DRAFT: Fun with Filon Quadrature — a Computer Algebra Perspective

Richard Fateman
Computer Science
University of California
Berkeley, CA, USA

August 19, 2007

Abstract

Filon quadrature, useful for oscillatory integrals, is easily implemented in a computer algebra system (CAS) using exact or approximate arithmetic. Although quadrature is almost always used in the context of approximation, we see that various properties are exhibited by running an exact algorithm on exact *symbolic* inputs. This experimental approach to the mathematics allows us to prove the implementation of the algorithm has the expected mathematical correctness properties, and is therefore more likely to be, itself, a correct implementation.

1   Introduction

Filon[2, 3] quadrature is supposed to be exact for integrands of the form $f(x)\sin(mx)$ where $f(x)$ is a polynomial of degree 2 or less. One parameter, namely the number of panels used for the formula, should be irrelevant if the formula is exact.

2   An exact formula

Here is an algorithm representing an exact Filon formula for the sin integral from 0 to 1, using the syntax of the Macsyma CAS (equivalent to the free open-source Maxima). The coding follows the documentation for ACN Algorithm 353 [3], and is much simpler than the ACM FORTRAN code (The FORTRAN has various limitations because of its fixed precision, but is also complicated by the inclusion of code related to bounding error.)

```
filonse(f,m,p) :=  /* Filon quadrature, Sin, Exact, f(x)*sin(m*x), from 0 to 1, p panels,*/
    block([h : 1/(2*p),k : m,theta,s2p,s2pm1,alf,bet,gam,f1],
          theta : k*h,
          s2p : block([sum : 0],
                    for i from 0 thru p do
                        sum : sum+apply(f,[2*h*i])*sin(2*k*h*i),
                    f1 : apply(f,[1]),
                    sum-f1*sin(k)/2),
          s2pm1 : block([sum : 0],
                      for i thru p do
                          sum : sum
```

```
                      +apply(f,[h*(2*i-1)])
                       *sin(k*h*(2*i-1)),
               sum),
        alf : 1/theta+sin(2*theta)/2/theta^2
           -2*sin(theta)^2/theta^3,
        bet : 2*((1+cos(theta)^2)/theta^2
              -sin(2*theta)/theta^3),
        gam : 4*(sin(theta)/theta^3-cos(theta)/theta^2),
        h*(alf*(apply(f,[0])-f1*cos(k))+bet*s2p+gam*s2pm1))
```

Other versions of this algorithm can be found elsewhere[1], which benefit by insisting relatively early in the calculation that all data will be a specific precision floating point. Here we proceed "exactly" as far as possible.

If we wish to integrate $(3x^2 + 4) * \sin(100x)$ from 0 to 1, an appropriate use of the function would be `filonse(lambda([x],3*x^2+4),100,3)` where the final parameter 3 is just specifying how many panels to use for subdividing the range of integration.

The filon formula should be exact, regardless of the number of panels, for any quadratic function, so `filonse(lambda([x],q*x^2+r*x+s),100,3)` should be exact. Invoking this command produces a large expression which can, by using a few additional commands[1] be reduced to

$$-\frac{(5000 \cos 100 - 5000) \, s + (5000 \cos 100 - 50 \sin 100) \, r + (-100 \sin 100 + 4999 \cos 100 + 1) \, q}{500000}.$$

This formula *is independent of the number of panels*, as can be seen by trying the same command with (say) 10 instead of 3 as the last parameter. This shows that the Filon formula for certain cases deduces exact symbolic definite integrals and could be used as a (possibly much faster) alternative to the usual CAS program. The conventional symbolic indefinite integration would most likely find the antiderivative, and then after checking for singularities would apply the "fundamental theorem of integral calculus", implemented by substitution of bounds.

Converting the above formula to a more compact (although now approximate) result, we learn that a formula for the integral is

$$0.00138 \, s - 0.00867 \, r - 0.00872 \, q$$

3 What else can be done with this algorithm?

Running the same experiment with an arbitrary cubic function shows that the formula depends on the number of panels, and thus this Filon formula is not exact for higher degree polynomials.

How does the argument of the sin enter into the formula? Consider `filonse(lambda([x],r*x+s),m,2)`, a command which asks to integrate $(rx + s) * sin(mx)$ from 0 to 1, using 2 panels. The command results in a large expression, but one which can, in a few simplification commands, be reduced to this:

$$-\frac{(m \cos m - m) \, s + (m \cos m - \sin m) \, r}{m^2}$$

Naturally, this formula is exact for any values of r, s, and non-zero m^2

---

[1]The commands "expand", "trigsimp" and "trigreduce".

[2]Considering $m = 0$ we could compute the limit of this expression and get the right answer, 0; alternatively, looking at the origin of the problem, a constant 0 integrand results in a 0 answer.

Could we also allow for a symbolic "number of panels"? In the algorithm as stated, no. We use p as a limit in an iterative loop. Another form of the algorithm, where we use an explicit "sum" from 0 to p which can, at least in principle, be simplified without being given an explicit p, would make such a general statement possible. We have done so with other algorithms (in particular a symbolic Simpson's rule), but this formula is more daunting.

We can leave parts of the formula even more symbolic. For example, with $p = 1$ (which actually means there are two panels), we can feed the algorithm an arbitrary function f and an arbitrary frequency m and determine the points at which f, sin, cos are evaluated. The answers being f evaluated at 0, 0.5, 1; sin and cos at $m/2$ and m. After some manipulation of the resulting expression, we can condense some of the pieces, eliminate some redundant subexpressions, and display the formula as

$$\frac{(3\,f\,(1) - 4\,f\,(0.5) + f\,(0))\,m\,\sin m + \left(-f\,(1)\,m^2 + 4\,f\,(1) - 8\,f\,(0.5) + 4\,f\,(0)\right)\cos m + f\,(0)\,m^2 - 4\,f\,(1) + 8\,f\,(0.5) - 4\,f}{m^3}$$

4   Conclusion

Writing, debugging, testing scientific software can benefit from symbolic computation. Symbolic execution of Filon-style quadrature programs provides a neat example of such testing.

References

[1] R. Fateman "Numerical Integration for Oscillatory Integrands: a Computer Algebra Perspective, Draft.

[2] L.N.G. Filon, "On a quadrature formula for trigonometric integrals," *Proc. Roy Soc. Edinburgh 49* 1928-29 38.

[3] S.M. Chase and L.D. Fosdick, "An algorithm for Filon Quadrature," *CACM 12 no 8* August 1969 453–457.

[4] K.C. Chung, G.A. Evans, J.R. Webster, "A method to generate generalized quadrature rules for oscillatory integrals," *Applied Numer. Math. 34 (2000)* 85—93.

[5] G.A. Evans, "How Integration by Parts Leads to Generalised Quadrature Methods," *Intern. J. Computer Math.,* 2003, vol 80(1), 75—81.

[6] R.J. Fateman, "When is a function oscillatory?" draft, June, 2007.

[7] R.J. Fateman, "Computer Algebra and Numerical Integration, Proc. SYMSAC'81 (ISSAC), August, 1981, 228–232.

[8] K.O. Geddes, "Numerical Integration in a Symbolic Context" Proc. SYMSAC-86, (ISSAC) July, 1986, 185–191.