International Journal of Numerical Methods and Applications
Volume 5, Number 1, 2011, Pages 17-29
Published Online: July 26, 2011
This paper is available online at http://pphmj.com/journals/ijnma.htm

NUMERICAL APPROXIMATION OF THE FERMI-DIRAC INTEGRAL OF ORDER $\frac{1}{2}$ BY MEANS OF COMPOSITE GAUSS-LEGENDRE QUADRATURE

J. S. C. PRENTICE

Department of Applied Mathematics
University of Johannesburg
P. O. Box 524, Auckland Park, 2006
South Africa
e-mail: jprentice@uj.ac.za

Abstract

We approximate the Fermi-Dirac integral $\int_0^\infty \frac{\sqrt{t}}{e^{t-x}+1}\,dt$, by means of composite ten-point Gauss-Legendre quadrature, for values of x in the range $x \in [-100, 100]$. For any x, the integral is approximated by composite quadrature on the interval $0 \le t \le 150$, which is subdivided into a number of subintervals. We achieve a relative error of no more than 10^{-3} with 10 subintervals, and an error of no more than 10^{-14} with 89 subintervals. On our computational platform, the real-time duration of the computation is faster than two milliseconds for any $x \in [-100, 100]$.

1. Introduction

Semiconductor device simulations typically require the evaluation of the Fermi-

2010 Mathematics Subject Classification: 65Z05, 65Y99, 82D37.

Keywords and phrases: Fermi-Dirac integral, numerical approximation, Gauss-Legendre, composite quadrature.

Received April 25, 2011

Dirac integral of order $\frac{1}{2}$

$$\mathcal{F}_{\frac{1}{2}}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^\infty \frac{\sqrt{t}}{e^{t-x}+1}\, dt, \tag{1}$$

where

$$x \equiv \frac{E_F - E_C}{k_B T} \quad \text{or} \quad x \equiv \frac{E_V - E_F}{k_B T}. \tag{2}$$

This integral is used to determine the concentration of electrons in the conduction band (using the first definition of x in (2)), and the concentration of holes in the valence band (using the second definition of x in (2)), assuming a parabolic band structure [1]. In (2), E_F is the Fermi level, E_C is the bottom of the conduction band, E_V is the top of the valence band, k_B is Boltzmann's constant, and T is the absolute temperature. For semiconductors, the range of x is reasonably set at $[-100, 50]$, although in this work, we will be somewhat more ambitious and consider the range $[-100, 100]$. In any case, we will see that, for high accuracy, the efficiency of our approximation is limited by the negative values of x.

Many approximation techniques for $\mathcal{F}_{\frac{1}{2}}(x)$ have been reported in the literature - series expansion [2], quadrature [3], interpolation of tabular data, and polynomial- and rational approximation [4, 5]. To the best of our knowledge, composite Gauss-Legendre quadrature has not been used to approximate $\mathcal{F}_{\frac{1}{2}}(x)$, and it is such an approach that we wish to investigate here.

2. Composite Gauss-Legendre Quadrature

We briefly describe essentials relating to ten-point Gauss-Legendre quadrature and composite ten-point Gauss-Legendre quadrature (see Kincaid and Cheney [6] for a general discussion of GL quadrature).

2.1. Ten-point Gauss-Legendre quadrature

Ten-point Gauss-Legendre quadrature (GL10) on the interval $[-1, 1]$ is given by

$$\int_{-1}^{1} f(t)\, dt \approx \sum_{i=1}^{10} c_i f(\tilde{t}_i).$$

Here, the ten nodes $\tilde{t}_i$ are the roots of the Legendre polynomial $P_{10}(t)$ on the interval $[-1, 1]$, and c_i are appropriate weights. On an arbitrary interval $[a, b]$, GL10 is

$$\int_a^b f(t)\,dt \approx \frac{(b-a)}{2}\sum_{i=1}^{10} c_i f(t_i). \tag{3}$$

We have used the symbol t_i for the nodes on $[a, b]$ to differentiate from the nodes $\tilde{t}_i$ on $[-1, 1]$; the relationship between these two sets of nodes is given by the linear map

$$t_i = \frac{1}{2}[(b-a)\tilde{t}_i + b + a]. \tag{4}$$

For the sake of reference, the nodes $\tilde{t}_i$ and weights c_i for GL10 are given in the appendix.

The approximation error in GL10 on $[a, b]$ is

$$\Delta_{10} = \frac{f^{(20)}(\xi)}{20!}\int_a^b [P_{10}(t)]^2\,dt, \tag{5}$$

where $\xi \in (a, b)$, and t is determined from the map (4).

2.2. Composite GL10

Consider now the integral

$$\int_A^B f(t)\,dt, \tag{6}$$

where $[A, B]$ is a relatively large interval. Composite GL10 quadrature (CGL10) on this interval involves subdividing $[A, B]$ into N subintervals (let us use $[a, b]$ as a generic symbol for each of these subintervals), and then implementing GL10, as in (3), on each $[a, b]$. The integral (6) is then approximated by the sum of these N quadratures. The virtue of composite quadrature is, first, that the order of the

quadrature formula is fixed (ten, in this case), and the quality of the approximation is dictated exclusively by the number of subintervals N; and second, the value of N necessary for a desired accuracy can often be determined *a priori*.

The approximation error in CGL10 can be shown to be

$$\Delta_{\mathrm{CGL10}} \propto D^{20} \max_{[A,\,B]} | f^{(20)}(x) |, \tag{7}$$

where D is the length of each of the N subintervals into which $[A, B]$ is subdivided (see appendix). We also note here that, for

$$f(t) = \frac{2}{\sqrt{\pi}} \left(\frac{\sqrt{t}}{e^{t-x} + 1} \right),$$

$f^{(20)}(t)$ contains a term proportional to

$$\frac{1}{(\sqrt{t})^{39}(e^{t-x} + 1)}, \tag{8}$$

in addition to many other terms of similar character (in the sense of being inversely proportional to a power of $\sqrt{t}$).

3. Transformation on $[0, 1]$

When using CGL10 to approximate $\mathcal{F}_{\frac{1}{2}}(x)$, there is an important point to consider. From (7), we see that as $D \to 0$, $\Delta_{\mathrm{CGL10}} \to 0$. However, on every subinterval on $[0, 1)$, $\sqrt{t} < 1$, so that on these subintervals, the terms in $f^{(20)}(t)$ such as that in (8) can be very large. This means that D has to be made very small to offset the large value of the derivative in (7). This naturally results in the first subinterval $[0, b]$ being very small. On such subinterval, because the right endpoint tends to zero as D is made smaller, even the smallest value of $f^{(20)}(t)$ becomes very large. The net result is that a very large number of subintervals is necessary to achieve even a moderate level of accuracy in the approximation of the integral.

The remedy lies in making a suitable change of variable for which $f^{(20)}$ does not diverge on $[0, 1]$. Such a transformation is given by

$$t = v^2, \quad \sqrt{t} = v, \quad dt = 2v dv \tag{9}$$

so that

$$\frac{2}{\sqrt{\pi}} \int_0^1 \frac{\sqrt{t}}{e^{t-x}+1}\, dt = \frac{2}{\sqrt{\pi}} \int_0^1 \frac{2v^2}{e^{v^2-x}+1}\, dv \equiv \int_0^1 g(v)\, dv.$$

The twentieth derivative of $g(v)$ contains no terms inversely proportional to powers of v, and so does not diverge on $[0, 1]$.

Thus, we have

$$\mathcal{F}_{\frac{1}{2}}(x) = \frac{2}{\sqrt{\pi}} \int_0^1 \frac{2v^2}{e^{v^2-x}+1}\, dv + \frac{2}{\sqrt{\pi}} \int_1^\infty \frac{\sqrt{t}}{e^{t-x}+1}\, dt \tag{10}$$

as the integral that we seek to approximate.

4. Determining the Cutoff

We deal with the upper limit of ∞ in (10) by replacing it with a finite limit, which we refer to as a *cutoff*. The cutoff t_c is easily determined by numerical experimentation. For our computational platform (Matlab R13, Windows XP Pro, Intel Celeron M, machine precision $\sim 10^{-16}$), we found that $t_c = 150$ gave results identical (to 15 digits) with those obtained for $t_c = 200$, for all $x \in [-100, 100]$, and so we have chosen a cutoff of $t_c = 150$ for our algorithm. In the numerical calculations used to find this cutoff, we used $N = 300$ subintervals on each of $[0, 1]$ and $[1, t_c]$ having determined that 300 subintervals on each of these intervals gave results identical to those obtained with greater numbers of subintervals (again, to 15 digits), for all $x \in [-100, 100]$. It must be noted that $N = 300$ on each interval $[0, 1]$ and $[1, t_c]$ is not optimal and, as we shall see later, smaller values of N can be tolerated, even for high levels of accuracy. However, in determining t_c,

we chose to err on the side of caution; hence, the large value of N used in this numerical experiment. We must acknowledge that the cutoff we have determined is dependent on our machine precision; a machine of higher precision might require a larger cutoff. Also, the range of x-values could affect the cutoff, but we have not investigated this effect outside the interval $x \in [-100, 100]$.

5. Numerical Calculations

Let us denote our CGL10 approximations to the two integrals in (10) by

$$Q[x;\, g;\, N_1] \approx \frac{2}{\sqrt{\pi}} \int_0^1 \frac{2v^2}{e^{v^2 - x} + 1}\, dv,$$

$$Q[x;\, f;\, N_2] \approx \frac{2}{\sqrt{\pi}} \int_1^{t_c} \frac{\sqrt{t}}{e^{t - x} + 1}\, dt,$$

$$\mathcal{F}_{\frac{1}{2}}(x) \approx Q[x;\, g;\, N_1] + Q[x;\, f;\, N_2] \equiv Q[x;\, g + f;\, N_1,\, N_2],$$

where N_1 and N_2 denote the number of subintervals used in each approximation, and we have defined $Q[x;\, g + f;\, N_1,\, N_2]$ as our approximation to $\mathcal{F}_{\frac{1}{2}}(x)$. So the approximations used to find the cutoff t_c previously are $Q[x;\, g;\, 300]$ and $Q[x;\, f;\, 300]$. We will take these approximations as the most accurate we can achieve on our platform (we consider them accurate to 15 places, as discussed in the previous section), and we will use them to measure the accuracy in approximations with fewer subintervals. Indeed, the relative error in such an approximation, for any $x \in [-100, 100]$, is determined from

$$\Delta_R(x;\, N_1,\, N_2) = \frac{Q[x;\, g + f;\, N_1,\, N_2] - Q[x;\, g + f;\, 300,\, 300]}{Q[x;\, g + f;\, 300,\, 300]}. \tag{11}$$

We have determined N_1 and N_2 for various upper bounds (tolerances) on

$$\Delta_M \equiv \max_{x \in [-100, 100]} \left| \Delta_R(x;\, N_1,\, N_2) \right|,$$

and these values are shown in Table 1.

Table 1. Minimum values of N_1 and N_2 such that Δ_M is less than the indicated tolerance (first and fourth column) for all $x \in [-100, 100]$

$\log_{10}(\Delta_M)$	N_1	N_2	$\log_{10}(\Delta_M)$	N_1	N_2
–2	1	6	–9	1	32
–3	1	9	–10	1	40
–4	1	13	–11	1	49
–5	1	16	–12	1	60
–6	1	19	–13	1	72
–7	1	23	–14	2	87
–8	1	27			

The values of N_1 and N_2 in this table are the minimum values such that $\log_{10}(\Delta_M)$ is not greater than the corresponding value in the first and fourth columns. It is clear that only one or two subintervals are required for $Q[x; g; N_1]$, and that $Q[x; f; N_2]$ requires an increasing amount of subintervals for stricter tolerances, as expected. Of course, it is also clear that higher accuracy would require more subintervals, but we have not considered tolerances beyond 10^{-14} because we consider $Q[x; g + f; 300, 300]$ to be the best approximation we can achieve on our computational platform, and it is accurate to about 10^{-16} in relative error. We feel that to use $Q[x; g + f; 300, 300]$ to attempt to measure relative errors smaller than 10^{-14} would be unreliable, because such measurement would most likely be contaminated by the error present in $Q[x; g + f; 300, 300]$. Nevertheless, we regard the data in Table 1 to be reliable for any machine with precision $\sim 10^{-16}$. In Figure 1, we show the relative errors (11) for the selected approximations $Q[x; g + f; N_1, N_2]$, as functions of x.

In the introduction, we stated that the efficiency of the approximation, for high accuracy, is limited by the negative values of x. This is in evidence in the lower plots in Figure 1, where we see that the indicated values of N_1 and N_2 give errors for positive values of x that are much less than the errors for negative values of x.

We have also been able to measure the physical time taken to compute $Q[x;\, g + f;\, N_1,\, N_2]$, for any value of x, on our platform. Times relative to that for a tolerance of 10^{-2} are shown in Figure 2. The actual time required at this tolerance, i.e., to compute $Q[x;\, g + f;\, 1,\, 6]$ is 1.02ms. For a tolerance of 10^{-14}, approximately 80% more computing time is required. Of course, computing times vary from platform to platform, and are also dependent on the structure and compilation of code, and so we present this result only for the sake of academic interest.

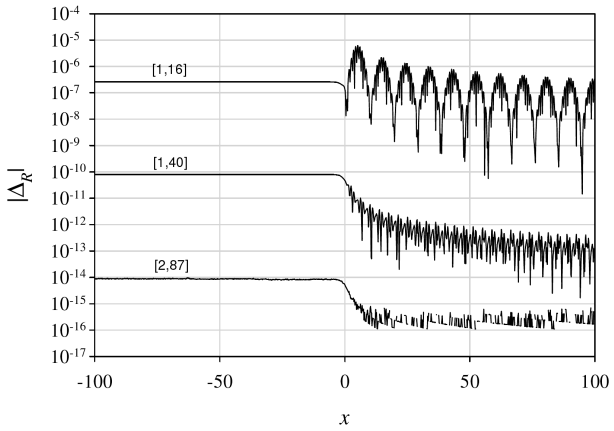

Figure 1. Relative error $|\Delta_R|$ vs x, for selected values of $[N_1,\, N_2]$, as indicated in the brackets.

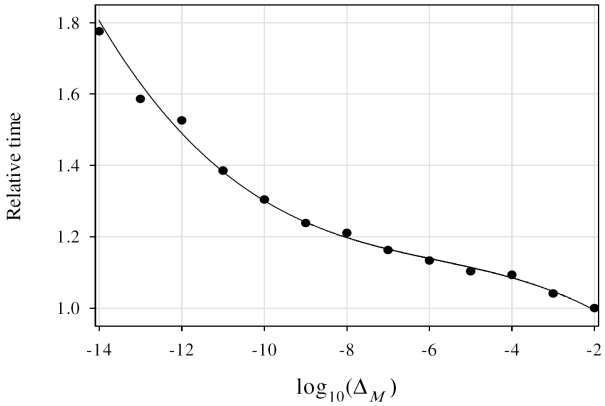

Figure 2. Relative computing times for indicated values of $\log_{10}(\Delta_M)$. The solid line is merely a visual guide.

5.1. Structure of the program

Here we give a brief sketch of our program, indicating its *vectorized* nature. Say we require $[N_1, N_2]$ nodes for the approximation, for a given x. Let V_i, $i = 1, 2, ..., N_1$ denote the row vector of nodes on subinterval i of $[0, 1]$. Of course, there are ten elements in each V_i. The row vector

$$\mathbf{V} \equiv [V_1\ V_2 \cdots V_{N_1}]$$

thus contains all the nodes necessary for composite quadrature on $[0, 1]$, in increasing order (there are $10N_1$ nodes in V). Compute the row vector

$$\mathbf{G} \equiv g(\mathbf{V}).$$

This is simply the function $g(v)$ evaluated at each node in **V**.

Now, let W_i denote the row vector of quadrature weights appropriate to subinterval i on $[0, 1]$, and let

$$\mathbf{W} \equiv [W_1\ W_2 \cdots W_{N_1}]$$

be the row vector with all quadrature weights relevant on $[0, 1]$ (there are $10N_1$ weights in **W**).

The scalar product

$$\mathbf{G} \cdot \mathbf{W}$$

gives the CGL10 approximation to the integral $\int_0^1 g(v)\,dv$.

For the interval $[1, t_c]$, we define analogous quantities: T_i, $i = 1, 2, ..., N_2$ is the row vector of nodes on subinterval i on $[1, t_c]$. This allows the construction of

$$\mathbf{T} \equiv [T_1\ T_2 \cdots T_{N_2}],$$

which has $10N_2$ elements, and the computation of

$$\mathbf{F} \equiv f(\mathbf{T}).$$

Let C_i denote the row vector of quadrature weights appropriate to subinterval i on $[1, t_c]$, and let

$$\mathbf{C} \equiv [C_1\ C_2 \cdots C_{N_2}]$$

be the row vector with all $10N_2$ quadrature weights relevant on $[1, t_c]$. Then the scalar product

$$\mathbf{F} \cdot \mathbf{C}$$

gives the CGL10 approximation to the integral $\int_1^{t_c} f(t)\,dt$.

Of course, we then have

$$\mathcal{F}_{\frac{1}{2}}(x) \approx \mathbf{G} \cdot \mathbf{W} + \mathbf{F} \cdot \mathbf{C}.$$

6. Similar Integrals

Other Fermi-Dirac integrals of importance are those of order $-\frac{1}{2}$ and $\frac{3}{2}$. Higher derivatives of the integrands in both of these integrals exhibit divergence on $[0, 1)$, as seen with $\mathcal{F}_{\frac{1}{2}}(x)$. As before, this pathology can be cured using the change of variable (9), which gives

$$\mathcal{F}_{-\frac{1}{2}}(x) = \frac{1}{\sqrt{\pi}} \int_0^1 \frac{2}{e^{v^2 - x} + 1}\,dv + \frac{1}{\sqrt{\pi}} \int_1^\infty \frac{1}{\sqrt{t}(e^{t-x} + 1)}\,dt,$$

$$\mathcal{F}_{\frac{3}{2}}(x) = \frac{4}{3\sqrt{\pi}} \int_0^1 \frac{2v^4}{e^{v^2 - x} + 1}\,dv + \frac{4}{3\sqrt{\pi}} \int_1^\infty \frac{(\sqrt{t})^3}{e^{t-x} + 1}\,dt.$$

Since these integrals are not relevant to our other work in photovoltaic simulation, we have not approximated them here. Nevertheless, we believe that CGL10 will be just as successful for these integrals as it has proved to be for $\mathcal{F}_{\frac{1}{2}}(x)$.

7. Conclusion

We have used composite ten-point Gauss-Legendre quadrature to approximate the Fermi-Dirac integral of order $\frac{1}{2}$ for $x \in [-100, 100]$. It was necessary to transform the integral to deal with a divergence on the interval $t \in [0, 1]$, and we replaced the infinite upper limit in the integral with a finite cutoff. We have determined the minimum number of subintervals necessary for the composite quadrature algorithm to yield approximations of various accuracies. For the most

accurate approximation (a relative error of no more than 10^{-14}) only 89 subintervals were necessary.

We have not investigated the use of higher-order CGL quadrature. Perhaps composite twenty-point GL quadrature would require fewer subintervals than CGL10; indeed, a rudimentary test has indicated that composite three-point GL quadrature requires far more subintervals than CGL10 to achieve similar accuracy. It is not clear to us which is more desirable - one or two subintervals, each with many nodes, or many subintervals, each with a few nodes? Our investigation here, with CGL10, has been performed in an attempt to study the nature of the approximation with what we believe is something of a compromise - not too many subintervals, each with not too many nodes. Exactly what the optimal case is, though, must be the subject of a future study.

Physical computing times have been measured, and indicate that the algorithm is fast, even for high accuracy. We acknowledge that our data in this regard is platform-dependent, but it is not unreasonable to assume that similar speed would be obtained on other modern platforms. Indeed, since our code is interpreted (we work in the Matlab environment), and not compiled, it may well be possible to develop an executable version of the algorithm, that is, in fact, much faster.

8. Appendix

8.1. Nodes and weights for GL10

Table 2. Nodes $\widetilde{t}_i$ and weights c_i for ten-point Gauss-Legendre quadrature on $[-1, 1]$. The high precision shown is due to these values having being determined using computer algebra software with 32-place accuracy

$\widetilde{t}_i$	c_i
−0.97390652851717172007796401208445	0.06667134430868813759356880989334
−0.86506336668898451073209668842349	0.14945134915058059314577633965770
−0.67940956829902440623432736511487	0.21908636251598204399553493422815
−0.43339539412924719079926594316579	0.26926671930999635509122692156947
−0.14887433898163121088482600112972	0.29552422471475287017389299465134
0.14887433898163121088482600112972	0.29552422471475287017389299465134
0.43339539412924719079926594316579	0.26926671930999635509122692156947
0.67940956829902440623432736511487	0.21908636251598204399553493422815
0.86506336668898451073209668842349	0.14945134915058059314577633965770
0.97390652851717172007796401208445	0.06667134430868813759356880989334

8.2. Derivation of (7)

From (5), the approximation error in GL10 on $[a, b]$ is

$$\Delta_{10} = \frac{f^{(20)}(\xi)}{20!}\int_a^b [P_{10}(t)]^2\,dt$$

$$= \frac{f^{(20)}(\xi)}{20!}\int_a^b \prod_{i=1}^{10}(t - t_i)^2\,dt,$$

where the t_i are the nodes on $[a, b]$ determined from (4). If we make the substitutions

$$h \equiv \frac{b-a}{11},$$

$$t = a + sh,$$

$$t_i = a + \sigma_i h,$$

where $s \in [0, 11]$ is a continuous variable, σ_i is an appropriate constant for each i, and h is the average length of the 11 subintervals into which $[a, b]$ is subdivided by the ten nodes t_i, we have

$$\Delta_{10} = \frac{f^{(20)}(\xi)h^{21}}{20!}\int_0^{11}\prod_{i=1}^{10}(s - \sigma_i)^2\,ds$$

for the approximation error on $[a, b]$. If, in CGL10, the interval of integration $[A, B]$ is subdivided into N subintervals of equal length D, then the total approximation error on $[A, B]$ is

$$\Delta_{\mathrm{CGL10}} = \sum_{j=1}^{N}\frac{f^{(20)}(\xi_j)h^{21}}{20!}\int_0^{11}\prod_{i=1}^{10}(s - \sigma_i)^2\,ds$$

$$\le Nh^{21}\frac{\max_{[A,B]}|f^{(20)}(x)|}{20!}\left|\int_0^{11}\prod_{i=1}^{10}(s - \sigma_i)^2\,ds\right|$$

and, since $D = 11h$ and $ND = B - A$, we have

$$\Delta_{\mathrm{CGL10}} \le \frac{(B-A)D^{20}}{11^{21}} \frac{\max\limits_{[A,B]} | f^{(20)}(x) |}{20!} \left| \int_0^{11} \prod_{i=1}^{10} (s - \sigma_i)^2 \, ds \right|,$$

indicating the proportionality referred to in (7).

References

[1] S. S. Li, Semiconductor Physical Electronics, Plenum Press, New York, 1993, pp. 88-94.

[2] F. G. Lether, Analytical expansion and numerical approximation of the Fermi-Dirac integrals $F_j(x)$ of order $j = -1/2$ and $j = 1/2$, J. Sci. Comput. 15(4) (2000), 479-497.

[3] N. Mohankumar, T. Kannan and S. Kanmani, On the evaluation of Ferni-Dirac integral and its derivatives by IMT and DE quadrature methods, Comput. Phys. Comm. 168 (2005), 71-77.

[4] W. J. Cody and H. C. Thacher, Rational Chebyshev approximations for Fermi-Dirac integrals of orders $-1/2$, $1/2$ and $3/2$, Math. Com. 21 (1967), 30-40.

[5] B. I. Reser, Numerical method for calculation of the Fermi integrals, J. Phys.: Condensed Matter 8 (1996), 3151-3160.

[6] D. Kincaid and W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, 3rd ed., Brooks/Cole, Pacific Grove, 2002, pp. 492-498.