Deriving quadrature rules from Gaussian processes

Thomas Minka

November 15, 2000

Abstract

Quadrature rules are often designed to achieve zero error on a small set of functions, e.g. polynomials of specified degree. A more robust method is to minimize average error over a large class or distribution of functions. If functions are distributed according to a Gaussian process, then designing an average-case quadrature rule reduces to solving a system of $2n$ equations, where n is the number of nodes in the rule (O'Hagan, 1991). It is shown how this very general technique can be used to design customized quadrature rules, in the style of Yarvin and Rokhlin (1998b), without the need for singular value decomposition and in any number of dimensions. It is also shown how classical Gaussian quadrature rules, trigonometric lattice rules, and spline rules can be extended to the average-case and to multiple dimensions by deriving them from Gaussian processes. In addition to being more robust, multidimensional quadrature rules designed for the average-case are found to be much less ambiguous than those designed for a given polynomial degree.

1 Introduction

In numerical integration we are interested in estimating the value of an integral

$$I = \int_A f(x)\mu(x)dx \tag{1}$$

over some domain A with fixed weight function $\mu(x)$, via a weighted sum of function evaluations:

$$\hat{I} = \sum_{i=1}^{n} w_i f(x_i) \tag{2}$$

To do this, we must have some idea about the kind of function being integrated. The quadrature design problem is to choose the optimal nodes x_i and weights w_i for a given class of functions.

The classical approach to this problem is to choose a set of $2n$ basis functions and design the rule to integrate these functions exactly. This leads to a system of $2n$ equations for the nodes and weights that can be solved numerically (Ma et al., 1996). However, the resulting rule is only guaranteed to work on functions spanned by that small basis.

Another approach, which is increasingly popular, is to design the rule to achieve a broad guarantee over the whole class of functions, such as minimizing the average-case or worst-case error. This paper focuses on the average-case problem, though the two are related. One of the most general design methods is that of Yarvin and Rokhlin (1998b). They show how to obtain approximate average-case rules by performing a singular value decomposition on the function class of interest. The rule is designed to exactly integrate the basis functions with largest singular values, which ensures that most of the function class will be covered by the rule. The drawback is that computing the singular value decomposition of an arbitrary class of functions is nontrivial.

The Gaussian process method is an alternative method for deriving average-case quadratures, which originated in the statistics literature (Diaconis, 1988; O'Hagan, 1991) and deserves to be better known. In this method, a probability distribution is placed on functions and the average quadrature error under this distribution is minimized. This results in a system of $2n$ equations, the same size as the classical approach, which is solved numerically.

A technique which is closely related to Gaussian processes is the design of worst-case rules in Hilbert space. This is because the squared error of a quadrature rule averaged over functions from a Gaussian process is equal to its worst error over functions in the unit ball of a certain reproducing kernel Hilbert space (Paskov, 1993). The kernel of the space is given by the covariance function of the process. The particular case considered by Paskov (1993) was a Weiner process. Similarly, Smale (1985) has shown how to compute the average absolute error of a quadrature rule by using Hilbert space manipulations (in his case, they were Sobolev spaces).

The Gaussian process and Hilbert space viewpoints each have their own advantages. This paper uses the Gaussian process viewpoint because in applications it is often more natural to think in terms of a distribution of functions, rather than the unit ball in some unusual Hilbert space (not just a Sobolev space).

This paper establishes connections between the Gaussian process method, Yarvin & Rokhlin's method, and quadratures based on polynomial, trigonometric, and spline interpolation. As a result of these connections, it is shown in section 3 that a problem considered by Yarvin and Rokhlin (1998b) can be solved more easily and accurately than previously possible. In sections 4, 5, and 6 it is shown how standard interpolatory quadratures can be generalized to the average-case and to multiple dimensions by using a Gaussian process formulation. Figure 2 gives an idea of the improvements to be had by moving to an average-case design criterion. The ease of handling multiple dimensions is illustrated with some optimal node arrangements computed in 2D.

Gaussian processes are treated here as a computational tool for constructing quadrature rules. However, it is hoped that the reformulation of the quadrature problem presented here may also lead to new theoretical advances.

2 The Gaussian process method

A good starting point for understanding Gaussian processes is to consider a linear combination of basis functions $\phi_i(x)$ where the coefficients are independent Gaussian random variables. That is,

$$\begin{aligned} f(x) &= \sum_i a_i \phi_i(x) && (3) \\ p(a_i) &\sim \mathcal{N}(0, v_i) = \frac{1}{\sqrt{2\pi v_i}} \exp(-\frac{a_i^2}{2v_i}) && (4) \end{aligned}$$

This obviously defines a probability distribution over functions $f(x)$. What kind of distribution is it? The value $f(x)$ at a fixed location x is Gaussian distributed, since it is a weighted sum of Gaussian random variables:

$$f(x) \sim \mathcal{N}(0, \sum_i v_i \phi_i(x)^2) \tag{5}$$

Similarly, the values $f(x)$ and $f(y)$ at two fixed locations x and y are jointly Gaussian, with covariance

$$\begin{aligned} k(x,y) &= E[f(x)f(y)] - E[f(x)]E[f(y)] && (6) \\ &= E[\sum_{ij} a_i a_j \phi_i(x) \phi_j(y)] - 0 && (7) \\ &= \sum_i v_i \phi_i(x) \phi_i(y) && (8) \end{aligned}$$

More generally, the values $f(D) = (f(x_1), ..., f(x_n))$ at any finite set of locations $D = (x_1, ..., x_n)$ are jointly Gaussian:

$$\begin{aligned} p(f(D)) &\sim \mathcal{N}(\mathbf{0}, k(D,D)) && (9) \\ k(D,D) &= \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & & \ddots & \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} && (10) \end{aligned}$$

The idea of Gaussian processes is to now use (9) as our definition of the distribution of f. All of the distribution's essential properties follow from $k(x,y)$, the covariance function, without reference to v_i or ϕ_i. The covariance function $k(x,y)$ need only be a symmetric and positive definite function. One can also have a process with nonzero mean, in which case both the mean function and covariance function must be specified. This additional flexibility is rarely needed and will not be used here.

If f is a function drawn from a Gaussian process and we have values $f(D) = [\, f(x_1) \cdots f(x_n)\,]^{\mathrm{T}}$ at locations $D = (x_1, ..., x_n)$, then we can interpolate the function to other locations x, via the conditional density

$$p(f(x)|f(D)) \sim \mathcal{N}(k(x,D)k(D,D)^{-1}f(D),\ k(x,x)-k(x,D)k(D,D)^{-1}k(D,x)) \quad (11)$$

$$k(x,D) = [\,k(x,x_1) \quad k(x,x_2) \quad \cdots \quad k(x,x_n)\,] = k(D,x)^{\mathrm{T}} \quad (12)$$

This formula comes from conditioning the joint Gaussian density of $(f(x), f(D))$ on $f(D)$. The conditional mean

$$E[f(x)|f(D)] = k(x,D)k(D,D)^{-1}f(D) \quad (13)$$

is the interpolant, which is evidently a linear combination of the functions $k(x, x_i), i = 1, ..., n$.

The integral of f with respect to weight function $\mu(x)$ has conditional density

$$p(\int_A f(x)\mu(x)dx|f(D)) \sim \mathcal{N}(\mathbf{u}(D)^{\mathrm{T}}k(D,D)^{-1}f(D),\ u-\mathbf{u}(D)^{\mathrm{T}}k(D,D)^{-1}\mathbf{u}(D)) \quad (14)$$

$$\mathbf{u}(D)^{\mathrm{T}} = \int_A k(x,D)\mu(x)dx \quad (15)$$

$$u = \int_A\int_A k(x,y)\mu(x)\mu(y)dxdy \quad (16)$$

The conditional mean

$$\hat{I} = \mathbf{u}(D)^{\mathrm{T}}k(D,D)^{-1}f(D) \quad (17)$$

is used as the estimate of the integral (O'Hagan, 1991). It is a quadrature rule of the form (2) with weight vector $\mathbf{w}^{\mathrm{T}} = \mathbf{u}(D)^{\mathrm{T}}k(D,D)^{-1}$ and is equivalent to integrating the interpolant (13). The conditional variance provides an error estimate to minimize to determine the optimal nodes $x_1, ..., x_n$ (O'Hagan, 1991). The same argument applies when x is multi-dimensional.

This procedure can also be understood under the basis function interpretation of the process. Consider the least-squares problem

$$J = \sum_j v_j\left(\sum_i w_i\phi_j(x_i) - \int_A \phi_j(x)\mu(x)dx\right)^2 \quad (18)$$

The normal equations for the weights w_i are

$$\sum_j v_j\phi_j(x_1)\left(\sum_i w_i\phi_j(x_i)\right) = \sum_j v_j\phi_j(x_1)\left(\int_A \phi_j(x)\mu(x)dx\right) \quad (19)$$

$$\vdots$$

$$\sum_j v_j\phi_j(x_n)\left(\sum_i w_i\phi_j(x_i)\right) = \sum_j v_j\phi_j(x_n)\left(\int_A \phi_j(x)\mu(x)dx\right) \quad (20)$$

which we can rewrite in terms of $k(x,y) = \sum_i v_i\phi_i(x)\phi_i(y)$ as

$$\sum_i w_i k(x_1, x_i) = \int_A k(x_1, x)\mu(x)dx \quad (21)$$

$$\vdots$$

$$\sum_i w_i k(x_n, x_i) = \int_A k(x_n, x)\mu(x)dx \quad (22)$$

In other words, $k(D, D)\mathbf{w} = \mathbf{u}(D)$. To solve for the nodes, expand the objective (18) into

$$\begin{aligned}
J &= \sum_j v_j \left(\sum_i w_i \phi_j(x_i)\right)^2 - 2\sum_j v_j \left(\sum_i w_i \phi_j(x_i)\right) \left(\int_A \phi_j(x)\mu(x)dx\right) && (23)\\
&\quad + \sum_j v_j \left(\int_A \phi_j(x)\mu(x)dx\right)^2 && (24)\\
&= \mathbf{w}^{\mathrm{T}} k(D, D)\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{u}(D) + u && (25)\\
&= u - \mathbf{u}(D)^{\mathrm{T}} k(D, D)^{-1}\mathbf{u}(D) && (26)
\end{aligned}$$

which is exactly the conditional variance in (14). Thus the quadrature nodes and weights which minimize average squared error for a (possibly infinite) set of basis functions ϕ_j are exactly the nodes and weights derived from the Gaussian process with covariance (8).

3 Yarvin & Rokhlin's problem

As an example of how to exploit the Gaussian process point of view, consider Yarvin & Rokhlin's problem of designing an average-case quadrature rule for the integrals

$$\int_0^\infty \exp(-tx)dx \qquad t \in [1, b] \tag{27}$$

This involves a continuously infinite set of functions $\exp(-tx)$. If we consider a linear combination of these functions with unit-variance coefficients, then the covariance function (8) is

$$\begin{aligned}
k(x, y) &= \int_1^b \exp(-tx)\exp(-ty)dt && (28)\\
&= \frac{\exp(-(x+y)) - \exp(-b(x+y))}{x+y} && (29)
\end{aligned}$$

The required integrals are

$$\begin{aligned}
\int_0^\infty k(x, y)dx &= \int_0^\infty \frac{\exp(-(x+y)) - \exp(-b(x+y))}{x+y}dx && (30)\\
&= \int_y^\infty \frac{\exp(-x) - \exp(-bx)}{x}dx && (31)\\
&= E_1(y) - E_1(by) && (32)\\
\int_0^\infty \int_0^\infty k(x, y)dxdy &= \frac{b-1}{b} && (33)
\end{aligned}$$

where $E_1(x)$ is the exponential integral:

$$E_1(x) = \int_x^\infty \exp(-y)/y\,dy \tag{34}$$

$n = 14, b = 501$	RMS error	Max absolute error
Gaussian process	2.9e-09	2e-08
Yarvin & Rokhlin	3.3e-09	4.4e-08

Figure 1: Comparison of custom quadrature rules for the integrals (27), computed using the Gaussian process method versus Yarvin & Rokhlin's SVD method. RMS is the root-mean-square error, which is explicitly minimized by the Gaussian process method.

We can now solve numerically for the optimal nodes and weights. To avoid local optima, one can use a continuation method as described by Yarvin and Rokhlin (1998b). The results in this paper come from the Nelder-Mead simplex method with multiple starting points. For $n = 14$ and $b = 501$, figure 1 compares the best quadrature obtained this way to Yarvin & Rokhlin's best result (Yarvin & Rokhlin, 1998a). The results are similar, and indeed they should be.

The difference is the method of computation. Yarvin & Rokhlin's method requires first computing a singular value decomposition of the functions $\exp(-xt)$. This is a nontrivial task which requires a fair amount of experimentation to get right. Their method involves sampling the set of t values and x values in order to get a discrete SVD problem. They report that "the interval of integration was divided into several subintervals, and a combination of a (classical) Gaussian quadrature at Legendre nodes and polynomial interpolation was used on each subinterval."

It goes without saying that these custom rules perform far better than general-purpose quadrature on these functions. Thus custom rules deserve serious consideration in many applications. The Gaussian process method makes it easier to exploit these opportunities.

Another advantage of the Gaussian process method is that we can give the basis functions different weight in the minimization, in order to reduce the error on more common cases. This idea is explored in the next section.

4 Polynomial interpolation and Gaussian quadrature

Suppose we want an average-case rule for the monomials

$$\int_{-1}^{1} x^t dx \qquad t = 0, 1, 2, \ldots \tag{35}$$

To emphasize low-order monomials in the average, we can give the functions exponentially decaying weight $v_t = b^t, 0 \leq b < 1$. Then the covariance function is

$$k(x, y) \;=\; \sum_{t=0}^{\infty} b^t x^t y^t = \frac{1}{1 - bxy} \qquad 0 \leq b < 1 \tag{36}$$

As $b \to 0$, low-order monomials are overwhelmingly favored, which means the interpolants produced by (13) will be polynomials of minimum order. The resulting quadrature rules will integrate low-order polynomials exactly—in other words, they will be Gaussian quadratures.

The integrals of the covariance function are

$$\int_{-1}^{1} k(x,y)dx = \frac{\log(1+by) - \log(1-by)}{by} = 2\frac{\operatorname{arctanh}(by)}{by} \tag{37}$$

$$\int_{-1}^{1}\int_{-1}^{1} k(x,y)dxdy = \frac{2}{b}(\operatorname{dilog}(1-b) - \operatorname{dilog}(1+b)) \tag{38}$$

$$\operatorname{dilog}(x) = \int_{1}^{x} \frac{\log(y)}{1-y}dy \tag{39}$$

For $b = 0.01$ and $n = 6$, the optimal nodes and weights found by the Gaussian process method are numerically similar to the Gauss-Legendre nodes and weights. Smaller values of b cannot be handled reliably since the covariance matrix becomes nearly singular. However, in that case we could just sum over the powers $t = 0, ..., n-1$ in the covariance function and dispense with the b parameter.

The benefit of the infinite sum in (36) is that we can include polynomials of higher order in the optimization simply by increasing b. For example, with $n = 8$ and $b = \exp(-1/25)$ we get the rule shown in figure 2(a). This rule has a maximum error of 0.0007 on the monomials up to degree 100, as shown in figure 2(c). By contrast, the eight-point Gauss-Legendre rule has a maximum error of 0.0168. To get the same maximum error as the average-case rule, we would have to use a Gauss-Legendre rule with 16 points—twice as many function evaluations.

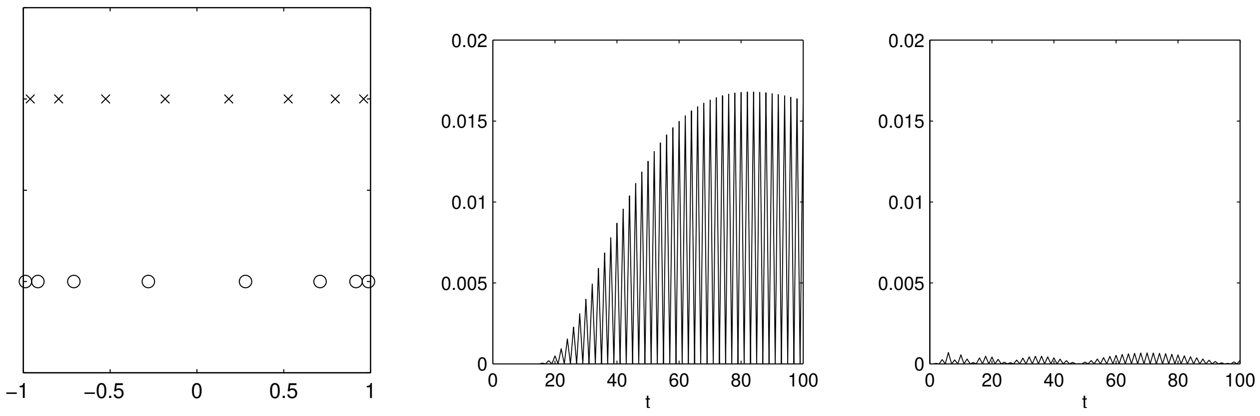
The classical Gaussian quadratures are all limited in this way. In order to achieve zero error on a small set of functions, they sacrifice performance on everything else. With this approach, the only way to get low error on a large set of functions is to increase the number of nodes. But with the average-case approach, the number of nodes only reflects the amount of computation you are willing to invest—it is independent of the class of functions you are trying to integrate. And it is surprising how far you can reach when you allow a slightly nonzero error as opposed to requiring zero error.

The other advantage of the Gaussian process method is that the objective function easily generalizes to multiple dimensions. In multiple dimensions, we want to sum over all multivariate monomials $x_1^{t_1} \cdots x_d^{t_d}$. This can be done by taking products of the covariance function:

$$k(\mathbf{x},\mathbf{y}) = \prod_{k=1}^{d}\left(\sum_{t=0}^{\infty} b^t x_k^t y_k^t\right) \tag{40}$$

$$= \prod_{k=1}^{d} \frac{1}{1 - b x_k y_k} \tag{41}$$

The monomial $x_1^{t_1} \cdots x_d^{t_d}$ will appear in this sum weighted by $b^{t_1+\cdots+t_d}$, so as $b \to 0$ the monomials with small degree $\sum_k t_k$ will be favored. The number of monomials with degree at most p is $\binom{d+p}{p}$,



(a) (b) (c)

Figure 2: (a) Crosses: nodes of the eight-point Gauss-Legendre rule. Circles: nodes of an eight-point average-case quadrature rule. (b) The absolute error of eight-point Gauss-Legendre quadrature versus (c) eight-point average-case quadrature on $\int_{-1}^{1} x^t dx, t = 0, 1, ..., 100$. Gauss-Legendre has a maximum error of 0.0168 while average-case has a maximum error of 0.0007. The error oscillates across even and odd powers.

so if $n = \binom{d+p}{p}$ then the Gaussian process interpolant as $b \to 0$ will be the unique polynomial interpolant of degree p. For intermediate values of n, the interpolant will be a compromise between a polynomial of degree p and $p-1$. For quadrature on $A = [-1, 1]^d$, the necessary integrals are simply products of those above, i.e.

$$\int_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) d\mathbf{x} = \prod_{k=1}^{d} \left(2 \frac{\operatorname{arctanh}(b y_k)}{b y_k} \right) \tag{42}$$

$$\int_{\mathbf{y}} \int_{\mathbf{x}} k(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} = \left(\frac{2}{b} (\operatorname{dilog}(1-b) - \operatorname{dilog}(1+b)) \right)^d \tag{43}$$

Quadratures are computed just as in one dimension, by minimizing (26)—there are simply more parameters to optimize. This is an improvement over previous numerical methods which involve systems of equations larger than the number of free parameters (Hammer & Stroud, 1958). For example, without imposing symmetry conditions, a degree 7 rule in three dimensions would normally require $\binom{3+7}{7} = 120$ equations in $4n$ variables (Hammer & Stroud, 1958). The Gaussian process method reduces this to $4n$ equations in $4n$ variables, with the polynomial degree chosen automatically by b.

Another difference is that, even for small b, the resulting rules will not merely achieve a specified polynomial degree but also try to minimize their error on higher order polynomials. From Cools and Rabinowitz (1993) we know that quadrature rules which achieve a specified polynomial degree are generally not unique, and sometimes there is an infinity of such rules. The Gaussian process method uses the error on higher-order polynomials as a tie-breaker. To demonstrate this, a numerical study of the optimal quadrature node arrangements was performed for $d = 2$ (the square) and $b = 0.1$ using the Nelder-Mead simplex method. The best arrangements found are shown in figure 3. This appears to be the full set, up to symmetry transformations—a sharp contrast with Cools and Rabinowitz (1993). Only $n = 8$ has a tie between two equally good rules. No symmetries were enforced in the optimization of these rules, and the initialization was random in each case. Notice how they focus especially on the boundaries of the square.

Interestingly, the rule for $n = 4$ is not the popular product rule $(\pm\sqrt{3}, \pm\sqrt{3})$. Both rules do well on polynomials of degree 3, but the product rule is much worse on polynomials of degree 4, and since b is not exactly zero, this matters. The rule given in the figure is

$$\mathbf{x} = \{(a,b),(-a,-b),(b,-a),(-b,a)\} \qquad a = 0.2732, b = 0.7696 \tag{44}$$

$$\mathbf{w} = \mathbf{1} \tag{45}$$

Similarly one can show that the four-point rule of Hammer and Stroud (1958) and the six-point rules of Wissman and Becker (1986) are inferior to the rules in figure 3. The relative uniqueness of average-case rules means that theoretical analysis of quadrature rules might be simpler from an average-case standpoint.

To get rules spanning higher polynomial degrees, we increase b. The resulting rules are largely the same but have their nodes moved closer to the boundaries, as we would expect. Also the optimization converges much faster, which suggests that setting b to an artificially high value and then lowering it during the optimization might be a useful strategy.

In multiple dimensions it is especially important to consider high-order average-case rules instead of zero-error rules. Consider that in 25 dimensions, there are $\binom{25+2}{2} = 351$ monomials of degree 2, and 3276 monomials of degree 3. The corresponding zero-error rules will grow at the same rate, which means that accurate integration of a function with high-order content, say degree 7, will require an astronomical number of Gaussian quadrature nodes. This provides an explanation for why zero-error interpolatory rules do so badly on integrals like (Novak et al., 1999)

$$\int_{\Re^d} \cos(|\mathbf{x}|)\exp(-\mathbf{x}^{\mathrm{T}}\mathbf{x})d\mathbf{x} \tag{46}$$

compared to simpler methods like quasi-Monte Carlo, even though the integrand is "smooth."

One drawback to the above method is that polynomials explode at infinity, which means the series (36) cannot be used for x on an infinite domain like $A = (-\infty, \infty)$. To accomodate this, we can use the damped polynomials $x^t \exp(-bx^2/2)$, with the fast-converging weight sequence $b^t/t!$:

$$k(x,y) = \sum_{t=0}^{\infty} \frac{b^t}{t!} x^t \exp(-\frac{b}{2}x^2) y^t \exp(-\frac{b}{2}y^2) \tag{47}$$

$$= \exp(bxy)\exp(-\frac{b}{2}x^2)\exp(-\frac{b}{2}y^2) = \exp(-\frac{b}{2}(x-y)^2) \tag{48}$$

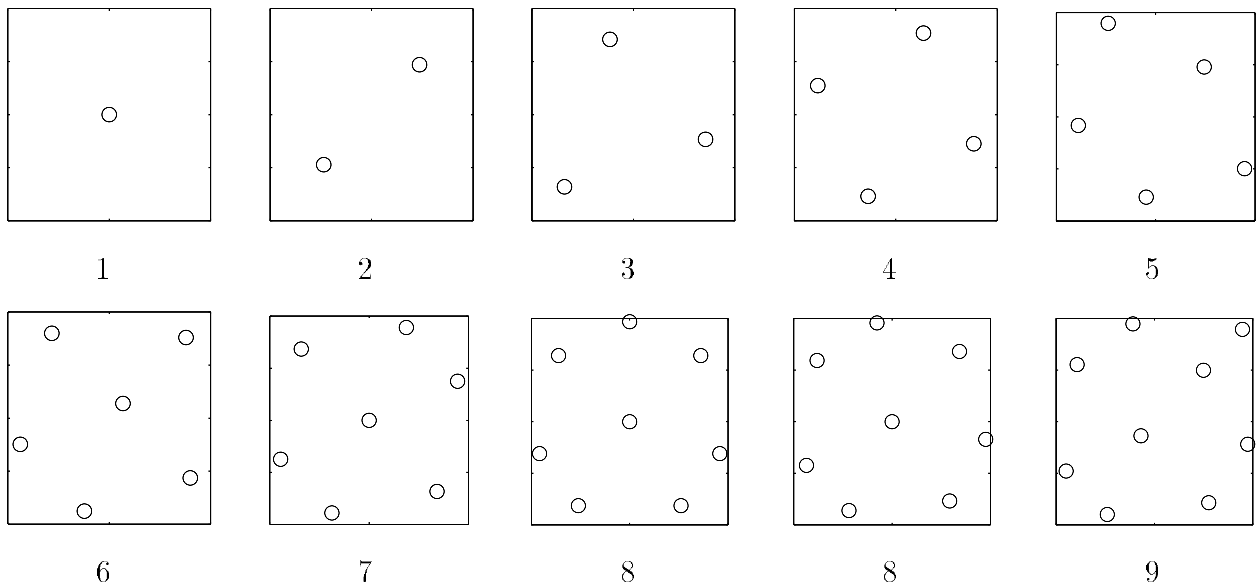


Figure 3: The best node arrangements found for average-case Gaussian quadrature in the square $[-1,1]^2$ for $n = 1, \ldots, 9$. It is conjectured that no rules are better or equal to these in the average case, excluding symmetry transformations.

As $b \to 0$, the function $x^t \exp(-bx^2/2) \to x^t$ and low-order polynomials will have priority, so the quadratures will be Gaussian. With the weight function $\mu(x) = \exp(-x^2)$, we get Gauss-Hermite quadratures. This covariance function was used by O'Hagan (1991), whose conjecture that the resulting quadratures are Gaussian as $b \to 0$ has now been shown to be correct. Unfortunately, the covariance function (47) is not very useful for deriving average-case rules, since the series $b^t/t!$ gives significant weight only to a narrow range of powers, selected by b. For an average-case rule, we would want a wide range of powers to be included.

5 Trigonometric quadrature and lattice rules

There has been much interest in integrating periodic functions, especially using lattice rules (Sloan, 1992). This section shows how the Gaussian process method can give rise to lattice rules as well as new non-lattice rules.

A (rank 1) lattice rule for integrating periodic functions on the hypercube $[0,1]^d$ has the form

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} f(\left\{ \frac{i\mathbf{z}}{n} \right\}) \tag{49}$$

where $\{x\}$ means the fractional part of x. The periodicity of f makes the brackets unnecessary but they are included anyway for clarity. Recently it has been pointed out that the known minimal rules for integrating trigonometric polynomials of specified degree are lattice rules (Cools & Sloan, 1996). This means that the Gaussian process method can be used to generate average-case lattice rules.

A natural basis for the periodic functions on $[0,1]$ is the Fourier basis:

$$f(x) = c_0 + \sum_{t=1}^{\infty} c_t \cos(2\pi t x) + \sum_{t=1}^{\infty} d_t \sin(2\pi t x) \tag{50}$$

As in section 4, we can construct a Gaussian process for these functions by giving higher order terms exponentially decaying weight:

$$\begin{aligned}
k(x,y) &= 1 + \sum_{t=1}^{\infty} b^t (\cos(2\pi t x)\cos(2\pi t y) + \sin(2\pi t x)\sin(2\pi t y)) && (51)\\
&= 1 + \sum_{t=1}^{\infty} b^t \cos(2\pi t(x-y)) && (52)\\
&= \mathrm{Re}\left(\frac{1}{1 - b\exp(2\pi i(x-y))}\right) \qquad i = \sqrt{-1} && (53)\\
&= \frac{1 - b\cos(2\pi(x-y))}{(1 - b\cos(2\pi(x-y)))^2 + (b\sin(2\pi(x-y)))^2} && (54)\\
&= \frac{1 - b\cos(2\pi(x-y))}{1 + b^2 - 2b\cos(2\pi(x-y))} && (55)\\
&= \frac{1}{2} + \frac{(1-b^2)/2}{1 + b^2 - 2b\cos(2\pi(x-y))} \qquad 0 \leq b < 1 && (56)
\end{aligned}$$

As $b \to 0$, the interpolants will be trigonometric polynomials of minimum degree, and the quadratures will be trigonometric quadratures. The integral of the covariance function is simply

$$\int_0^1 k(x,y)dx = \int_0^1 \left(1 + \sum_{t=1}^{\infty} b^t \cos(2\pi t(x-y))\right) dx = 1 \tag{57}$$

(One can also define trigonometric polynomials via complex exponentials, in which case the covariance function is complex-valued: $k(x,y) = 1/(1 - b\exp(2\pi i(x-y)))$.) In multiple dimensions, we sum over multivariate trigonometric polynomials by taking the product

$$\begin{aligned}
k(\mathbf{x},\mathbf{y}) &= \prod_{k=1}^{d} k(x_k, y_k) && (58)\\
&= \prod_{k=1}^{d} \left(\frac{1}{2} + \frac{(1-b^2)/2}{1 + b^2 - 2b\cos(2\pi(x-y))}\right) && (59)
\end{aligned}$$

The monomial $\cos(2\pi t_1 x_1)\cdots\cos(2\pi t_d x_d)$ will appear in this sum weighted by $b^{t_1+\cdots+t_d}$, so as $b \to 0$ the monomials with small degree $\sum_k t_k$ will be favored. The integral of this covariance function over $[0,1]^d$ is 1.

What do the optimal quadrature node arrangements look like? A numerical study was performed for $d = 2$ (the square) and $b = 0.001$. Because of the periodicity, the first node $\mathbf{x}_1$ is restricted to be $\mathbf{0}$, without loss of generality. In one dimension, the optimal nodes are known to be equally spaced (Cools & Sloan, 1996):

$$x_i = \frac{i-1}{n} \qquad i = 1..n \tag{60}$$

In two dimensions, the local minima of the error (26) seem to always satisfy projection consistency: any projection of the nodes onto one dimension gives the nodes (60) in some order. This makes sense considering the separable nature of the covariance function, and is conjectured here to always be true. Note that this projection property is not satisfied by a regular grid of nodes.

This conjecture allows us to reduce the problem from a continuous optimization to a combinatorial search over projection-consistent arrangements. Figure 4 gives the full set of optimal nodes obtained this way for $n = 2, ..., 9$. The results for $n = 2, 5, 8$ match the known optimal rules (Cools & Sloan, 1996). The optimal nodes for $n = 2, 3, 5, 7, 8$ form a lattice, and their weights are constant ($w_i = 1/n$), which means they are lattice rules. For $n = 4, 6, 9$ the optimal rules are not lattice rules: the nodes do not form a lattice and the weights are not constant.

We can make average-case rules in the same way, simply by increasing b. Interestingly, the optimal node arrangements are found to be exactly the same. The weights are simply scaled down. Scaling down the weights decreases the error on high degrees, whose true integral is zero, and increases the error only on $f(x) = 1$. Thus lattice rules have significance beyond the fact that they are minimal rules for integrating trigonometric polynomials of specified degree; they are also average-case rules.

Cools and Sloan (1996) have presented a class of minimal rules in 2D which are not unique for a given n and do not necessarily satisfy projection-consistency, which seems to contradict the results above. In fact, there is no contradiction: their optimality criterion is simply less restrictive. They only care about whether the rule achieves zero error on trigonometric polynomials of a certain degree, whereas the rules above are derived as the limit of an average-case rule as $b \to 0$. The limiting operation serves as a tie breaker between rules which perform equally well on polynomials of a particular degree but differently on polynomials of higher degree. In fact, it is straightforward to check that among the Cools-Sloan family of rules for $n = 8$, only one member performs well on polynomials of higher degree, and that is the rule shown in figure 4.

The Gaussian process method could be useful in exploring the structure of trigonometric rules in higher dimensions. However, the above method for finding these rules would require a great deal of computation. In 2D, there are only $O(n!)$ projection-consistent rules to search over (when symmetries are excluded), but in 3D there are $O(n!^2)$. My hope is that the results presented here will inspire researchers to find a better way of doing this.

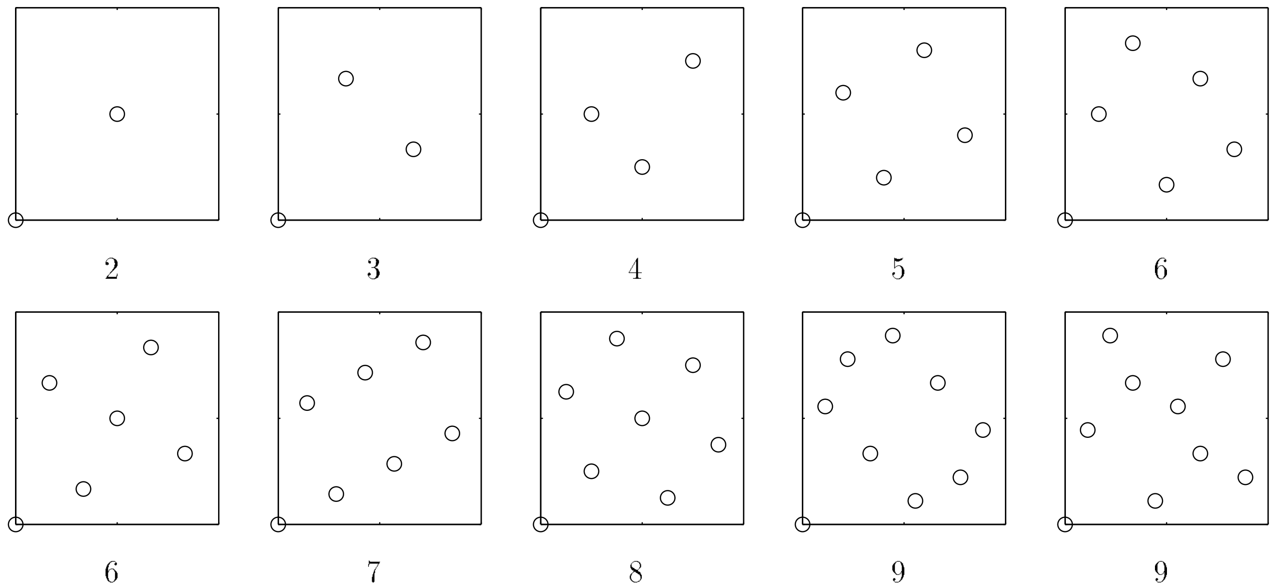


Figure 4: Optimal node arrangements for trigonometric quadrature in the square $[0,1]^2$ for $n = 2, \ldots, 9$, obtained by the methods in this paper. This is the full set, up to symmetry transformations; no rules are better than or equal to these. Except for $n = 4, 6, 9$, these are all lattice rules.

6 Splines

In the case of odd-order splines, it has been known for quite some time how to obtain them from a Gaussian process (Wahba, 1990). However, the usual derivation involves a lengthy discussion on reproducing kernel Hilbert spaces. This section shows how the basis function interpretation of Gaussian processes leads to a simpler derivation and to processes that are stationary. These spline processes are attractive for general curve-fitting and resemble those that appear throughout the Gaussian process literature. They can also be used to design quadrature rules.

The derivation of Wahba (1990) goes like this. A natural spline interpolant g of order $2m-1$ minimizes $\int_A g^{(m)}(x)^2 dx$; e.g. linear splines minimize $\int_A g'(x)^2 dx$ and cubic splines minimize $\int_A g''(x)^2 dx$. A Gaussian process interpolant maximizes its probability under the model, which for zero mean processes corresponds to minimizing the norm of the interpolant in an appropriate Hilbert space. By constructing a Hilbert space whose norm is $\int_A g^{(m)}(x)^2 dx$, one can get a Gaussian process which produces splines.

The new derivation is based on simply treating splines as piecewise polynomials that arise from piecewise basis functions. We start with a process consisting of piecewise constant functions and then take integrals of these functions to get splines of higher and higher order.

6.1 Linear splines

Any piecewise constant function on $[0,1]$ can be written as a constant plus a weighted sum of step functions:

$$f(x) = a_0 + \int_0^1 a(t)\operatorname{sgn}(x-t)dt \tag{61}$$

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{62}$$

The covariance function (8) corresponding to this basis is

$$\begin{aligned} k(x,y) &= 1 + b\int_0^1 \operatorname{sgn}(x-t)\operatorname{sgn}(y-t)dt & (63)\\ &= 1 + b - 2b|x-y| \qquad b > 0 & (64) \end{aligned}$$

In this formula, all step functions have a constant weight b. One could also use a non-uniform distribution of weights, to reflect prior knowledge about the location of discontinuities.

Since the Gaussian process interpolant is always a linear combination of the functions $k(x, x_i)$, the interpolants from (64) are piecewise linear, as shown in figure (5). This makes the process

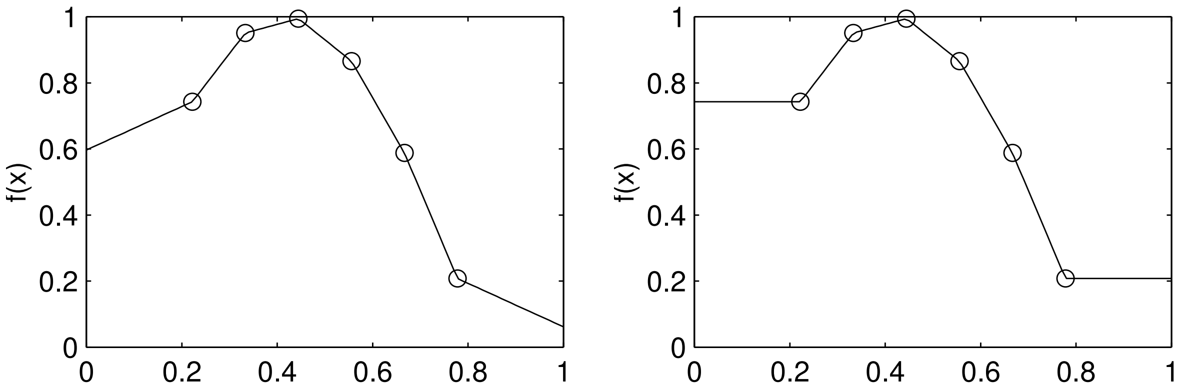


Figure 5: Piecewise linear interpolants using $b = 1$ (left) and $b = 10^{-4}$ (right). The interpolant on the right corresponds to the "natural" linear spline.

attractive for nonparametric regression, as done in Currin et al. (1991). The parameter b controls the expected number of discontinuities in the function and affects how the interpolant behaves outside the range of the data. As $b \to 0$, discontinuities become very improbable and so the function is assumed constant outside the range of the data. The result is a natural spline of order 1.

This is our first example of a process whose interpolants do not have the same form as functions from the process. The interpolant is piecewise linear because it is an average over many possible piecewise constant functions. This discrepancy between the functions and their interpolants can be expected when there are discontinuities at varying points in the functions. Functions with continuous derivatives like polynomials and sinewaves do not have this discrepancy; the interpolants are also polynomials and sinewaves.

Linear spline quadrature can thus be seen as average-case quadrature, in the sense of averaging over possible discontinuities in the function. For quadrature on $A = [0, 1]$ with weight $\mu(x) = 1$, the integrals of (64) are

$$\int_0^1 k(x, y)dx = 1 + b - b(y^2 + (1-y)^2) \tag{65}$$

$$\int_0^1 \int_0^1 k(x, y)dxdy = 1 + b/3 \tag{66}$$

As $b \to 0$, the optimal quadrature nodes are found numerically to be the midpoint nodes:

$$x_i = \frac{2i-1}{2n} \qquad i = 1, ..., n \tag{67}$$

for which the optimal weights are constant: $w_i = 1/n$. For larger b, corresponding to more expected discontinuities, the nodes remain equally spaced, but by a smaller amount, i.e. they move inward.

The process (64) is closely related to the Weiner process. In fact, we can get the Weiner process by using the asymmetric step function

$$(x-t)_+^0 = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases} \tag{68}$$

$$= \frac{d}{dx}(x-t)_+ \tag{69}$$

which yields

$$k(x,y) = 1 + b\int_0^1 (x-t)_+^0 (y-t)_+^0 dt \tag{70}$$

$$= 1 + b\min(x,y) = 1 + \frac{b}{2}(x+y) - \frac{b}{2}|x-y| \tag{71}$$

This is the process which arises from Wahba's derivation. It also gives piecewise linear interpolants, though it is nonstationary. Note that it is not exactly the same as the Weiner process: it has an additive constant which is crucial for penalizing discontinuities and making the interpolant "natural." This covariance function allows x and y in any range, not just $[0,1]$.

Another covariance function, that turns out to be related, is (Blight & Ott, 1975; Currin et al., 1991)

$$k(x,z) = \exp(-b|x-z|) + b \tag{72}$$

As $b \to 0$, the covariance function reduces to $k(x,z) \approx 1 - b|x-z|$ and the interpolants are piecewise linear as above. With this covariance function we also have the option of increasing b to make the interpolant flatter and nonlinear. As $b \to \infty$, the interpolant is almost everywhere equal to the constant $\frac{1}{n}\sum_i f(x_i)$.

The process (64) can be extended to multiple dimensions by analogy to the polynomial case—we take products of step functions along different dimensions, with step functions of higher "order" weighted by higher powers of b. The covariance function becomes the product over all dimensions:

$$k(\mathbf{x},\mathbf{y}) = \prod_{k=1}^{d}(1 + b - 2b|x_k - y_k|) \tag{73}$$

$$\int_0^1 k(\mathbf{x},\mathbf{y})d\mathbf{x} = \prod_{k=1}^{d}(1 + b - b(y_k^2 + (1-y_k)^2)) \tag{74}$$

$$\int_0^1\int_0^1 k(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} = (1 + b/3)^d \tag{75}$$

A typical interpolant is shown in figure 6. See Currin et al. (1991) for another derivation of this covariance function.

Figure 7 shows some optimal node arrangements computed for $d = 2$ and $b = 0.01$. The optimal nodes are unique for each n. As in the trigonometric case, the local optima satisfy projection

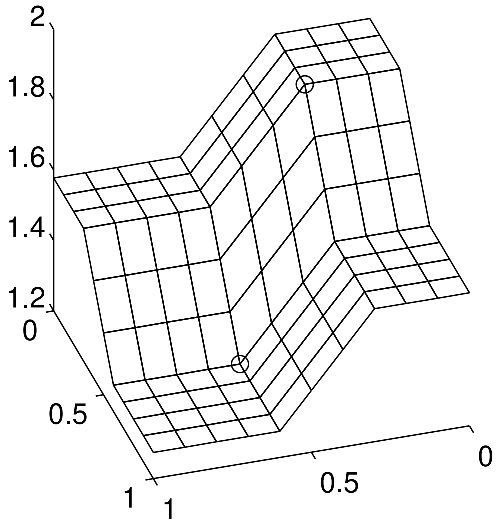


Figure 6: A piecewise-planar interpolant in two dimensions, produced by the process (73) from two data points (circled)

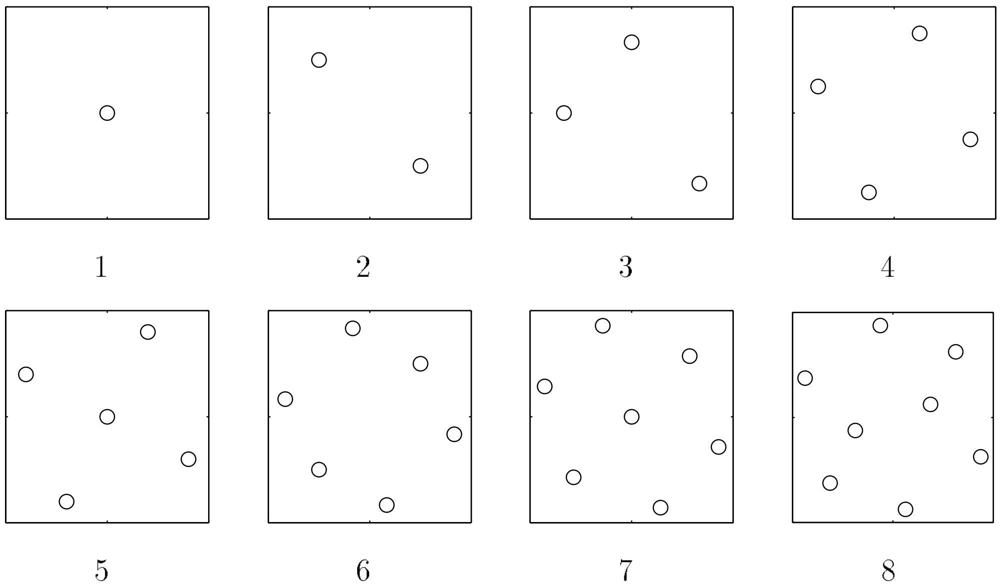


Figure 7: Optimal node arrangements for linear spline quadrature in the square $[0,1]^2$ for $n = 1, \ldots, 8$, obtained by the methods in this paper. This is the full set, up to symmetry transformations; no rules are better than or equal to these.

consistency: when projected onto one dimension, they produce the midpoint nodes (67). Here it is easy to see why this is so: step functions which involve one dimension are weighted by b in the optimization, while step functions involving two dimensions, i.e. cross terms, are weighted by b^2, giving them lower priority. This means we can reduce the search space by only considering projection-consistent rules. This reduction was used to generate figure 7. All of these rules have constant quadrature weights $w_i = 1/n$, which follows from projection consistency. Compared to the node arrangements for polynomial or trigonometric quadrature, these have the most even coverage of the square.

Another way to define piecewise-constant functions in multiple dimensions is to use a weighted sum of oriented step functions $\mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} - t)$, not just axis-parallel step functions. With this basis, it is convenient to use the unit ball $\mathbf{x}^{\mathrm{T}}\mathbf{x} \leq 1$ as the region A, instead of the unit square. The basis expansion with coefficients $a(\mathbf{w}, t)$ is

$$f(\mathbf{x}) = a_0 + \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} \int_{-1}^{1} a(\mathbf{w}, t)\ \mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} - t)\ dt d\mathbf{w} \tag{76}$$

If all step functions have constant weight b, then the covariance function for this basis is

$$\begin{aligned}
k(\mathbf{x}, \mathbf{y}) &= 1 + b \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} \int_{-1}^{1} \mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} - t)\mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{y} - t)\ dt d\mathbf{w} && (77)\\
&= 1 + b \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} 2 - 2|\mathbf{w}^{\mathrm{T}}\mathbf{x} - \mathbf{w}^{\mathrm{T}}\mathbf{y}|\ d\mathbf{w} && (78)\\
&= 1 + 2C_1 b - 2C_2 b\ |\mathbf{x} - \mathbf{y}| \qquad b > 0 && (79)\\
C_1 &= \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} 1\ d\mathbf{w} \qquad \text{(area of sphere)} && (80)\\
C_2 &= \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} |\cos \angle(\mathbf{w}, \mathbf{x} - \mathbf{y})|\ d\mathbf{w} = \int_{\mathbf{w}^{\mathrm{T}}\mathbf{w}=1} |w_1|\ d\mathbf{w} && (81)
\end{aligned}$$

This covariance function is related to isotropic Brownian motion and thin-plate splines (Wahba, 1990). The resulting interpolants and quadratures can be interpreted as averaging over oriented step edges. The analogous generalization of (72) is $k(\mathbf{x}, \mathbf{y}) = \exp(-b|\mathbf{x} - \mathbf{y}|) + b$, which is related to the Ornstein-Uhlenbeck process.

6.2 Cubic splines

Take the piecewise constant functions (61) and integrate them. The result is piecewise linear functions:

$$f(x) = a_1 + a_0 x + \int_0^1 a(t)(|x - t| - t)dt \tag{82}$$

whose covariance function is

$$\begin{aligned}
k(x, y) &= 1 + xy + b \int_0^1 (|x - t| - t)(|y - t| - t)dt && (83)\\
&= 1 + (1 + b)xy + \frac{b}{3}(|x - y|^3 - x^3 - y^3) && (84)
\end{aligned}$$

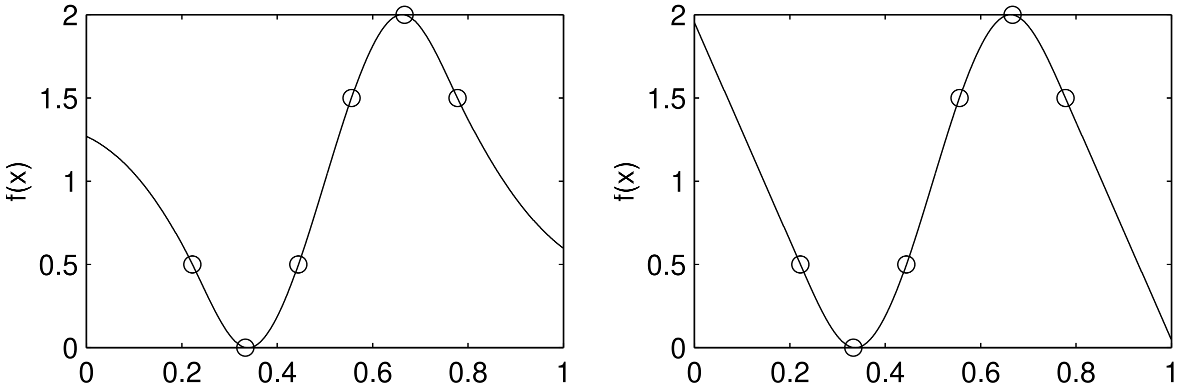


Figure 8: A piecewise cubic interpolant from the process (84) for $b = 5$ (left) and $b = 0.0001$ (right). The interpolant on the right corresponds to the natural cubic spline.

This result also follows from the fact that the covariance of $f(x) = \int g(x)dx$ is the double integral of the covariance of $g(x)$:

$$k_f(x, y) = E[f(x)f(y)] \tag{85}$$

$$= E[\int\int g(x)g(y)dydx] \tag{86}$$

$$= \int\int k_g(x, y)dydx \tag{87}$$

Apply (87) to the linear spline covariance function (64) to get (84). This relationship was also suggested by Diaconis (1988) and Currin et al. (1991).

The interpolants for this process are linear combinations of (84) and so are piecewise cubic. We can see that, in general, a process consisting of piecewise polynomials of order m will have interpolants which are piecewise polynomials of order $2m + 1$. The parameter b controls the boundary conditions; as $b \to 0$ we get a natural cubic spline (linear outside the data)—see figure 8.

Another approach is to integrate the Weiner process (70) to get cubic splines, as in Wahba's derivation. But not every piecewise cubic covariance function produces natural splines. For example, the stationary covariance $1 + b_1(x - y)^2 + b_2|x - y|^3$ given by Currin et al. (1991) does not produce natural splines, for any setting of the parameters, because it has extra x^2 and y^2 terms.

We can perform quadrature using cubic splines, which is average-case in the sense of averaging over possible discontinuities in the derivative of the function. The integral of (84) on $A = [0, 1]$ with $\mu(x) = 1$ is

$$\int_0^1 k(x, y)dx = 1 + \frac{1+b}{2}y + \frac{b}{12}(y^4 + (1 - y)^4 - 1 - 4y^3) \tag{88}$$

$$\int_0^1 \int_0^1 k(x,y)dxdy = 1 + \frac{1+b}{4} + \frac{b}{12}(\frac{1}{5} + \frac{1}{5} - 1 - 1) = \frac{5}{4} + \frac{7}{60}b \tag{89}$$

For $b = 0.0001$, the optimal quadrature nodes are found numerically to be equally spaced, but by more than $1/n$. In other words, they are more spread out than the optimal nodes in the linear case. Perhaps this is because end effects are more severe in the cubic case.

Multiple dimensions are handled just as in the linear case. We can either use

$$k(x,y) = \prod_{k=1}^{d} \left(1 + (1+b)x_k y_k + \frac{b}{3}(|x_k - y_k|^3 - x_k^3 - y_k^3)\right) \tag{90}$$

which is the double integral of (73), or we can try to generalize the isotropic covariance (79). The optimal node arrangements for (90) are not projection-consistent, which makes it more difficult to find them. The best arrangements found so far, via the Nelder-Mead simplex method, strongly resemble the arrangements in figure 7, except for some minor changes including a scale factor which pushes the nodes closer to the boundary. This agrees with the behavior in the one-dimensional case.

7 Summary and future work

The Gaussian process method has been shown to make it easier to design average-case and multidimensional quadrature rules. It can be used to make customized high-accuracy quadrature rules, in the style of Yarvin and Rokhlin (1998b), or to make general-purpose quadrature rules derived from interpolation schemes.

Since Gaussian processes are probabilistic models which can be estimated from data, a tantalizing possibility is to make an integration tool which learns over time how to best integrate the functions it is typically given. One approach is to parameterize the covariance function and learn the parameters from data (Blight & Ott, 1975; O'Hagan, 1991; Warnes & Ripley, 1987). In doing so, it is important to remember that interpolants represent *averages* over the process—not necessarily what is typical from that process (section 6.1).

References

Blight, B. J. N., & Ott, L. (1975). A Bayesian approach to model inadequacy for polynomial regression. *Biometrika*, *62*, 79–88.

Cools, R., & Rabinowitz, P. (1993). Monomial cubature rules since "Stroud": a compilation. *J Comp. Appl. Math.*, *48*, 309–326.

Cools, R., & Sloan, I. H. (1996). Minimal cubature formulae of trigonometric degree. *Mathematics of Computation*, *65*, 1583–1600.

Currin, C., Mitchell, T., Morris, M., & Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, *86*, 953–963.

Diaconis, P. (1988). Bayesian numerical analysis. *Statistical Decision Theory and Related Topics IV* (pp. 163–175).

Hammer, P. C., & Stroud, A. H. (1958). Numerical evaluation of multiple integrals II. *Math. Tables Aids Comput.*, *12*, 272–280.

Ma, J., Rokhlin, V., & Wandzura, S. (1996). Generalized Gaussian quadrature rules for systems of arbitrary functions. *SIAM J Numerical Analysis*, *33*, 971–996.

Novak, E., Ritter, K., Schmitt, R., & Steinbauer, A. (1999). On an interpolatory method for high dimensional integration. *J Computational and Applied Mathematics*, *112*, 215–228.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *J Statistical Planning and Inference*, *29*, 245–260.

Paskov, S. H. (1993). Average case complexity of multivariate integration for smooth functions. *Journal of Complexity*, *9*, 291–312.

Sloan, I. (1992). Numerical integration in high dimensions: the lattice rule approach. *Numerical integration: Recent developments, software and applications* (pp. 55–70).

Smale, S. (1985). On the efficiency of algorithms of analysis. *Bulletin of the American Mathematical Society*, *13*, 87–121.

Wahba, G. (1990). *Spline models for observational data.* Society for Industrial and Applied Mathematics.

Warnes, J. J., & Ripley, B. D. (1987). Problems with likelihood estimation of covariance functions of spatial Gaussian processes. *Biometrika*, *74*, 640–642.

Wissman, J. W., & Becker, T. (1986). Partially symmetric cubature formulas for even degrees of exactness. *SIAM J Numerical Analysis*, *23*, 676–685.

Yarvin, N., & Rokhlin, V. (1998a). `http://www.netlib.org/pdes/multipole/wts500.f`.

Yarvin, N., & Rokhlin, V. (1998b). Generalized Gaussian quadratures. *SIAM Journal on Scientific Computing*, *20*, 699–718.