty} f(x)\,\phi(\mu,\sigma;x)\,dx \\ =& \int_{-\infty}^{+\infty} f(\mu+\sigma\xi)\,\phi(0,1;\xi)\,\sigma d\xi \\ =& \sigma\cdot\int_{-\infty}^{+\infty} f(\mu+\sigma\xi)\,\phi(0,1;\xi)\,d\xi \\ \approx& \sum_{i=1}^{n} \sigma\cdot w_i\cdot f(\mu+\sigma\xi_i)\end{aligned}$$

Hence, a corresponding n-point rule (X, W) for the general normal PDF will use points $X_i = \mu + \sigma \cdot \xi_i$, and weights $W_i = \sigma \cdot w_i$.

Thus there is no need to develop a new theory of quadrature rules for the general normal PDF. Moreover, this means that product rules and sparse grid rules can be constructed in a way that is completely analogous to the procedures developed for the standard normal PDF.

We have now completed our discussion of the standard and general normal PDF's, which will we need to refer to repeatedly as we begin our main topic, the truncated normal PDF.

3 The Truncated Normal Distribution

3.1 Mathematical Definition

Informally, the truncated normal probability density function is defined in two steps. We choose a general normal PDF by specifying parameters μ and σ, and then a truncation range (a, b). The PDF associated with the general normal distribution is modified by setting values outside the range to zero, and uniformly scaling the values inside the range so that the total integral is 1.

Depending on the truncation range, we have four cases:

- the nontruncated case: $-\infty = a, b = +\infty$;
- the lower truncated case: $-\infty < a, b = +\infty$;
- the upper truncated case: $-\infty = a, b < +\infty$;
- the doubly truncated case: $-\infty < a, b < +\infty$;

Formally, the truncated normal probability density function will be symbolized by $\psi(\overline{\mu}, \overline{\sigma}, a, b; x)$, where

- $\overline{\mu}$ and $\overline{\sigma}$ are the mean and standard deviation of the "parent" general normal PDF;
- a and b specify the truncation interval.

The PDF may be evaluated by the formula:

$$\psi(\overline{\mu}, \overline{\sigma}, a, b; x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{\phi(\overline{\mu}, \overline{\sigma}; x)}{\Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)} & \text{if } a < x < b \\ 0 & \text{if } b \leq x \end{cases}$$

The parameters $\overline{\mu}$ and $\overline{\sigma}$ are the mean and standard deviation of the parent general normal PDF; they do not represent the mean and standard deviation of the truncated normal PDF itself! The calculation of those quantities will be considered shortly.

As an example, we evaluate $\psi(\overline{\mu}, \overline{\sigma}, a, b; x)$ for $\overline{\mu} = 100$, $\overline{\sigma} = 25$, $a = 50$, $b = 150$ for random x arguments:

x	$\psi(\overline{\mu}, \overline{\sigma}, a, b; x)$
81.63	0.0127629
137.962	0.00527826
122.367	0.0112043
103.704	0.0165359
94.899	0.016374
65.8326	0.00657044
84.5743	0.0138204
71.5672	0.00875626
62.0654	0.00528716
108.155	0.0158521

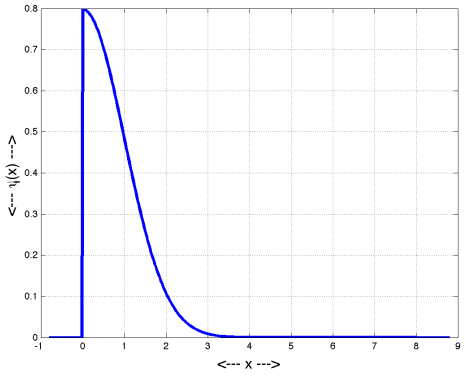

Figure 12: $\psi(0, 1, a = 0, b = +\infty; x)$.

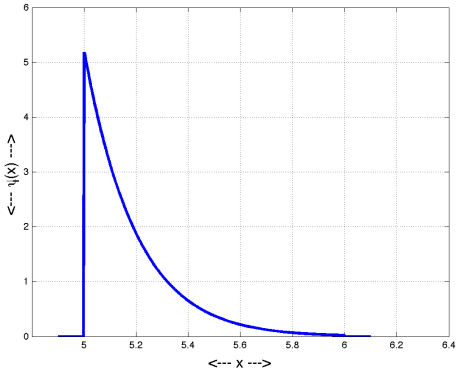

Figure 13: $\psi(0, 1, a = 5, b = 6; x)$.

3.2 Effect of the Truncation Range

All general normal PDF's are related to the standard normal PDF by a scaling and shifting, and in some sense, all the PDF's have essentially the same shape.

In a typical use, a truncated normal distribution might be defined by a symmetric truncation interval that extends five standard deviations to either side of the mean value; in this case the graph would look very similar to a general normal PDF, and the scaling would be so mild that the function values, the mean and the variance would also be close.

But, restricting ourselves to the case where the truncated normal distribution is derived from the standard normal distribution, let us simply consider a few examples of the variety of distributions that can result by various choices of the truncation range.

3.3 The Cumulative Density Function

The cumulative density function $\Psi(\overline{\mu}, \overline{\sigma}, a, b; x)$ is simply the integral of the PDF up to the argument x:

$$\Psi(\overline{\mu}, \overline{\sigma}, a, b; x) = \int_a^x \psi(\overline{\mu}, \overline{\sigma}, a, b; t)\, dt.$$

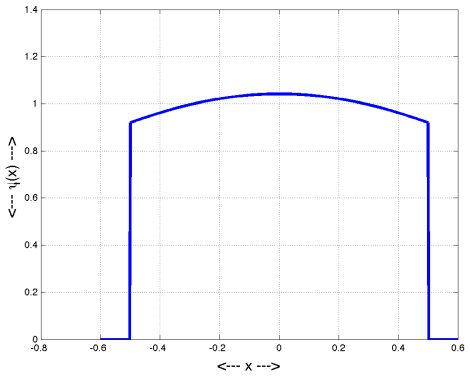

Figure 14: $\psi(0, 1, a = -0.5, b = +0.5; x)$.

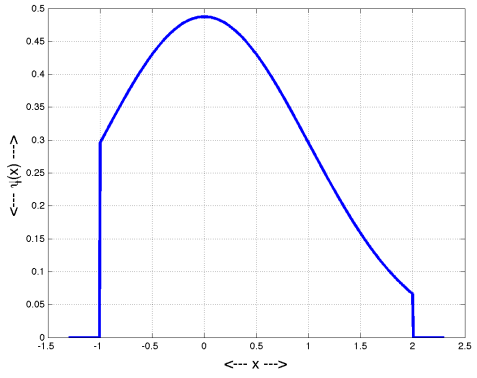

Figure 15: $\psi(0, 1, a = -1, b = +2; x)$.

We can simply integrate the known formula for $\psi()$, assuming that $a < x < b$:

$$\begin{aligned}\Psi(\overline{\mu},\overline{\sigma},a,b;x) &= \int_{-\infty}^{x} \psi(\overline{\mu},\overline{\sigma},a,b;t)\,dt \\ &= \int_{a}^{x} \psi(\overline{\mu},\overline{\sigma},a,b;t)\,dt \\ &= \int_{a}^{x} \frac{\phi(\overline{\mu},\overline{\sigma};t)}{\Phi(\overline{\mu},\overline{\sigma};b)-\Phi(\overline{\mu},\overline{\sigma};a)}dt \\ &= \frac{\int_{a}^{x}\phi(\overline{\mu},\overline{\sigma};t)dt}{\Phi(\overline{\mu},\overline{\sigma};b)-\Phi(\overline{\mu},\overline{\sigma};a)} \\ &= \frac{\Phi(\overline{\mu},\overline{\sigma};x)-\Phi(\overline{\mu},\overline{\sigma};a)}{\Phi(\overline{\mu},\overline{\sigma};b)-\Phi(\overline{\mu},\overline{\sigma};a)}\end{aligned}$$

In summary, we have

$$\Psi(\overline{\mu},\overline{\sigma},a,b;x) = \begin{cases} 0 & \text{if } x \le a \\ \frac{\Phi(\overline{\mu},\overline{\sigma};x)-\Phi(\overline{\mu},\overline{\sigma};a)}{\Phi(\overline{\mu},\overline{\sigma};b)-\Phi(\overline{\mu},\overline{\sigma};a)} & \text{if } a < x < b \\ 1 & \text{if } b \le x \end{cases}$$

from which result it should be immediately clear that $\Psi()$ is 0 at a and 1 at b, and, for values of x inbetween, is simply a scaled and shifted version of the behavior of $\Phi()$.

If it is desired to numerically evaluate $\Psi()$, the formula above shows us that we may do so by evaluating $\Phi()$, and the means of doing this has been discussed earlier.

As an example, we evaluate $\Psi(\overline{\mu},\overline{\sigma},a,b;x)$ for $\overline{\mu} = 100$, $\overline{\sigma} = 25$, $a = 50$, $b = 150$ for random x arguments:

x	$\Psi(\overline{\mu},\overline{\sigma},a,b;x)$
81.63	0.218418
137.962	0.956318
122.367	0.829509
103.704	0.561695
94.899	0.415307
65.8326	0.0661187
84.5743	0.257578
71.5672	0.109957
62.0654	0.043829
108.155	0.633966

3.4 The Inverse Cumulative Density Function

We suppose that we are given a value $0 \le p \le 1$ which represents the result of an evaluation of $\Psi(\overline{\mu},\overline{\sigma},a,b;x)$, and we seek $a \le x \le b$ satisfying:

$$p = \Psi(\overline{\mu},\overline{\sigma},a,b;x)$$

However, we have

$$\begin{aligned} p =& \Psi(\overline{\mu}, \overline{\sigma}, a, b; x) \\ =& \frac{\Phi(\overline{\mu}, \overline{\sigma}; x) - \Phi(\overline{\mu}, \overline{\sigma}; a)}{\Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)} \\ \Phi(\overline{\mu}, \overline{\sigma}; x) =& \Phi(\overline{\mu}, \overline{\sigma}; a) + p \cdot (\Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)) \\ x =& \Phi^{-1}(\overline{\mu}, \overline{\sigma}; \Phi(\overline{\mu}, \overline{\sigma}; a) + p \cdot (\Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a))) \end{aligned}$$

which we may summarize as

$$\Psi^{-1}(\overline{\mu}, \overline{\sigma}, a, b; p) = \Phi^{-1}(\overline{\mu}, \overline{\sigma}; \Phi(\overline{\mu}, \overline{\sigma}; a) + p \cdot (\Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)))$$

As an example, we evaluate $\Psi^{-1}(\overline{\mu}, \overline{\sigma}, a, b; x)$ for $\overline{\mu} = 100$, $\overline{\sigma} = 25$, $a = 50$, $b = 150$ for random values of $\Psi(\overline{\mu}, \overline{\sigma}, a, b; x)$:

$\Psi(\overline{\mu}, \overline{\sigma}, a, b; x)$	$\Psi^{-1}(\overline{\mu}, \overline{\sigma}, a, b; x)$
0.218418	81.63
0.956318	137.962
0.829509	122.367
0.561695	103.704
0.415307	94.899
0.0661187	65.8326
0.257578	84.5743
0.109957	71.5672
0.043829	62.0654
0.633966	108.155

3.5 Sampling the Truncated Normal Distribution

To sample from a truncated normal distribution $\psi(\overline{\mu}, \overline{\sigma}, a, b; *)$ we simply generate a uniform random value and apply the inverse CDF function:

$$\begin{aligned} p =& \text{rand}() \\ x =& \Psi^{-1}(\overline{\mu}, \overline{\sigma}, a, b; p) \end{aligned}$$

An alternative approach has been suggested for sampling the truncated normal distribution, which assumes that the inverse CDF for the general normal distribution is available, and imposes the truncation interval by rejection:

$$\begin{aligned} &\text{repeat:} \\ &\qquad p = \text{rand}() \\ &\qquad x = \Phi^{-1}(\overline{\mu}, \overline{\sigma}; p) \\ &\text{until } (a \leq x \leq b) \end{aligned}$$

This construction has the advantage that it does not require determining the inverse of the truncated normal. Quick acceptance is likely if the truncation interval includes most of the probability mass of the original normal distribution; for example, if (a, b) is $(-5\overline{\sigma}, +5\overline{\sigma})$. However, the truncation interval might represent a very small part of the normal probability mass, as when (a, b) is $(+5\overline{\sigma}, +\infty)$. In such a case, the acceptance/rejection approach must evaluate and reject thousands of iterates for each acceptance. By

contrast, the direct approach is only slightly more complex computationally, and is guaranteed to generate a new sample for every random number generated, regardless of what truncation interval is used.

As an example, we compute 1,000 samples from $\Psi(\overline{\mu}, \overline{\sigma}, a, b; x)$ for $\overline{\mu} = 100$, $\overline{\sigma} = 25$, $a = 50$, $b = 150$, and compare the sample mean and variance to the PDF mean and variance. (We will discuss how to determine the PDF mean and variance shortly.)

Source	μ	σ^2
PDF	100.0	483.588
Sample	100.123	486.064

3.6 The Mean

The mean μ of the truncated normal distribution can be regarded as a perturbation of the mean $\overline{mu}$ of the parent normal distribution. Its value can be determined by referencing the normal PDF ϕ and CDF Φ, as presented in Johnson [11]. First, define:

$$\alpha = \frac{a - \overline{\mu}}{\overline{\sigma}}; \quad \beta = \frac{b - \overline{\mu}}{\overline{\sigma}};$$

Then we have:

$$\mu = \overline{\mu} - \overline{\sigma} \cdot \frac{\phi(0,1;\beta) - \phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)}$$

3.7 The Variance

The variance σ^2 of the truncated normal distribution can also be regarded as a perturbation of the variance $\overline{\sigma}^2$ of the parent normal distribution. A formula for evaluating it is presented in Johnson [11]. Defining α and β as above, we have:

$$\sigma^2 = \overline{\sigma}^2 \cdot (1 - \frac{\beta\phi(0,1;\beta) - \alpha\phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)} - (\frac{\phi(0,1;\beta) - \phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)})^2)$$

3.8 Moments

The computation of moments is crucial for the Golub-Welsch method of determining quadrature rules associated with the truncated normal distribution.

The k-th moment of the truncated normal distribution is the quantity

$$\mu'_k = \int_a^b x^k \, \psi(\overline{\mu}, \overline{\sigma}, a, b; x) \, dx$$

As earlier, define:

$$\alpha = \frac{a - \overline{\mu}}{\overline{\sigma}}; \quad \beta = \frac{b - \overline{\mu}}{\overline{\sigma}};$$

Then (as before) we have

$$\mu'_k = \sum_{i=0}^{k} \binom{k}{i} \overline{\sigma}^i \, \overline{\mu}^{k-i} \, L_i$$

where the quantity L_i satisfies the recursion:

$$\begin{aligned} L_0 &= 1 \\ L_1 &= -\frac{\phi(0,1;\beta) - \phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)} \\ L_i &= -\frac{\beta^{i-1}\phi(0,1;\beta) - \alpha^{i-1}\phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)} + (i-1)L_{i-2} \end{aligned}$$

And thus we may evaluate any moment μ'_k.

As a sample calculation, let us set the parameters of the upper truncated normal distribution to $\overline{\mu} = 5$, $\overline{\sigma} = 1$, $a = -\infty$, $b = 10$, and compare the moment integrals μ'_k returned by our formula to values from Mathematica:

k	Formula	Mathematica
0	1	1
1	5	5
2	26	26
3	140	140
4	777.997	777.997
5	4449.97	4449.97
6	26139.69	26139.67
7	157396.75	157396.71
8	969946.73	969946.45

3.9 Central Moments

Although we will not need them here, the k-th central moment of the truncated normal distribution, given that the mean is μ, is the quantity

$$\mu_k = \int_a^b (x - \mu)^k \, \psi(\overline{\mu}, \overline{\sigma}, a, b; x) \, dx$$

Johnson [11] suggests that these moments can be evaluated in terms of the Hh functions of Fisher [5] [6], defined as:

$$\mathrm{Hh}_n(y) = \frac{1}{n!} \int_y^{+\infty} (1 - y)^n \, e^{-t^2} \, dt$$

3.10 Quadrature Rule Computation

The primary Golub and Welsch approach for computing an n-point quadrature rule for a PDF constructs the Jacobi matrix based on the recurrence coefficients of a family of orthogonal polynomials associated with the PDF. The details of this procedure were discussed earlier, in the context of the standard normal distribution, for which such a polynomial family is known.

For the truncated normal distribution, however, it is not obvious how to determine the appropriate polynomial family, and so the primary Golub-Welsch approach may not be suitable. However, Golub and Welsch include an alternative procedure, known as the *moment matrix method*, for computing the same Jacobi matrix J, without knowledge of an orthogonal polynomial family.

This alternative procedure requires constructing a matrix of moments, computing the Cholesky factorization, and then using elements of the Cholesky factorization to construct J.

The method begins by constructing the moment matrix M. Recall the notation previously defined, for which μ'_k represents the k-th moment of a distribution:

$$\mu'_k = \int_a^b x^k \, \mathrm{pdf}(x) \, dx$$

The $n+1$ by $n+1$ moment matrix is then:

$$M = \begin{pmatrix} \mu'_0 & \mu'_1 & \mu'_2 & ... & \mu'_n \\ \mu'_1 & \mu'_2 & \mu'_3 & ... & \mu'_{n+1} \\ \mu'_2 & \mu'_3 & \mu'_4 & ... & \mu'_{n+2} \\ ... & ... & ... & ... & ... \\ \mu'_n & \mu'_{n+1} & \mu'_{n+2} & ... & \mu'_{2n} \end{pmatrix}$$

The $n+1$ by $n+1$ upper triangular Cholesky factor R of the moment matrix M has the property that

$$M = R' \cdot R$$

This factorization exists assuming that M is symmetric and positive semidefinite.

Once we have the Cholesky factor matrix R, we compute a vector α of length n, and a vector β of length $n-1$. The algorithm for this step is suggested by the following MATLAB code:

```
alpha(1) = r(1,2) / r(1,1);
for i = 2 : n
  alpha(i) = r(i,i+1) / r(i,i) - r(i-1,i) / r(i-1,i-1);
end

for i = 1 : n - 1
  beta(i) = r(i+1,i+1) / r(i,i);
end
```

These vectors α and β are used to define the n by n Golub-Welsch matrix J, as suggested by this MATLAB code:

```
J = zeros ( n, n );

for i = 1 : n
  J(i,i) = alpha(i);
end

for i = 1 : n - 1
  J(i,i+1) = beta(i);
  J(i+1,i) = beta(i);
end
```

Now we assume we can determine the eigenvalues L and eigenvectors V of the symmetric tridiagonal matrix J. For example, MATLAB returns two matrices containing this information, after application of the **eig()** command:

```
[ V, L ] = eig ( J );
```

where the columns of V are eigenvectors and the diagonal of L contains the corresponding eigenvalues. If we have used MATLAB in this way, then the determination of the points x and weights w of the n point quadrature rule is now simple:

```
x = diag ( eval );
w = mu0 * evec(1,1:n).^2;
```

That is, the points are the eigenvalues, and the weights are the squares of the first entries of the corresponding eigenvectors, scaled by the μ_0, the integral of the PDF, which should be 1 by our default normalization.

This completes the alternate Golub-Welsch procedure for producing an n-point quadrature rule associated with a PDF for which an orthogonal polynomial family is not known.

We remark that, if MATLAB is not available, most of the calculation can be done easily in any computational language. If the programmer does not have access to a good eigenvalue package to decompose the matrix J, then it is not difficult to set up this calculation explicitly, because the J has a simple structure as a symmetric, tridiagonal matrix. This means that we can apply the standard Jacobi eigenvalue iteration [8] to produce excellent estimates of the eigenvalues and eigenvectors.

Using software that implements this algorithm, we computed various examples of quadrature rules. For example, given the parameters $\overline{\mu} = 2$, $\overline{\sigma} = 0.5$, $a = 0$, $b = +\infty$, the software computed the following 9 point rule:

i	w_i	x_i
1	0.000423602	0.181700
2	0.00977398	0.642168
3	0.0873219	1.13382
4	0.292167	1.62238
5	0.381303	2.10999
6	0.192723	2.60480
7	0.0345412	3.11888
8	0.00173333	3.67288
9	0.000012624	4.31747

3.11 Experiment with the Quadrature Rule

As a demonstration of the use of quadrature rules for the truncated normal distribution, let us consider the integrand $f(x) = \sin(x)$, and approximate the following integral:

$$I(f) = \frac{1}{S\sqrt{2\pi\overline{\sigma}^2}} \int_a^b \sin(x) e^{-\frac{(x-\overline{\mu})^2}{2\overline{\sigma}^2}} dx$$

where $S = \Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)$ is the usual normalization factor for the truncated normal PDF.

Let's use the lower truncated normal distribution, with parameters $\overline{\mu} = 0$, $\overline{\sigma} = 1$, $a = -3$, $b = +\infty$. Now we can compute quadrature rules of increasing order n, and compare the resulting integral estimates $Q(f)$ for the integral:

N	$Q(f)$
1	0.004437820
2	-0.002956940
3	0.000399622
4	-0.000236540
5	-0.000173932
6	-0.000177684
7	-0.000177529
8	-0.000177534
9	-0.000177534

For this same problem, Mathematica returns the value -0.000177531...

3.12 The Multidimensional Case

The simplest multidimensional generalization of the truncated normal distribution will suppose that we have a d-dimensional vector quantity $\vec{x}$, whose statistical distribution is governed by the following multivariate PDF in which all the parameters are scalars:

$$\rho(\overline{\mu}, \overline{\sigma}, a, b; \vec{x}) = \frac{1}{(S\sqrt{2\pi\overline{\sigma}^2})^d} e^{-\frac{\sum(\vec{x}-\overline{\mu})^2}{2\overline{\sigma}^2}}$$

where $S = \Phi(\overline{\mu}, \overline{\sigma}; b) - \Phi(\overline{\mu}, \overline{\sigma}; a)$ is the usual normalization factor for the 1D truncated normal PDF.

Assume we are given $f(\vec{x})$, a scalar function of a multivariate argument, and we desire to evaluate the integral of f over the product region $[a, b]^d$, weighted with the multivariate truncated normal distribution:

$$I(f) = \int_{[a,b]^d} f(\vec{x})\, \rho(\overline{\mu}, \overline{\sigma}, a, b; \vec{x})\, d\vec{x}$$

and we seek to do this by using a multivariate quadrature rule of the usual form:

$$I(f) \approx Q(f) = \sum_{i=1}^{n} w_i f(\vec{x}_i)$$

In cases like this, where the integration region is a product region, and the integration weight function is a product weight function, and 1D quadrature rules are available for each of the factor spaces, there is a standard method for producing a multivariate quadrature rule, called the *product rule*. If Q_i is a 1D quadrature rule suitable for the i-th factor space of the product region, then we may formally construct a multivariate quadrature rule as the product of 1D factor rules:

$$\mathcal{Q} = Q_1 \otimes Q_2 \otimes ... \otimes Q_d$$

Computationally, the rule $\mathcal{Q}$ uses $n_1 \cdot n_2 \cdot ... \cdot n_d$ points. The points in the rule are formed by taking all elements of the product space of the sets of 1D points; the weight of any multivariate point is simply the product of the weights associated with each of its 1D components. Note that if a product rule is formed by d copies of the same n-point rule, it uses n^d points, implying a severe limitation on applicability of product rules to problems that involve even moderate dimensionality and exactness.

3.13 Experiment with the Product Rule

As a demonstration of the use of a product rule formed from quadrature rules for the truncated normal distribution, let us consider the integrand $f(x, y) = \frac{1}{1+x^2+y^2}$, and approximate the following integral:

$$I(f) = \frac{1}{S^2 2\pi} \int_{-1}^{+1} \int_{-1}^{+1} \frac{1}{1 + x^2 + y^2} e^{-\frac{x^2+y^2}{2}} dx$$

where $S = \Phi(0, 1; +1) - \Phi(0, 1; -1)$ is the usual normalization factor for the (1D) truncated normal PDF.

We are using the doubly truncated normal distribution, with parameters $\overline{\mu} = 0$, $\overline{\sigma} = 1$, $a = -1$, $b = +1$. We can use **truncated_normal_rule** to compute a 5 point quadrature rule, and then apply the product rule procedure, perhaps as implemented in the **power_rule** program, to create a 5×5 rule appropriate for our two dimensional product region. The points and weights of this rule are:

I	W	X	Y
1	0.9755792102570658E-02	-0.8984499284579617	-0.8984499284579617
2	0.2392604934020298E-01	-0.8984499284579617	-0.5213172863125558
3	0.3140773050198433E-01	-0.8984499284579617	0.0
4	0.2392604934020296E-01	-0.8984499284579617	0.5213172863125560
5	0.9755792102570658E-02	-0.8984499284579617	0.8984499284579617
6	0.2392604934020298E-01	-0.5213172863125558	-0.8984499284579617
7	0.5867856049115529E-01	-0.5213172863125558	-0.5213172863125558
8	0.7702735992664954E-01	-0.5213172863125558	0.0
9	0.5867856049115525E-01	-0.5213172863125558	0.5213172863125560
10	0.2392604934020298E-01	-0.5213172863125558	0.8984499284579617
11	0.3140773050198433E-01	0.0	-0.8984499284579617
12	0.7702735992664954E-01	0.0	-0.5213172863125558
13	0.1011138331889368E+00	0.0	0.0
14	0.7702735992664948E-01	0.0	0.5213172863125560
15	0.3140773050198433E-01	0.0	0.8984499284579617
16	0.2392604934020296E-01	0.5213172863125560	-0.8984499284579617
17	0.5867856049115525E-01	0.5213172863125560	-0.5213172863125558
18	0.7702735992664948E-01	0.5213172863125560	0.0
19	0.5867856049115520E-01	0.5213172863125560	0.5213172863125560
20	0.2392604934020296E-01	0.5213172863125560	0.8984499284579617
21	0.9755792102570658E-02	0.8984499284579617	-0.8984499284579617
22	0.2392604934020298E-01	0.8984499284579617	-0.5213172863125558
23	0.3140773050198433E-01	0.8984499284579617	0.0
24	0.2392604934020296E-01	0.8984499284579617	0.5213172863125560
25	0.9755792102570658E-02	0.8984499284579617	0.8984499284579617

We compare a sequence of product rule estimates for this integral against the results returned from Matlab's quad2d() function and Mathematica's NIntegrate[] numerical integration function. Note that the Matlab and Mathematica functions are adaptive, and have a great deal of sophisticated programming in order to get an accurate answer cheaply.

Method	N	Q
tn01x01	1	1.0
tn03x03	9	0.6788136155
tn05x05	25	0.6719544360
tn07x07	49	0.6717845630
tn09x09	81	0.6717805532
tn11x11	121	0.6717801747
Matlab quad2d()	73	0.6717800326
Mathematica NIntegrate[]	?	0.6717800275

For high dimensional integration, product rules are very expensive (because the number of function evaluations grows exponentially) but very attractive (because they are easy to construct, easy to analyze, and have excellent convergence properties for smooth integrands).

As an attempt to extend the applicability of product rules to higher dimensions and degrees of exactness, sparse grid rules were developed. These rules seek to achieve the exactness of product rules at a lower cost. A surprising fact is that sparse grids are themselves constructed from product rules; however the component product rules are chosen very carefully in an attempt to avoid the explosion in the number of function evaluations required.

3.14 Definition of a Sparse Grid Rule

In order to implement a sparse grid rule, one needs to define, for each spatial dimension i, a family of 1D quadrature rules indexed by increasing levels of precision. Common examples of such families involve the Clenshaw-Curtis **(CC)** rules, Gauss-Legendre **(GL)**, or Gauss-Hermite **(GH)**.

Our desire is to construct a sparse grid that uses the Truncated Normal **(TN)** family, which depends on the choice of the four parameters $\overline{\mu}$, $\overline{\sigma}$, a and b. Technically, we could use a different family for each dimension, such as **(TN)** for dimension 1, and **(CC)** for dimension 2. Even if we only plan to use the **(TN)** family, we would technically be free to specify a different set of parameters in every dimension. In the interests of simplicity, however, we will focus on the construction of a sparse grid that only uses the **TN** family, and which uses a fixed set of parameters that are used for all spatial dimensions.

While it is possible to precompute the nodes and weights of the elements of the more common quadrature families, we have seen that the evaluation of any element of the **TN** family requires a substantial calculation. Therefore, a program which repeatedly requires instances of such rules should save all previously computed rules, and then, upon a new request, "remember" whether it has already computed that rule, so that unnecessary recomputation can be avoided.

Novak and Ritter [12] suggest a criterion for selecting the elements of a suitable family for sparse grids by specifying that, for level indices $\ell = 0, 1, 2, ...$, we select a quadrature rule Q^ℓ with the property that the rule has exactness at least $2 \cdot \ell + 1$; in other words, the family should have an exactness sequence of (at least) 1, 3, 5, 7, 9, and so on. Novak and Ritter show that for a general class of integration problems, if sparse grids are constructed from such 1D families, then the sparse grid of index L will have a multivariate exactness of at least $2 \cdot L + 1$. Thus, if we choose our 1D quadrature factors properly, we can guarantee the exactness of our sparse grids.

Since the **TN** family is computed using a Gauss procedure, the quadrature rule of n points will have exactness $2 \cdot n - 1$. This suggests that we could simply choose as our 1D family the quadrature rules of order 1, 2, 3, 4, ..., since these will have exactness 1, 3, 5, 7, as desired. Let us call this the Truncated Normal Linear **TNL** growth rule.

However, it turns out that, for sparse grids, it is highly desirable to take advantage of nesting, that is, the repeated use of abscissas in the 1D rules. In this case, if we add the stipulation that the 1D rules must be odd, then our family will involve the sequence of rules of order 1, 3, 3, 5, 5, ... with exactness 1, 5, 5, 7, 7, and so on. We call this the Truncated Normal Odd **TNO** growth rule. This choice means that our 1D rules will often be slightly more exact than necessary, but there will be a substantial reduction in the point count of the resulting sparse grids, due to the partial nesting of the 1D rules.

Let us represent our underlying 1D quadrature rules as $\mathcal{Q}$, and the element of level ℓ by $\mathcal{Q}^\ell$. The standard Smolyak formula then indicates how to build, in any spatial dimension D, a family of sparse grids $\mathcal{A}(L, D)$ indexed by level L, which forms a sequence of increasing exactness. For completeness, we cite the formal definition of such a rule:

$$\mathcal{A}(L, M) = \sum_{L-M+1 \leq |\vec{\ell}| \leq L} (-1)^{L-|\vec{\ell}|} \left(\begin{array}{c} M-1 \\ L-|\vec{\ell}| \end{array} \right) (\mathcal{Q}^{\ell_1} \otimes \cdots \otimes \mathcal{Q}^{\ell_M})$$

This formula tells us how to start with the points x and weights w of a family of 1D quadrature rules, and produce an array of multidimensional points X and a vector of weight W that form our sparse grid quadrature rule. Since we have determined how to evaluate the 1D information, the remaining technical issue is to implement the procedure that combines that information to produce $\mathcal{A}(L, M)$.

3.15 Implementation of a Sparse Grid Rule

A Matlab program by Heiss and Winschel [9] is available, called **nwspgr**, which implements the sparse grid construction procedure for a set of predefined 1D quadrature families for the unit interval $[0, 1]$ with unit weight function, and for the infinite interval $(-\infty, +\infty)$, with the normal weight function.

What makes this program interesting, though, is that the authors made it fairly easy for a user to create sparse grids based on other 1D factor families. In particular, the user needs only to define a Matlab function of the form

```
function [ x, w ] = rule ( l )
```

which is given as input an index $1 \leq l$ and which returns the points x and weights w of the corresponding 1D quadrature rule.

Thus, in order to make a simple implementation of a sparse grid based on a single **TN** family with the **TNO** growth rate, we simply need to write a function **tno.m** which accepts the input l, determines the corresponding point count n, sets the values of the four parameters $\overline{\mu}$, $\overline{\sigma}$, a and b, and then follows the procedures discussed earlier to evaluate the points and weights of the quadrature rule. Moreover, the function should save the results from each computation, so as to have them at hand in case the same rule is requested later.

A special version of the Heiss and Winschel code was prepared, called **truncated_normal_sparse_grid**, to which was added the necessary **tno.m** function, and a number of sparse grids were computed, always with a single set of parameters that applied in every dimension.

3.16 Experiment with the Sparse Grid

If we use **truncated_normal_sparse_grid** to compute a sparse grid in dimension 2, of level 3, with the fixed parameters:

- $\overline{\mu} = 0.0$;
- $\overline{\sigma} = 1.0$;
- $a = -1.0$
- $b = +1.0$

then we get the following sparse grid, with exactness 7. Note that certain points (X, Y) occur more than once in the list. A more compact list can be formed by deleting the duplicated (X, Y) and adding its weight to the original.

I	W	X	Y
1	0.09877141338753155	-0.8984499284579618	0.0
2	0.06636281106291925	-0.7516984074121438	-0.7516984074121438
3	0.1248841817693534	-0.7516984074121438	0.0
4	-0.2576098038951918	-0.7516984074121438	0.0
5	0.06636281106291922	-0.7516984074121438	0.7516984074121438
6	0.2422365795893659	-0.521317286312556	0.0
7	0.1248841817693534	0.0	-0.7516984074121438
8	0.2350120286709092	0.0	0.0
9	-0.4847803922096159	0.0	0.0
10	0.1248841817693534	0.0	0.7516984074121438
11	0.09877141338753155	0.0	-0.8984499284579618
12	-0.2576098038951918	0.0	-0.7516984074121438
13	0.2422365795893659	0.0	-0.521317286312556
14	-0.4847803922096159	0.0	0.0
15	0.3179840140462047	0.0	0.0
16	0.2422365795893658	0.0	0.5213172863125559
17	-0.2576098038951917	0.0	0.7516984074121438
18	0.09877141338753152	0.0	0.898449928457962
19	0.3179840140462047	0.0	0.0
20	0.2422365795893658	0.5213172863125559	0.0
21	0.06636281106291922	0.7516984074121438	-0.7516984074121438
22	0.1248841817693534	0.7516984074121438	0.0
23	-0.2576098038951917	0.7516984074121438	0.0
24	0.06636281106291919	0.7516984074121438	0.7516984074121438
25	0.09877141338753152	0.898449928457962	0.0

Recall the integration problem used for the 2D product rule:

$$I(f) = \frac{1}{S^2 2\pi} \int_{-1}^{+1} \int_{-1}^{+1} \frac{1}{1+x^2+y^2} e^{-\frac{x^2+y^2}{2}} dx$$

for which the 5x5 product rule returned an estimate $Q(f) \approx 0.535$

For that problem, the PDF parameters were:

- $\overline{\mu} = 0.0$;
- $\overline{\sigma} = 1.0$;
- $a = -1.0$
- $b = +1.0$

We estimate $I(f)$, using **truncated_normal_sparse_grid** to compute a sequence of sparse grid rules of increasing exactness:

Method	N	Q
tnsg0	1	1.0
tnsg1	7	0.6279671543
tnsg2	15	0.6688136155
tnsg3	25	0.6673175509
tnsg4	49	0.6717738876
tnsg5	63	0.6714288818
tnsg6	111	0.6717645364
Matlab quad2d()	73	0.6717800326
Mathematica NIntegrate[]	?	0.6717800275

Note that the point counts N for the sparse grids have not been reduced by merging duplicate points. Note also that, assuming that the correct integral lies somewhere between Matlab and Mathematica's estimates, the sparse grid of level 4 comes quite close, but then there is a bit of a deviation at level 5, and a partial "correction" at level 6. This is just an experiment, so we will not discuss it to death.

3.17 Software

All the software is available in C, C++, Fortran77, Fortran90, MATLAB, Octave and Python. Each software item is available on a web page, classified by name of software and implementation language. The language directories are **c_src**, **cpp_src**, **f77_src**, **f_src**, **m_src**, **octave_src** and **py_src**. Thus, the MATLAB version of the **truncated_normal** software can be located by accessing the web page:
"https://people.sc.fsu.edu/~jburkardt/m_src/truncated_normal/truncated_normal.html".

truncated_normal is the software implementing the tasks of computing the PDF, CDF, inverse CDF, sampling, mean, variance, and moments.

jacobi_eigenvalue is the software for the Jacobi iteration that compute the eigenvalues and eigenvectors of a symmetric positive semi-definite matrix.

quadmom is the software for computing quadrature rules by applying the moment-matrix method of Golub and Welsch. The user is required to supply information defining the moments. The software comes with examples of moment calculations for the Gauss-Legendre, Gauss-Laguerre, Gauss-Hermite (standard normal) weight functions, as well as upper, lower, and doubly-truncated normal distributions.

truncated_normal_rule is the software for computing quadrature rules involving the truncated normal distribution; the necessary moment calculation is built into the software.

power_rule is the software for computing a multidimensional quadrature rule that is a product of identical 1D quadrature rules. This is the simplest way of constructing a multivariate quadrature rule.

product_rule is the software for computing a multidimensional quadrature rule that is a product of 1D quadrature rules, each of which may have a different number of points, and be associated with a different weight function.

truncated_normal_sparse_grid is the software for computing a multidimensional sparse grid based on the truncated normal distribution. The software is a modification of the Heiss and Winschel Matlab function **nwspgr**, including a new function **tno** which evaluates the necessary 1D quadrature rules.

3.18 Conclusion

This discussion was provoked by the desire to be able to apply the sparse grid approach to stochastic quantities governed by a truncated normal distribution. We have gathered together a variety of information in order to make it possible to understand the truncated normal distribution, to see how to define univariate and multivariate integrals in which the truncated normal distribution plays the role of a PDF, how to determine quadratures rules to estimate such integrals, and how to combine product rules to create efficient sparse grids for the multivariate case.

Software was developed to illustrate the discussion, and a number of numerical examples were carried out and documented, so that others may verify the statements made here, and make comparisons with other approaches.

References

[1] JD Beasley, SG Springer, Algorithm AS 111: The Percentage Points of the Normal Distribution, Applied Statistics, Volume 26, Number 1, 1977, pages 118-121.

[2] Barry Brown, James Lovato, Kathy Russell, CDFLIB, available at https://www.netlib.org/random/.

[3] John Cook, General formula for normal moments, available at https://www.johndcook.com/blog/2012/11/06/general-formula-for-normal-moments/

[4] Sylvan Elhay, Jaroslav Kautsky, Algorithm 655: IQPACK, FORTRAN Subroutines for the Weights of Interpolatory Quadrature, ACM Transactions on Mathematical Software, Volume 13, Number 4, December 1987, pages 399-415.

[5] Ronald Fisher, The moments of the distributions of normal samples of measures of departure from normality, Proceedings of the Royal Society of London, Series A, Volume 130, Number 812, pages 16-28, 1930.

[6] Ronald Fisher, The truncated normal distribution, British Association for the Advancement of Science, Mathematical Tables, Volume 5, pages xxxiii-xxxiv, 1931.

[7] Gene Golub, John Welsch, Calculation of Gaussian Quadrature Rules, Mathematics of Computation, Volume 23, Number 106, April 1969, pages 221-230.

[8] Gene Golub, Charles Van Loan, Matrix Computations, Third Edition, Johns Hopkins, 1996.

[9] Florian Heiss, Viktor Winschel, Likelihood approximation by numerical integration on sparse grids, Journal of Econometrics, Volume 144, Number 1, May 2008, pages 62-80.

[10] David Hill, Algorithm AS 66: The Normal Integral, Applied Statistics, Volume 22, Number 3, 1973, pages 424-427.

[11] Norman Johnson, Samuel Kotz, Narayanaswamy Balakrishnan, Continuous Univariate Distributions, Second edition, Wiley, 1994.

[12] Erich Novak, Klaus Ritter, Simple cubature formulas with high polynomial exactness, Constructive Approximation, Volume 15, Number 4, December 1999, pages 499-522.

[13] Michael Wichura, Algorithm AS 241: The Percentage Points of the Normal Distribution, Applied Statistics, Volume 37, Number 3, 1988, pages 477-484.