

The race for high order Gauss–Legendre quadrature

Alex Townsend

A typical quadrature rule is the approximation of a definite integral by a finite sum of the form

$$\int_{-1}^1 f(x) dx \approx \sum_{j=1}^n w_j f(x_j), \quad (1)$$

where x_1, \dots, x_n and w_1, \dots, w_n are referred to as the quadrature *nodes* and *weights*, respectively. In 1814 Gauss [3] described a particularly ingenious choice for the nodes and weights that is optimal in the sense that for each n it exactly integrates polynomials of degree $2n - 1$ or less. It can be shown that no other quadrature rule with n nodes can do this or better. Today we call this Gauss–Legendre quadrature due to the pioneering work of Jacobi that showed the nodes are the zeros of the degree n Legendre polynomial $P_n(x)$ and $w_k = 2(1 - x_k^2)^{-1} [P'_n(x_k)]^{-2}$.

There is a catch. For large n there is no explicit closed-form expression for the Gauss–Legendre nodes or weights. And Gauss knew this. To demonstrate it practically he calculated (by hand!) the nodes and weights to 16 digits for $n = 7$. Ever since then, and especially since the advent of the modern computer, there has been in hindsight a race for computing the nodes and weights for larger and larger n to more and more digits. A race that the famous Golub–Welsch algorithm never led. Instead, it is Ignace Bogaert from Ghent University that has a new and winning algorithm. Here is the race report (see Figure 1).

Hand calculations led the way for over a century. Tallquist (1905), Moors (1905), Nyström (1930), and Bayly (1938) used fountain pens and dogged determination to calculate the quadrature nodes for $n \leq 12$. Eventually, with presumably a small army of human calculators Lowan, Davids, and Levenson (1942) tabulated the nodes and weights for $1 \leq n \leq 16$ for the Mathematical Tables Project.

A decade later computers were beginning to dominate over tedious hand calculations and large tabulations of nodes and weights made profit. The most popular algorithm for computing Gauss nodes was the Newton–Raphson method for finding the roots of $P_n(x)$ with evaluation of P_n and P'_n using a three-term recurrence. Huge strides were made. Gawlik (1958) briefly led with $n = 64$ before Davis and Rabinowitz (1958) got $n = 96$, and finally Stroud and Secrest (1966) achieved $n = 512$. This was the golden age for Gauss–Legendre

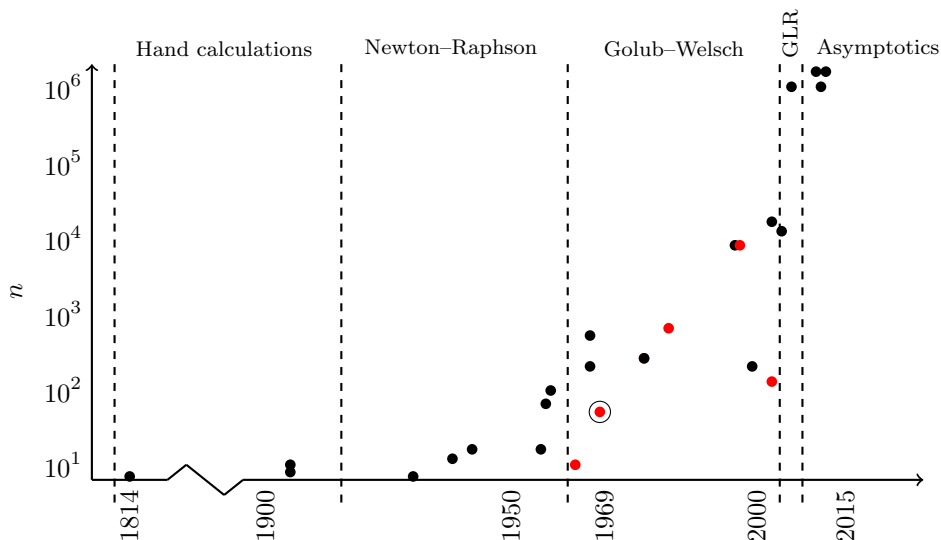


Figure 1: The 100 year race for high order Gauss–Legendre quadrature. A dot represents published work, located at the publication year and the largest Gauss–Legendre rule reported therein. A red dot is a paper based on variants of the Golub–Welsch algorithm. The dot for Golub and Welsch (1969) is circled. For a list of the papers used see <http://math.mit.edu/~ajt/GaussQuadrature/>.

quadrature.

By the 1960s orthogonal algorithms for eigenproblems was hot-off-the-press and Gene Golub was becoming famous. The Golub–Welsch algorithm [5] — featuring both the QR algorithm and Golub — was for the times. It quickly overshadowed the work by Rutishauser in 1963 that preceded it. However, contrary to popular belief the Golub–Welsch algorithm is not, and was not, the state-of-the-art algorithm for computing Gauss–Legendre quadrature rules in terms of accuracy and speed. Yet, by elegantly bringing together eigenproblems and Gauss quadrature, it radically changed how the world computed integrals. Before 1969, a few would compute quadratures by carefully extracting the tabulated values from Stroud and Secrest (1966) and calculating (1). After 1969, every practitioner was computing Gauss nodes and weights for themselves. Tabulations were already falling out of favor across the computational sciences, but the Golub–Welsch algorithm made it a reality for Gauss nodes and weights. This makes 1969 a year to remember for more than just the moon landing.

In the years that followed only a handful of experts noticed the work by Lether (1978), Yakimiw (1996), Petras (1999), and Swartztrauber (2003) on improving the details of the Newton–Raphson approach. While the Golub–Welsch algorithm was computing a few hundred nodes and weights, the Newton–Raphson approach was computing thousands of them. Many are still unaware of

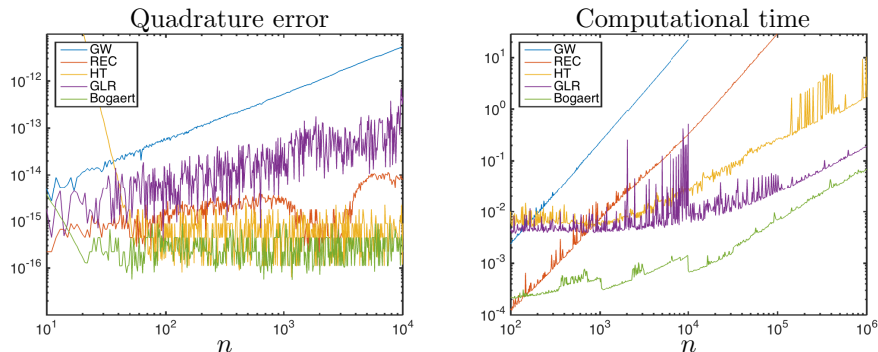


Figure 2: Quadrature error (left) and computational time (right) for Gauss–Legendre nodes and weights computed by the Golub–Welsch algorithm [5] (GW), Newton–Raphson with three-term recurrence (REC), Newton–Raphson with asymptotic formulas [6] (HT), the Glaser–Lui–Rokhlin algorithm [4] (GLR), and Bogaert’s formulas [1] (Bogaert). The timings here are for implementations in different programming languages and cannot be used for direct comparisons.

the developments after 1969 and have concluded that Gauss–Legendre quadrature is not computationally feasible for large n . Attention has shifted to adaptive and piecewise quadrature schemes.

In 2007 Glaser, Lui, and Rokhlin described a ground-breaking algorithm that can compute one million quadrature nodes in a handful of seconds [4]. Accolades should have followed, but it awakened too little interest. A few years later Bogaert, Michiels, and Fostier [2] and Hale and Townsend [6] showed that the Newton–Raphson method for finding the roots of $P_n(x)$ with careful evaluation of P_n and P'_n by asymptotic formulas could be just as fast and with a better accuracy than the world had seen before.¹ The golden age had returned. Figure 2 shows the quadrature error (see ϵ_{quad} in [6] for the exact definition) and the timings for five historically important algorithms. It was after careful numerical comparisons like these that the race was fully realized.

The epilogue was written by Bogaert a few months ago [1]. He derived explicit asymptotic formulas for the Gauss–Legendre nodes and weights that are accurate to 16 digits for any $n \geq 20$. Using his formulas I just computed one billion and two Gauss–Legendre nodes and weights on my laptop. This is a world record! So large is this rule that neighboring nodes near ± 1 are identical to 15 decimal places. One thousand nodes now takes less than a millisecond and one million less than a tenth of a second. Ignace Bogaert is the winner of this 100 year race. Bravo.

It was a fun race with a deserved winner. We are now searching for applications for which thousands of nodes and weights are needed. Our algorithms are poised. If you have an application in mind then please email ajt@mit.edu.

¹See [6] for computing Gauss–Jacobi, Gauss–Lobatto, and Gauss–Radau quadratures.

One million Gauss–Legendre nodes and weights. No problem. But why?

Acknowledgements

I thank Ignace Bogaert and Nick Hale for helpful comments and suggestions. I also thank Nick Trefethen for encouraging me to write this article.

References

- [1] I. BOGAERT, *Iteration-free computation of Gauss–Legendre quadrature nodes and weights*, SIAM J. Sci. Comput., 36 (2014), pp. C1008–C1026.
- [2] I. BOGAERT, B. MICHIELS, AND J. FOSTIER, *$\mathcal{O}(1)$ computation of Legendre polynomials and Gauss–Legendre nodes and weights for parallel computing*, SIAM J. Sci. Comput., 34 (2012), pp. C83–C101.
- [3] C. F. GAUSS, *Methodus nova integralium valores per approximationem inveniendi*, Comment. Soc. Reg. Scient. Gotting. Recent., (1814).
- [4] A. GLASER, X. LIU, AND V. ROKHLIN, *A Fast algorithm for the calculation of the roots of special functions*, SIAM J. Sci. Comput., 29 (2007), pp. 1420–1438.
- [5] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.
- [6] N. HALE AND A. TOWNSEND, *Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights*, SIAM J. Sci. Comput., 25 (2013), pp. A652–A674.