Research Statement

James Bremer
Department of Mathematics, University of California, Davis
Email: bremer@math.ucdavis.edu
Webpage: https.math.ucdavis.edu/~bremer

I work in the field of numerical analysis, which is the study of analytic processes under the severe information-theoretic constraints imposed by the use of computers and finite precision arithmetic. My focus is on the numerical solution of differential and integral equations, particularly those that model wave propagation.

Owing to their fundamental importance in analysis and mathematical physics, as well as their many applications, the numerical solution of such equations has been the focus of intense research for several decades. Yet, there are still many cases of great interest in which dramatic improvements in efficiency, accuracy and reliability are both possible and highly desirable.

Below, I briefly discuss some of my recent research projects.

1. The variable coefficient Helmholtz equation in the high-frequency regime

Much of my recent work concerns the numerical solution of second order differential equations of the form

\[ y''(t) + q(t)y(t) = 0 \quad \text{for all } a < t < b. \]  

(1)

The solutions of (1) oscillate in regions in which the coefficient \( q \) is positive, and they do so with frequencies which increase with the magnitude of \( q \). Among other things, this means that standard solvers for equations of this form are extremely inefficient when \( q \) is positive and of large magnitude. Indeed, when such methods are applied to (1), the cost to compute a solution is \( O(\lambda) \) with \( \lambda \) the number of oscillations of the solution on the interval \((a, b)\).

In [4], I introduce a highly effective numerical method for the solution of equations of the form (1) which runs in time which is independent of the frequency oscillation of its solutions. It proceeds by constructing a nonoscillatory phase function \( \alpha \) such that the functions

\[ \frac{\cos(\alpha(t))}{\sqrt{\alpha'(t)}} \quad \text{and} \quad \frac{\sin(\alpha(t))}{\sqrt{\alpha'(t)}} \]

form a basis in the space of solutions of (1). In [8] and [11], Vladimir Rokhlin, his former student Zhu Heitman and I develop estimates on the complexity of \( \alpha \). More explicitly, we show that if a certain function \( p \) derived from \( q \) has a Fourier transform which decays exponentially, then (1) admits a phase function \( \alpha \) such that

\[ \alpha(t) = \tilde{\alpha}(t) + O(\exp(-c\lambda)) \]

with \( \tilde{\alpha} \) a function which extends to the real line and has a Fourier transform which decays at roughly the same rate, \( c \) is constant whose magnitude also depends on the rate of decay of the Fourier transform of \( p \), and \( \lambda \) is the number of oscillations of the solutions of Equation (1) on the interval \((a, b)\). A Fortran 90 implementation of the algorithm of [4] is available on my GitHub page. Figure 1 shows a plot of the nonoscillatory phase function it generated for the equation (1) when

\[ q(t) = \lambda^2 \frac{1}{0.1 + t^2} + \lambda^{3/2} \frac{\sin(4t)^2}{(0.1 + (t - 0.5)^2)^4} \]

(2)
and \( \lambda = 10^3 \). Figure 2 contains a plot of one of the solutions of (1) represented by the phase function shown in Figure 1.

![Figure 1: A plot of the nonoscillatory phase function for Equation (1) with \( q \) defined via (2) and \( \lambda = 10^3 \).](image1)

The algorithm took about \( 10^{-3} \) seconds to construct this phase function on my run-of-the-mill laptop computer and evaluating one of the solutions of (1) at a point using \( \alpha \) requires around \( 10^{-7} \) seconds. The time required to construct \( \alpha \) and the time required to evaluate the solutions of (1) does not increase with \( \lambda \). The only input to the code is a procedure for evaluating the coefficient \( q \).

![Figure 2: A plot of one of the solutions of Equation (1) with \( q \) defined via (2) and \( \lambda = 10^3 \).](image2)

The algorithm of [4] is superficially related to techniques for the numerical solution of ordinary differential equations via Magnus expansions (see, for instance, [14, 12, 13]). However, Magnus expansions
for equations of the form (1) converge at a rate which depends on the magnitude of the coefficient \( q \), with the consequence that most numerical methods based on them run in time which increases with the frequency of oscillations of the solutions. There are a number of modified Magnus expansion methods (such as that described in Section 5 of [12]) which attempt to accelerate convergence through preconditioning via the fundamental matrix of an equation with known highly oscillatory solutions. The principal (but not only) difficulty with such methods is they require the evaluation of a large number of highly oscillatory integrals, which can only be carried out efficiently in certain cases.

2. Rapid numerical evaluation of special functions and their roots

In [3], I describe a method, based on nonoscillatory phase functions, for evaluating the associated Legendre functions \( P_{\mu}^{\nu} \) and \( Q_{\mu}^{\nu} \). It runs in time independent of degree \( \nu \) and order \( \mu \), and has been extensively tested for degrees between 0 and \( 10^8 \) and orders between 0 and \( \nu \). Each evaluation takes roughly \( 10^{-6} \) seconds on my run-of-the-mill laptop computer. This is the first numerical code for the \( O(1) \) evaluation of associated Legendre functions. There are existing \( O(1) \) numerical methods for the evaluation of Legendre polynomials [1] and for Jacobi polynomials with small parameters [10], but these methods do not generalize to the case of associated Legendre functions in general. A discussion of why various well-known Liouville-Green expansions and trigonometric expansions do not provide an effective means for the numerical evaluation of associated Legendre functions can be found in the introduction of [3]. The approach of [3] can be extended to many other special functions of interest; see, for instance, [9] describes an approach to the rapid numerical evaluation of a large class of Jacobi polynomials via nonoscillatory phase functions.

The ability to rapidly compute nonoscillatory phase functions has implications beyond the rapid evaluation of special functions. Phase functions also provide an extremely efficient mechanism for evaluating the zeros of special functions satisfying second order differential equations. In [2], I introduce an algorithm for calculating the zeros of all of a large class of special functions which includes the Jacobi polynomials, Bessel functions, Mathieu functions, associated Legendre functions, generalized Laguerre polynomials, Hermite polynomials, etc. Any specified zero can be computed in \( O(1) \) operations; that is, in a number of operations independent of the parameters on which the special function depends. The approach can also be used to rapidly calculate various Gaussian quadrature rules.

3. Fast special function transforms via butterfly algorithms

Together with Haizhao Yang of the National University of Singapore, I am developing methods for the fast application of special function transforms. Our algorithms combine butterfly algorithms such as those described in [16] and [15] with techniques for the rapid numerical evaluation of special functions based on phase function methods.

Butterfly algorithms, first introduced in [17], are a class of algorithms for rapidly applying oscillatory integral operators. In order to apply an \( n \times n \) matrix discretizing such an operator, classical butterfly algorithms require a precomputation phase which takes \( O(n^2) \) operations. Once these precomputations have been completed, the matrix can be applied in \( O(n \log(n)) \) operations. A new class of butterfly algorithms such as [15] and [16] reduce the complexity of the precomputation stage to \( O(n \log(n)) \) operations. However, they require an \( O(1) \) method for the evaluation of entries of the discrete operator being applied.

Phase function methods provide such mechanisms for a large class of special functions. By coupling phase function methods with modern butterfly algorithms, we will obtain asymptotically optimal
and extremely efficient algorithms for the application of a large class of special function transforms. In [9], we apply this methodology to obtain fast algorithms for the rapid computation of Jacobi expansions. Our code is available at my GitHub page. We are now extending the method of [9] to spherical harmonic transforms. Spherical harmonic transforms, of course, have applications in astronomy, meteorology, physics, chemistry, spectral methods for the solution of partial differential equations, and many other areas.

4. Fast solvers for integral operators given on surfaces with complex geometry

A large class of linear elliptic boundary value problems can be reformulated as integral equations. To give one example, the Lippmann-Schwinger equation is a particular integral equation reformulation of the variable coefficient Helmholtz equation

\[ \Delta u(x) + q(x)u(x) = 0. \] (3)

There are a number of motivations for reformulating partial differential equations as integral equations. In the case of constant coefficient equations such as

\[ \Delta u(x) + \lambda^2 u(x) = 0 \] (4)

doing so reduces the dimensionality of the problem by one. However, there are benefits to this approach even in the case of variable coefficient equations such as (3). Perhaps the principal one is that the reformulation of linear elliptic boundary value problems generally leads to well-conditioned integral operators which act on spaces of square integrable functions. Producing high-accuracy discretizations of such operators is often easier than producing high-accuracy discretizations of the corresponding differential operators, which act on various Sobolev spaces. For this reason, among others, integral equation methods are one of the most widely used techniques for the numerical solution of problems in scattering theory.

The integral operators which arise from linear elliptic boundary value problems have singular kernels. In the case of such operators given on volume regions and curves, high-accuracy and extremely efficient discretization methods are available. However, methods for the discretization of singular integral operators on surfaces, particularly surfaces which are themselves singular, are considerably less well-developed.

In [6], Zydrunas Gimbutas of NIST Boulder and I introduce a highly efficient method for discretizing a large class of singular integral operators on surfaces. It uses a relatively small table of precomputed generalized Gaussian quadratures constructed via the numerical method of [7] to efficiently evaluate the singular integrals which arise in the course of discretizing integral operators on surfaces. The method achieves high-order convergence and is extremely accurate. Moreover, while there are other methods such as quadrature by expansion (QBX) and those based on smooth partitions of unity which are faster in certain cases, the approach of [6] is one of the most widely applicable and robust schemes available. Although not designed explicitly for this purpose, it is particularly effective in the case of surfaces with edge and corner points such as those shown in Figure 1.

In [5], Per-Gunnar Martinsson (Oxford University), Adrianna Gillman (Rice University) and I couple the discretization method of [6] with a fast direct solver for integral equations. This combination of a fast solver for integral equations with a high-order discretization method yields an extremely efficient mechanism for the numerical solution of a number of large-scale scattering problems. More recently, the algorithm of [6] has been used in [18] in order to solve Laplace-Beltrami equations given on surfaces.
Figure 3: Two surfaces with edge and corner points. The algorithm of [6] provides a means for the highly accurate discretization of a large class of singular integral operators given on surfaces of this type. In [5], a fast direct solver is coupled with the discretization method of [6] to solve integral equations on domains such as these.

5. References


