RELIABLE ADAPTIVE, AUTOMATIC QUADRATURE FOR CONES OF INTEGRANDS

FRED J. HICKERNELL, MARTHA RAZO, AND SUNNY YUN

Abstract. Hi

1. Introduction

Although some mathematical problems can be solved using pencil and paper, many can only be solved via numerical algorithms. These algorithms typically generate a sequence of approximations that converge to the true solution. The user wants to know at what point in the sequence the approximation differs from the true solution by no more than a specified tolerance. Stopping rules used in practice are based either on inconvenient assumptions or on heuristics without guarantees. Here we describe a practical, adaptive trapezoidal rule for integration that is guaranteed to provide the correct answer. The key is to consider cones of integrands. We also suggest changing the way numerical quadrature is taught.

1.1. **Why Numerical Integration Is Needed?** There are many methods that we learn for evaluating integrals, e.g., substitution, integration by parts, partial fractions, etc. Unfortunately, there are many elementary functions whose integrals *cannot* be expressed as elementary functions. An important example is the standard normal probability density, $\phi : x \mapsto \mathrm{e}^{-x^2}/\sqrt{2\pi}$, whose integral is the standard normal distribution, $\Phi : x \mapsto \int_{-\infty}^{x} \phi(t)\,\mathrm{d}t$. Since the Φ is not an elementary function, values of Φ must be provided via tables in statistics textbooks and special functions in calculators. By contrast, the derivatives of elementary functions are always elementary functions.

1.2. **The Trapezoidal Rule.** To compute integrals that are not amenable to analytic methods one must turn to numerical integration (quadrature) methods such as the trapezoidal rule, T_n, which is often introduced in calculus courses:

$$\int_0^1 f(x)\,\mathrm{d}x \approx T_n(f) := \frac{1}{2n}\left[f(0) + 2f(1/n) + \cdots + 2f(1-1/n) + f(1)\right]. \tag{1}$$

The trapezoidal rule is the integral of the piecewise linear spline approximation to the integrand. The name of the rule comes from the fact that the integral of the linear spline is a sum of areas of trapezoids (see Figure **??**). Under mild smoothness conditions on f one observes that $T_n(f)$ approaches $\int_0^1 f(x)\,\mathrm{d}x$ as n increases.

For the user's convenience one would like to turn T_n into an *automatic* quadrature algorithm — one that automatically chooses n to ensure that the trapezoidal rule error is small enough. Given the user's tolerance for error, ε, an automatic

trapezoidal rule should choose n such that

$$\text{err}(f,n) := \left| \int_0^1 f(x)\,\mathrm{d}x - T_n(f) \right| \le \varepsilon. \tag{2}$$

Although $\text{err}(f,n)$ is rarely known exactly, there exist rigorous upper bounds on the error that are proportional to $1/n^2$ and the roughness of the integrand, e.g., Brass and Petras (2011, Sect. 7.2, (7.15)):

$$\text{err}(f,n) \le \frac{\text{Var}(f')}{8n^2}, \qquad n \in \mathbb{N} := \{1,2,\ldots\}. \tag{3}$$

Here $\text{Var}(\cdot)$ represents the (total) variation, which is defined as

$$\text{Var}(f) := \sup \left\{ \sum_{i=1}^{n} |f(x_i) - f(x_{i-1})| : n \in \mathbb{N},\ 0 \le x_0 < x_1 < \cdots < x_n \le 1 \right\}. \tag{4}$$

The trapezoidal rule gives the exact answer for linear integrands. Error bound (3) reflects this fact since if $f(x) = ax + b$, then $f'(x) = a$, and so $\text{Var}(f') = 0$.

To illustrate this error bound, consider the normal probability example mentioned above. The trapezoidal rule approximation using $n = 16$ trapezoids, the actual error, and the error bound are given as follows:

$$f(x) = \phi(x) = \frac{\mathrm{e}^{-x^2}}{\sqrt{2\pi}}, \qquad \int_0^1 f(x)\,\mathrm{d}x =, \qquad T_{16}(f) =??, \qquad \text{err}(f,16) =???.$$

(The exact value of the integral to four significant digits may be found by a variety of different quadrature rules.) The point to note is that ???. Figure **??** above with

Error bound (3) helps us determine how large n must be to meet the error tolerance, but unlike the above example, we do not know how large $\text{Var}(f')$ is for most f occurring in practice. Section 2 provides in the details for integrands lying in a prescribed ball. Section 3 describes stopping rule that is taught in elementary numerical analysis courses, but which is not guaranteed to succeed. A guaranteed stopping rule for integrands lying in cones is described in Section **??**.

2. An Automatic Algorithm for Balls of Integrands

If f has a simple form, one might be able to derive an upper bound on $\text{Var}(f')$ and turn error bound (3) into an automatic algorithm. The algorithm below is guaranteed to produce $\text{err}(f,n) \le \varepsilon$ provided that $\text{Var}(f') \le M_1$.

Automatic numerical algorithms for univariate integration appear in many widely used software packages, such as MATLAB (2013), Mathematica (2012), NAG (2012), and R (2012). Automatic algorithms determine adaptively where and how much to sample the integrand by estimating the algorithm error from the sampled function values. Unfortunately, such error estimation methods are unreliable, as pointed out by James Lyness (1983).

Here we show how to overcome Lyness's objections by employing a new paradigm for error bounds, which is based on cones, rather than balls, of integrands rather than balls. This article illustrates with greater detail the more general setting in Clancy et al. (2013). We argue that the traditional way of teaching error estimation for numerical integration should be changed to reflect this new paradigm. Moreover, this paradigm can, and should, be applied to other problems requiring automatic algorithms.

Algorithm 1. Given an error tolerance, ε, set

$$n = \left\lceil \sqrt{M_1/(8\varepsilon)} \right\rceil, \tag{5}$$

and return the trapezoidal rule approximation $T_n(f)$ as the answer.

For example, for ϕ defined above, $\mathrm{Var}(f') = 1/\sqrt{2\pi \mathrm{e}}$, and so one may choose any $n \geq 1/\sqrt{8\sqrt{2\pi \mathrm{e}}\,\varepsilon}$. However, not all integrands have first derivatives whose variations are so easy to bound.

3. The Flaws in the Prevalent Data-Based Error Estimate

Automatic quadrature algorithms do not require the user to specify M_1, but bound or estimate the error based solely on the function values, $f(x_i)$, and then determine the sample size accordingly. Here we focus on this error estimation process. As we demonstrate in this section, the popular error estimate for the trapezoidal rule is flawed.

Numerical methods texts, such as Burden and Faires (2010, p. 223–224), Cheney and Kincaid (2013, p. 233), and Sauer (2012, p. 270), advise readers to estimate the error of $T_n(f)$ in terms of its from $T_{n/2}(f)$, specifically,

$$\widehat{\mathrm{err}}(f,n) = \frac{\left|T_n(f) - T_{n/2}(f)\right|}{3}, \qquad \frac{n}{2} \in \mathbb{N}. \tag{6}$$

This algorithm leads to the following automatic and adaptive quadrature algorithm.

Algorithm 2. Given an error tolerance, ε, let $j = 1$ and $n_1 = 2$.

Stage 1: Compute the error estimate $\widehat{\mathrm{err}}(f, n_j)$ according to (6).

Stage 2: If $\widehat{\mathrm{err}}(f, n_j) \leq \varepsilon$, then return the trapezoidal rule approximation $T_{n_j}(f)$ as the answer.

Stage 3: Otherwise let $n_{j+1} = 2n_j$, increase j by one, and go to Stage 1.

By doubling the number of trapezoids for each iteration, the data from the previous iterations can be reused. Automatic quadrature algorithms in commonly used numerical libraries are more sophisticated than Algorithm 2, but their stopping criteria are similar in spirit and have the same intrinsic flaws.

While, Algorithm 2 requires no a priori bound on the variation of the first derivative of the integrand, there is no guarantee that $\mathrm{err}(f,n) \leq \varepsilon$. In fact, one may easily to construct integrands, f, such that this error estimate fails completely, e.g.,

$$\int_0^1 f(x)\,\mathrm{d}x = 1, \quad T_n(f) = T_{n/2}(f) = -1, \quad \mathrm{err}(f,n) = 2 \neq 0 = \widehat{\mathrm{err}}(f,n). \tag{7}$$

Figure 1 shows examples of two integrands satisfying these conditions. These two integrands illustrate two ways in which the error estimate $\widehat{\mathrm{err}}(\cdot)$, and Algorithm 2, can fail. One way is ultimately unavoidable but ought to be quantifiable. The other way is inexcusable, but is prevalent in widely used automatic numerical quadrature.

3.1. Spiky Integrands.

Any quadrature algorithm that depends only on function values to estimate error will be fooled by sufficiently spiky integrands. Suppose that the quadrature algorithm samples the integrand at a seqeunce of points, $\xi_1, \xi_2, \ldots$, and suppose that $f(\xi_i)$ is the same value for all ξ_i. Here the algorithm may even be adaptive, meaning that ξ_{i+1} may depend on $(\xi_1, f(\xi_1)), \ldots, (\xi_i, f(\xi_i))$. Eventually the algorithm stops at n points. Under mild conditions on the set of integrands of interest the best estimate of the integral is that constant.

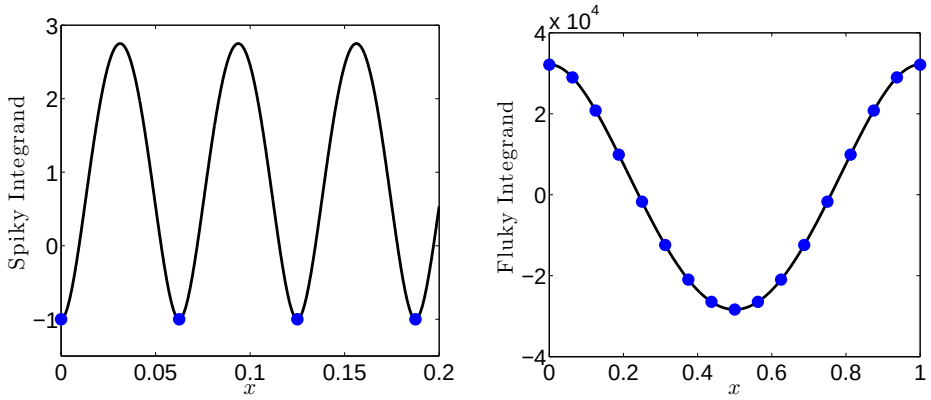

FIGURE 1. Examples of a spiky integrand (left) and a fluky integrand (right), which satisfy the failure conditions (7) for $n = 32$.

Figure 1 (left) plots the following spiky integrand that is defined in terms of the quartic Bernoulli polynomial:

$$f_{\text{spiky}}(x;n) = 1 + 60B_4(\{nx\}), \qquad n \in \mathbb{N}, \quad \{x\} := x \bmod 1. \tag{8}$$

The Bernoulli polynomials are described by Abramowitz and Stegun (1964, Chap. 23, `http://dlmf.nist.gov/24`). The first six are defined as follows

$$B_0(x) = 1, \qquad B_1(x) = x - 1/2, \qquad B_2(x) = 1/6 - x(1-x), \tag{9a}$$

$$B_3(x) = \frac{1}{2}x(1-x)(1-2x), \qquad B_4(x) = -1/30 + [x(1-x)]^2, \tag{9b}$$

$$B_5(x) = \frac{1}{6}x(1-x)(1-2x)(1+3x-3x^2). \tag{9c}$$

The integrand $f_{\text{spiky}}(\cdot;n)$ satisfies (7) for even n and has

- $n+1$ local minima of -1 at $x = 0, 1/n, \ldots, 1$, and
- n local maxima of $11/4$ at $x = 1/(2n), 3/(2n), \ldots, 1 - 1/(2n)$.

The problem of integrating spiky functions never goes away, but it should be quantifiable. An algorithm ought to integrate functions accurately, provided that they have spikes no thinner than say, δ. Moreover, the algorithm should be tunable so that the user can specifiy δ. We describe such an algorithm in Section 4, along with a precise definition of spike width.

3.2. **Fluky Integrands.** Thirty years ago James Lyness wrote a SIAM Review article, *When Not to Use an Automatic Quadrature Routine*. While recognizing the problem of spiky integrands, he claims that there is even a more serious problem with automatic quadrature methods (Lyness, 1983, p. 69):

> While prepared to take the risk of being misled by chance alignment of zeros in the integrand function, or by narrow peaks which are "missed," the user may wish to be reassured that for "reasonable" integrand functions which do not have these characteristics all will be well. It is the purpose of the rest of this section to demonstrate by example that he cannot be reassured on this point. In fact the

> routine is likely to be unreliable in a significant proportion of the problems it faces (say 1 to 5%) and there is no way of predicting in a straightforward way in which of any set of apparently reasonable problems this will happen.

Lyness went on to describe how to construct, what we would call a fluky integrand.

Figure 1 (right) plots a fluky integrand that is defined in terms of the quadratic and quartic Bernoulli polynomials:

$$f_{\text{fluky}}(x;n) = 1 - 15n^2[B_2(x) + n^2 B_4(x)], \qquad n \in \mathbb{N}. \tag{10}$$

The integrand $f_{\text{fluky}}(\cdot;n)$ satisfies (7) for even n and has only

- 1 local minimum of $(16 + 20n^2 - 7n^4)/16$ at $x = 1/2$, and
- 2 local maxima of $(19 - 10n^2 + 2n^4)/4$ at $x = (1 \pm \sqrt{1 - 2/n^2})/2$.

Integrands like this one are termed fluky by us because their construction is rather delicate. They satisfy conditions (7) without a large number of local optima. To illustrate this delicacy, note that removing either the B_2 or the B_4 term does not affect the value of the integral but does cause $\widehat{\text{err}}(f_{\text{fluky}}(\cdot;n),n)$ to be significantly different from zero.

The failure of the error estimate in (6) for $f_{\text{fluky}}(\cdot;n)$ seems avoidable. For even moderate n, the sampled function values, $\big(f_{\text{fluky}}(i/n;n)\big)_{i=1}^{n}$, are spread across a large range. This should indicate that true error, $\text{err}(f_{\text{fluky}}(\cdot;n),n)$, might be substantial, and far from the error estimate, $\widehat{\text{err}}(f_{\text{fluky}}(\cdot;n),n) = 0$. In Section 4 we transform this intuition into a better method for bounding the error of the trapezoidal rule.

3.3. **Necessary Conditions for Failure of the Error Estimate.** Integrands that satisfy (7) and fool the error estimate in (6) must satisfy certain conditions. First note that the error bound in (3) implies that

$$\text{Var}(f') \geq 8n^2 \,\text{err}(f,n) = 16n^2 \asymp n^2 \qquad \text{as } n \to \infty,$$

so the total variation of the first derivative must be increasingly large as the number of trapezoids increases. Moreover, the trapezoidal rule added to the error estimate in (6) is actually Simpson's rule:

$$\begin{aligned} S_n(f) &:= T_n(f) + \widehat{\text{err}}(f,n) = \frac{4T_n(f) - T_{n/2}(f)}{3} \\ &= \frac{1}{3n}\left[f(0) + 4f(1/n) + 2f(2/n) + 4f(3/n) + \cdots + 4f(1-1/n) + f(1)\right]. \end{aligned}$$

The error bound of Simpson's rule then serves as an bound on the error of the error estimate (Brass and Petras, 2011, Sect. 7.3, p. 231):

$$|\text{err}(f,n) - \widehat{\text{err}}(f,n)| = \left|\int_0^1 f(x)\,\mathrm{d}x - S_n(f)\right| \leq \frac{\text{Var}(f''')}{36n^4}, \qquad \frac{n}{2} \in \mathbb{N}. \tag{11}$$

So, integrands satisfying (7) must also satisfy

$$\text{Var}(f''') \geq 36n^4\,|\text{err}(f,n) - \widehat{\text{err}}(f,n)| = 36n^4 \asymp n^4 \qquad \text{as } n \to \infty.$$

Thus, the total variation of the third derivative must increase even faster than the minimum increase of $\text{Var}(f')$ as the number of trapezoids increases.

The derivatives of Bernoulli polynomials are multiples of lower degree Bernoulli polynomials and the integrals of nonconstant Bernoulli polynomials vanish (see Abramowitz and Stegun (1964, Chap. 23, `http://dlmf.nist.gov/24`)):

$$B_n'(x) = nB_{n-1}(x), \quad \int_0^1 B_{n+1}(x)\,\mathrm{d}x, \qquad n \in \mathbb{N}.$$

The formula for the derivatives facilitates the computation of the total variation of the spiky and fluky examples above, confirming that the variations of their first and third derivatives satisfy the above inequalities. For the spiky integrand defined in (8),

$$\mathrm{Var}(f_{\mathrm{spiky}}(\cdot;n)) = \frac{15n}{2}, \qquad \mathrm{Var}(f'_{\mathrm{spiky}}(\cdot;n)) = \frac{80n^2}{\sqrt{3}} > 16n^2, \tag{12a}$$

$$\mathrm{Var}(f''_{\mathrm{spiky}}(\cdot;n)) = 360n^3, \qquad \mathrm{Var}(f'''_{\mathrm{spiky}}(\cdot;n)) = 1440n^4 > 36n^4. \tag{12b}$$

For the fluky integrand defined in (10),

$$\mathrm{Var}(f_{\mathrm{fluky}}(\cdot;n)) = \frac{15}{8}(8 - 4n^2 + n^4), \tag{13a}$$

$$\mathrm{Var}(f'_{\mathrm{fluky}}(\cdot;n)) = \frac{10n}{3}\left[9n + 2\sqrt{3(-2+n^2)^3}\right] > 16n^2, \tag{13b}$$

$$\mathrm{Var}(f''_{\mathrm{fluky}}(\cdot;n)) = 90n^4, \qquad \mathrm{Var}(f'''_{\mathrm{fluky}}(\cdot;n)) = 360n^4 > 36n^4. \tag{13c}$$

For the spiky integrand the asymptotic order of the total variation for large n increases with the order of the derivative while for the fluky integrand it stays the same. Specifically,

$$\mathrm{Var}(f^{(j)}_{\mathrm{spiky}}(\cdot;n)) \asymp n^{j+1}, \quad \mathrm{Var}(f^{(j)}_{\mathrm{fluky}}(\cdot;n)) \asymp n^4, \qquad j = 0, \ldots, 3.$$

This difference in asymptotic behaviors distinguishes whether a better error bound developed in the next section will be successful: no for the spiky integrand , but yes for the fluky integrand.

4. A Guaranteed Quadrature Algorithm

There are two paths that one might take in constructing a rigorous *data-based* error bound for the trapezoidal rule. Since the error in the error estimate, $\mathrm{err}(f,n) - \widehat{\mathrm{err}}(f,n)$ is proportional to $\mathrm{Var}(f''')$, one might try to construct or assume a bound on $\mathrm{Var}(f''')$, but this is even more difficult than bounding the term $\mathrm{Var}(f')$ that occurs in the original error bound, $\mathrm{err}(f,n)$. Estimating $\mathrm{Var}(f''')$ from data requires knowledge about even higher order derivatives of f. The perhaps less obvious, but successful, approach is to estimate a weaker semi-norm than $f \mapsto \mathrm{Var}(f')$.

Let A_n denote the linear spline approximation using $n+1$ equally spaced points, i.e.,

$$A_n(f)(x) = \begin{cases} f(0)(1/n - x) + f(1/n)x, & 0 \le x < 1/n, \\ f(1/n)(2/n - x) + f(2/n)(x - 1/n), & 1/n \le x < 2/n, \\ \qquad\vdots & \qquad\vdots \\ f(1-1/n)(1-x) + f(1)(x - 1 + 1/n), & 1 - 1/n \le x \le 1. \end{cases} \tag{14}$$

The trapezoidal rule can be written as $T_n(f) = \int_0^1 A_n(f)(x)\,\mathrm{d}x$. Using this definition we focus on the semi-norm defined by $f \mapsto \mathrm{Var}(f - A_1(f))$. Like $f \mapsto \mathrm{Var}(f')$ this weaker semi-norm also vanishes for functions that are linear in x.

One may approximate $\mathrm{Var}(f)$ well by $\mathrm{Var}(A_n(f))$ provided that $\mathrm{Var}(f')$ is not too large. For finite $\mathrm{Var}(f')$ it follows that $\mathrm{Var}(f) = \|f'\|_1$. This facilitates an bounds on $\mathrm{Var}(f) - \mathrm{Var}(A_n(f))$:

$$\begin{aligned}\mathrm{Var}(f) - \mathrm{Var}(A_n(f)) &= \int_0^1 |f'(x)|\,\mathrm{d}x - \sum_{i=1}^n |f(i/n) - f((i-1)/n)| \\ &= \sum_{i=1}^n \int_{(i-1)/n}^{i/n} [|f'(x)| - n\,|f(i/n) - f((i-1)/n)|]\,\mathrm{d}x \\ &\le \sum_{i=1}^n \int_{(i-1)/n}^{i/n} |f'(x) - n[f(i/n) - f((i-1)/n)]|\,\mathrm{d}x.\end{aligned}$$

The second right hand side above must be no less than zero, so $\mathrm{Var}(f) - \mathrm{Var}(A_n(f)) \ge 0$. Using integration by parts it can be verified that

$$f'(x) - n[f(i/n) - f((i-1)/n)] = (n-1)\int_{(i-1)/n}^{i/n} w_i(t,x) f''(t)\,\mathrm{d}t,$$

where

$$w_i(t,x) = \begin{cases} t - (i-1)/n, & (i-1)/n \le t \le x \le i/n, \\ t - i/n, & (i-1)/n \le x < t \le i/n. \end{cases}$$

Switching the order of integration then leads to the upper bound

$$\begin{aligned}&\mathrm{Var}(f) - \mathrm{Var}(A_n(f)) \\ &\quad\le (n-1)\sum_{i=1}^n \int_{(i-1)/n}^{i/n}\int_{(i-1)/n}^{i/n} |w(t,x)||f''(t)|\mathrm{d}t\,\mathrm{d}x \\ &\quad\le (n-1)\sum_{i=1}^n \int_{(i-1)/n}^{i/n} 2[t - (i-1)/n](i/n - t)|f''(t)|\,\mathrm{d}t \\ &\quad\le (n-1)\sum_{i=1}^n \max_{(i-1)/n\le t\le i/n} |2(t-x_i)(i/n-t)| \int_{(i-1)/n}^{i/n} |f''(t)|\,\mathrm{d}t \\ &\quad\le \sum_{i=1}^n \frac{1}{2(n-1)} \int_{(i-1)/n}^{i/n} |f''(t)|\,\mathrm{d}t.\end{aligned}$$

This implies the following upper bound on a piece of $\|f'\|_1$:

$$\begin{aligned}&\int_{(i-1)/n}^{i/n} |f'(x) - A_n(f)'(x)|\,\mathrm{d}x \\ &\quad\le (n-1)\int_{(i-1)/n}^{i/n}\int_{(i-1)/n}^{i/n} |w(t,x)||f''(t)|\mathrm{d}t\,\mathrm{d}x \\ &\quad\le (n-1)\int_{(i-1)/n}^{i/n} 2[t - (i-1)/n](i/n - t)|f''(t)|\,\mathrm{d}t \\ &\quad\le (n-1)\max_{(i-1)/n\le t\le i/n} |2(t-x_i)(i/n-t)| \int_{(i-1)/n}^{i/n} |f''(t)|\,\mathrm{d}t \\ &\quad\le \frac{1}{2(n-1)} \int_{(i-1)/n}^{i/n} |f''(t)|\,\mathrm{d}t.\end{aligned}$$

At this point it is worth reminding ourselves that algorithms, such as the trapezoidal rule, $T_n(f)$, Simpson's rule, and any automatic quadrature rule that one can imagine, are based on limited information about the integrand, namely, a finite number of function values.

5. Acknowledgements

The authors are grateful for discussions with a number of colleagues. This research is supported in part by grant NSF-DMS-1115392.

References

Abramowitz, M. and I. A. Stegun (eds.) 1964. *Handbook of mathematical functions with formulas, graphs and mathematical tables*, U. S. Government Printing Office, Washington, DC.

Brass, H. and K. Petras. 2011. *Quadrature theory: the theory of numerical integration on a compact interval*, First, American Mathematical Society, Rhode Island.

Burden, R. L. and J. D. Faires. 2010. *Numerical analysis*, Ninth, Cengage Brooks/Cole, Belmont, CA.

Cheney, W. and D. Kincaid. 2013. *Numerical mathematics and computing*, Seventh, Brooks/Cole, Boston.

Clancy, N., Y. Ding, C. Hamilton, F. J. Hickernell, and Y. Zhang. 2013. *The complexity of guaranteed automatic algorithms: Cones, not balls*. submitted for publication, arXiv.org:1303.2412 [math.NA].

Lyness, J. N. 1983. *When not to use an automatic quadrature routine*, SIAM Rev. **25**, 63–87.

R Development Core Team. 2012. *The R Project for Statistical Computing*.

Sauer, T. 2012. *Numerical analysis*, Pearson.

The MathWorks, Inc. 2013. *MATLAB 8.1*, Natick, MA.

The Numerical Algorithms Group. 2012. *The NAG library*, Mark 23, Oxford.

Wolfram Research Inc. 2012. *Mathematica 9*, Champaign, IL.